New Insights into Divide-and-Conquer Attacks on the Round-Reduced Keccak-MAC

Chen-Dong Ye[1] and Tian Tian[1,*]

[1]National Digital Switching System Engineering & Technological Research Center, P.O. Box 407, 62 Kexue Road, Zhengzhou, 450001, China.
[*]tiantian_d@126.com

Abstract: Keccak is the final winner of SHA-3 competition and it can be used as message authentic codes as well. The basic and balanced divide-and-conquer attacks on Keccak-MAC were proposed by Dinur et al. at Eurocrypt 2015. The idea of cube attacks is used in the two attacks to divide key bits into small portions. In this paper, by carefully analysing the mappings used in Keccak-MAC, it is found that some cube variables could divide key bits into smaller portions and so better divide-and-conquer attacks are obtained. Furthermore, in order to evaluate the resistance of Keccak-MAC against divide-and-conquer attacks based on cubes, we theoretically analyse the lower bounds of the complexities of divide-and-conquer attacks. It is shown that the lower bounds of the complexities are still not better than those of the conditional cube tester proposed by Senyang Huang et al.. This indicates that Keccak-MAC can resist the divide-and-conquer attack better than the conditional cube tester. We hope that these techniques still could provide some new insights on the future cryptanalysis of Keccak.

Keywords: Keccak-MAC, Divide-and-Conquer Attacks, Cube Attacks

1. Introduction

Keccak [1] is a family of sponge functions consisting of seven permutations, say Keccak-$f[b]$, with $b = 25, 50, 100, 200, 400, 800$ or 1600. The largest permutation, i.e., Keccak-$f[1600]$, was selected as the new SHA-3 standard by the U.S. National Institute of Standards and Technology in 2012, and so its security property has attracted much attention.

In keyless modes, Keccak is used as a hash function, and so attacking techniques against hash functions like preimage attacks and collision attacks etc. have been studied for Keccak by some researchers. In [2] and [3], the preimage attack and the second preimage attack were implemented for round-reduced Keccak. In [4], a collision attack on the 2-round and a near collision attack on the 3-round $\lfloor\text{Keccak}\rfloor_{224}$ and $\lfloor\text{Keccak}\rfloor_{256}$ were proposed. In [5], the attacking rounds of $\lfloor\text{Keccak}\rfloor_{224}$ and $\lfloor\text{Keccak}\rfloor_{256}$ were improved to 4 and 5 for collision attacks and near collision attacks respectively. In [6], based on the propagation analysis, the authors constructed distinguishers on the 6-round Keccak. Other techniques such as zero-sum distinguishers could also be applied to analyze Keccak. In [7], several zero-sum partitions of size 2^{1586} for 20-round Keccak were found. In [8] the authors gave some zero-sum partitions of size 2^{1590} for the the full 24-round Keccak, and the size of the zero-sum partitions were further reduced to 2^{1579} in [9].

In keyed modes, Keccak is used as stream ciphers, message authentication codes (MAC) denoted by Keccak-MAC and authenticated encryption schemes (see Keyak [10]). For these keyed

modes, the published cryptanalytical results are all given by cube attacks. In [11], the author proposed three different attacks on stream ciphers based on Keccak, Keccak-MAC and Keyak. Firstly, the authors used cube attacks to break up to some 6-round Keccak variants. Secondly, cube testers were applied to predict output bits. Finally, the authors showed how to utilize the divide-and-conquer attack based on cubes to recover the key for the 6-round Keccak-MAC and theoretically analysed the complexities of the 7-round case. In [12], the authors proposed a new type of cube distinguisher called conditional cube tester to recover key bits of Keccak-MAC and Keyak. Making use of the conditional cube tester, compared with the attack in [11], the authors either decreased the complexity or increased the number of attacking rounds.

In this paper, we focus on Keccak-MAC only. With more careful analysis of the mappings used in Keccak, we make a new investigation into the divide-and-conquer attack based on cubes.

Our contributions

In this paper, we mainly focus on the divide-and-conquer attack on the round-reduced Keccak-MAC. It is known that the complexity of the divide-and-conquer attack depends on how small portions the key bits can be divided into. In [11], the authors divide the 128-bit key into two 64-bit small keys (basic attacks), which resulted in a high time complexity of the preprocessing phase. Fortunately, by analysing ρ mapping used in Keccak, we find cubes with eighteen (resp. thirty-four) effective key bits, which could divide the 128-bit key into smaller portions, for the 6-round (reps. 7-round) Keccak-MAC. This indicates that complexities of the divide-and-conquer attack could be dramatically reduced without any extra cost. Furthermore, in order to evaluate the resistance of Keccak-MAC against the divide-and-conquer attack, we theoretically analyse the lower bounds of the complexity of the divide-and-conquer attack on the round-reduced Keccak-MAC in two scenarios. In Scenario I, inspired by the attack idea in [11], we only consider such cubes that the cube sums (derived from outputting bits) are only dependent on one of $A[0][0]$ and $A[1][0]$. It is proved that in Scenario I for the 6-round (resp. 7-round) Keccak-MAC, the preprocessing time is lower-bounded by 2^{41} (resp. 2^{76}), the memory is lower-bounded by 2^9 (resp. 2^{12}), and the online time is lower-bounded by 2^{40} (resp. 2^{75}). Since there is no proof that an optimal cube for the divide-and-conquer attack should satisfy such restriction, we further remove the restriction on cubes in Scenario II. This means that in Scenario II, the cube sums of a cube may be dependent on both of $A[0][0]$ and $A[1][0]$, and we only care about whether the total number of effective key bits related to the cube could attain the minimum. Moreover, in Scenario II, we even neglect the specific rotation constants used in ρ, to see how small the attacking complexity could eventually be. It is proved that in Scenario II for the 6-round (resp. 7-round) Keccak-MAC, the preprocessing time is lower-bounded by 2^{40} (resp. 2^{73}), the memory is lower-bounded by 2^8 (resp. 2^9), and the online time is lower-bounded by 2^{39} (resp. 2^{73}). A comparison of our work with the best previously known attacks on the round-reduced Keccak-MAC is presented in Table 1. It can be seen that even in such ideal scenario as scenario II, the attacking complexity is no better than that given in [12]. Thus the Keccak-MAC can resist the divide-and-conquer attack better than the conditional cube tester proposed in [12].

This paper is organized as follow. In Section 2, we introduce Keccak, Keccak-MAC and cube attacks briefly, review the divide-and-conquer attack presented in [11] and introduce our attack model in detail. In Section 3, we show that some cubes could divide the 128-bit key into smaller portions for 6- and 7-round Keccak-MAC. In Section 4, we theoretically analyse the lower bounds of the complexities of the divide-and-conquer attack on 6- and 7-round Keccak-MAC in two different scenarios. Finally, we conclude this paper in Section 5.

Table 1 Summary of complexities of the pervious attacks and our work

Rounds	Attacks	TC in PP	MC in PP	TC in OP
6	The Basic Attack in [11]	2^{96}	2^{64}	2^{32}
6	The Balanced Attack in [11]	2^{64}	2^{32}	2^{66}
6	The Balanced Attack in Subsection 3.1	2^{45}	2^{13}	2^{45}
6	The Attack in [12]	2^{40}	negligible	2^{40}
7	The Balanced Attack in [11]	2^{97}	2^{64}	2^{98}
7	The Balanced Attack in Subsection 3.2	2^{84}	2^{67}	2^{84}
7	The Attack in [12]	2^{72}	negligible	2^{72}

TC in PP: the time complexity of the preprocessing phase
MC in PP: the memory complexity of the preprocessing phase
TC in OP: the time complexity of the online phase

2. Prelimiaries

2.1. Keccak

Keccak is designed based on the sponge construction. Keccak sponge functions have seven permutations, indicated by Keccak-$f[b]$, where $b = 25 \times 2^l$ is called the *width* and l ranges from 0 to 6. The permutation Keccak-$f[b]$ is over $\mathbf{F}_2^b$ and consists of a family of operations on a b-bit state A. The internal state A is usually described by a three-dimensional array over $\mathbf{F}_2$, i.e., $A[5][5][w]$, where $w = 2^l$. The names of parts of the Keccak-$f[b]$ state are illustrated in Fig. 1, which help simplify the description of our analysis. Furthermore, to remove any vagueness in the following discussions, the specific location of each lane is shown in Fig. 2, where each square represents a lane (w bits) of the internal state.

Keccak-$f[b]$ is an iterated permutation consisting of a sequence of n_r rounds $R(A, RC[i_r])$ $(0 \leq i_r \leq n_r - 1)$, where A is a state and $RC[i_r]$ is a round constant. A round consists of five mappings, namely $\theta, \rho, \pi, \chi, \iota$, which is described in the form of pseudo-code, see Table 2. All these n_r rounds are exactly the same except the round constants $RC[i_r]$. Note that all the operations on indices in the pseudo-code are done modulo 5.

The sponge function denoted by Keccak$[r, c]$ is built with Keccak-$f[b]$, muti-rate padding and the bitrate r. The value of $c = b - r$ is called the capacity. The b-bit internal state is divided into two parts. One part includes the first r bits of the internal state, and the other part includes the remaining c bits. The message is padded with $10 * 1$ and divided into r-bit blocks. The initial state

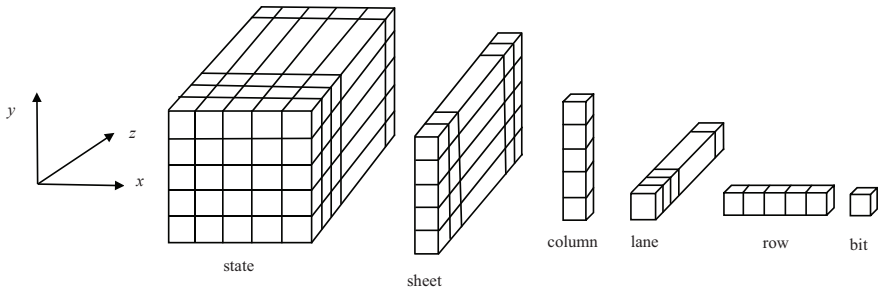

Fig. 1. *Illustration of Keccak-f internal state*

[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]

Fig. 2. *Lane coordinates*

Table 2 Pseudo-code of a round permutation of Keccak-$f[b]$

Round(A, $RC[i_r]$)
{
θ step
$C[x]$ = $A[x][0]$ xor $A[x][1]$ xor $A[x][2]$ xor$A[x][3]$ xor$A[x][4]$, for all x in $(0 \dots 4)$
$D[x]$ = $C[x-1]$ xor $(C[x+1] <<< 1)$, for all x in $(0 \dots 4)$
$A[x][y]$ = $A[x][y]$ xor $D[x]$, for all (x, y) in $(0 \dots 4,0 \dots 4)$
ρ step
$A[x][y]$ = $A[x][y] <<< r[x][y]$, for all (x, y) in $(0 \dots 4,0 \dots 4)$
π step
$B[y][2 \cdot x + 3 \cdot y]$ = $A[x][y]$, for all (x, y) in $(0 \dots 4,0 \dots 4)$
χ step
$A[x][y]$ = $B[x][y]$ xor $((notB[x+1][y])$ and $B[x+2][y])$, for all (x, y) in $(0, \dots 4,0 \dots 4)$
ι step
$A[0][0]$ = $A[0][0]$ xor $RC[i_r]$
return A
}

is set to be zeros. The whole procedure includes two phases. During the first phase, which is called the absorbing phase, a r-bit message block is OXRed with the first r bits of the internal state every time. After absorbing all the message blocks, it moves to the second phase, which is called the squeezing phase. During this phase, it runs over the internal state iteratively and return the first r bits as output till the desired length of message digest is satisfied.

The default value for Keccak sponge function is $b = 1600$, $c = 576$, and $n_r = 24$. For more information about Keccak sponge function, the readers can refer to [1].

2.2. Keccak-MAC

The Keccak sponge function, say Keccak$[r, c]$, can be used in the keyed modes, such as MACs. Keccak-MAC is constructed simply by prepending the secrete key to the message to calculate a tag. In this paper, we will focus on MACs of short messages such that the internal permutation Keccak-$f[r+c]$ is applied only once. The construction of Keccak-MAC working on a single block is shown in Fig. 3.

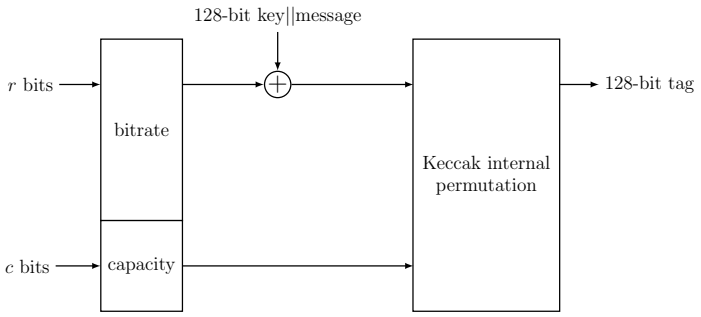

Fig. 3. *Keccak-MAC*

2.3. Cube attacks

The cube attack was proposed by Itai Dinur and Adi Shamir in [13]. It could be applied to a wide range of cryptographic primitives, such as stream cipher Trivium [13], block cipher KATAN [14], and hash function Keccak [11]. In this subsection, we briefly introduce the idea of cube attacks, and readers could refer to [13] for more details.

In the cube attack against a cipher, an output bit z of the cipher is seen as a blackbox polynomial on secrete variables SV and public variables PV, say $z = f(SV, PV)$. For a subset of public variables, say $I = \{v_i | 0 \leq i \leq k-1\}$, z can be rewritten as

$$z = T \cdot g(SV, PV \setminus C) \oplus h(SV, PV),$$

where $T = \prod_{i=0}^{k-1} v_i$, any term of h is not divisible by T, and $PV \setminus I$ represents the public variables which do not belong to I. Then the sum of f over all values of the variables in I is given by

$$\sum_{(v_0,\cdots,v_{k-1})\in C} f(SV, PV) = g(SV, PV \setminus I),$$

where $C = \{0,1\}^k$. If g is of low degree about SV (usually linear), then it is useful for recovering the value of secret variables. In the cube attack, the elements in I are called cube variables, C which collects all possible values of cube variables is called a k-dimensional **cube**, and g is called the **superpoly** of C (or of I).

The cube attack includes two phases, namely the preprocessing phase and the online phase.

- The preprocessing phase

  The preprocessing phase does not depend on the value of secret key bits and needs to be carried out only once for a cipher. We need to find a number of cubes such that their superpolies are of low degree, particularly linear expressions. Since generally the algebraic normal form of $f(SV, PV)$ is very complex, this is accomplished by black polynomial test.

- The online phase

  In this phase, an attacker online calculates the value of the superpoly for each cube found in the preprocessing phase (public variables not in a cube are fixed to be 0). This leads to a system of equations in secrete key variables. Solving the system of equations, some or all key bits could be recovered.

2.4. The previous divide-and-conquer attack

In [11], the authors proposed two divide-and-conquer attacks on the 6-round Keccak-MAC. The main idea of the two attacks was to find cubes whose cube sums of all output bits only depend on a part of key bits. A cube in $A[2][2]$ and $A[2][3]$ was proven to have the following two properties.

Property 1. The cube sum of each output bit after six rounds does not depend on the value of $A[1][0]$.

Property 2. The cube sums of the output bits after six rounds depend on the value of $A[0][0]$.

Exploiting the cube given above, the authors presented a basic divide-and-conquer attack on the round-reduced Keccak-MAC. The attack included two phases, namely, the preprocessing phase and the online phase.

- The preprocessing phase
  1. Set the state bits to 0 except $A[0][0]$ and the cube variables.
  2. For each of the 2^{64} possible value of $A[0][0]$,

     (a) calculate the cube sums after 6 rounds for all output bits. Store the cube sums and the corresponding value of $A[0][0]$ in the list L.
- The online phase
  1. Calculate the cube sums for all output bits and search them in L.
  2. For each match in L, retrieve $A[0][0]$ and store all of its possible values.

Similarly, by selecting another cube in $A[4][2]$ and $A[4][3]$, the possible values of $A[1][0]$ could be retrieved. Finally, in order to recover the full key, all combinations of the candidates independently obtained for $A[0][0]$ and $A[1][0]$ were enumerated and tested. The time complexity in the preprocessing phase was 2^{96} and that of the online phase was 2^{32}. Thus, the total time complexity was 2^{96}, the data complexity was 2^{32}, and the memory complexity was 2^{64}.

In this basic attack, the time complexity was dominated by the preprocessing phase, and so an improved attack called balanced attack was further proposed in [11] to tradeoff the complexity of the preprocessing and online phases. In the balanced attack, additional auxiliary variables in $A[0][1]$ were used to eliminate the dependency of the cube sums of the output bits on some of the variables of $A[0][0]$. Thus, the time complexity of the preprocessing phase could be reduced from 2^{96} to 2^{64}.

It was mentioned in Subsection 6.3 of [11] that the 6-round balanced attack could be extended to a 7-round attack by selecting a borderline cube of sixty-four variables in $A[2][2]$ and $A[2][3]$ with the data complexity 2^{64}, the time complexity 2^{97}, and the memory complexity 2^{32}.

2.5. Attack models and parameters

It is known that there are two types of attacks towards a MAC. The first one is that an adversary could forge a MAC for a message that has not been authenticated. The second one is that an adversary could recover the secret key. Obviously, the second one is much stronger. In this paper, we would concentrate on recovering the secret key for the short messages such that the internal permutation would be applied only once. More precisely, we would target at some variants of Keccak-MAC with a 1600-bit state, the capacity parameter $c = 256$, and a 128-bit key. This means that in our attack model, we could choose a 1344-bit input message and obtain its 128-bit

outputting tag. Thus, for a cube C, a cube sum could be derived by examining each 128 outputting bits, and so in the following paper, the cube sums of C refer to all 128 sums.

3. More careful investigations into the divide-and-conquer attacks on Keccak-MAC

Recall that the 128 key bits are placed in $A[0][0]$ and $A[1][0]$. Without loss of generality, we assume that $A[0][0][i] = k_i$ and $A[1][0][i] = k_{i+64}, 0 \leq i \leq 63$.

It is well known that the action of θ depends on the column parities only. This property is exploited in the previous paper [11] to make the algebraic degree of Keccak remain 1 on cube variables after the first round, by setting the cube variables in such a way that all the column parities are constants at the beginning of the first round. In this paper, we also exploit this property of θ, and we simply say that setting the cube variables satisfying the constant condition instead.

Besides, for the sake of convenience, we introduce the following definition of effective key bits of a cube.

Definition 1. For a given cube, a key bit that will be multiplied with the cube variables by χ in the ***first*** round is called an effective key bit (or variable) of the cube.

By setting the cube variables satisfying the constant condition, the set of effective key bits is actually the same as the set of key bits which the 128 cube sums of outputting bits depended on. The reason for this is as follows. Considering the n-round Keccak-MAC, the degree of the Boolean function f representing an output bit would reach 2^n after n rounds. But if the cube variables are set to satisfy the constant condition initially, then as discussed above cube variables will not multiply with each other after the first round and so the degree of f on cube variables would be at most 2^{n-1}. Hence, for a cube C of 2^{n-1}-variables ensuring that the column parities are constant initially, each 128 cube sums of the outputting bits depend and only depend on the key bits multiplied with the 2^{n-1} cube variables in the first round.

3.1. New results on the 6-round Keccak-MAC

In Section 6.1 of [11], the authors selected the cube variables in $A[2][2]$ and $A[2][3]$. For this cube, the authors pointed out that since the cube sums of output bits after six rounds depended on all the sixty-four bits of $A[0][0]$ (key bits), the time complexity of the preprocessing phase became as high as 2^{96}. Consequently, in Section 6.2 of [11], the authors proposed to use auxiliary variables to reduce the time complexity of the preprocessing phase. However, in the following part of this subsection, we shall show that in fact for some carefully chosen cubes, the cube sums only depend on eighteen key bits after six rounds. Thus, the time complexity and the memory complexity of the preprocessing phase could be dramatically decreased without any extra cost.

Theorem 1. For $A[0][0]$, let $C = \{v_i = A[2][2][3 \cdot i] | 0 \leq i \leq 16\} \cup \{v_{i+17} = A[2][3][3 \cdot i] | 0 \leq i \leq 14\}$ and set

$$A[2][0][3 \cdot i] = \begin{cases} v_i \oplus v_{i+17} \ (0 \leq i \leq 14) \\ v_i \ (15 \leq i \leq 16) \end{cases}$$

to keep the column parities of $A[2][*]$ constant, then the effective key bits of C are $k_2, k_5, \cdots, k_{47}, k_{60}, k_{63}$, namely, C has eighteen effective key bits.

Proof. We fix the value of $A[1][0]$ to an arbitrary constant, and symbolically represent the sixty-four bits of $A[0][0]$ as secret variables. Since we choose cube variables in $A[2][0]$, $A[2][2]$ and $A[2][3]$, the initial state is shown in Figure 4 (a) where the lane in orange(resp. pink) means that it includes secret variables(resp. cube variables). After θ, the state is shown in the Figure 4 (b). The number in each lane is the offset that it has been rotated compared with the initial state.

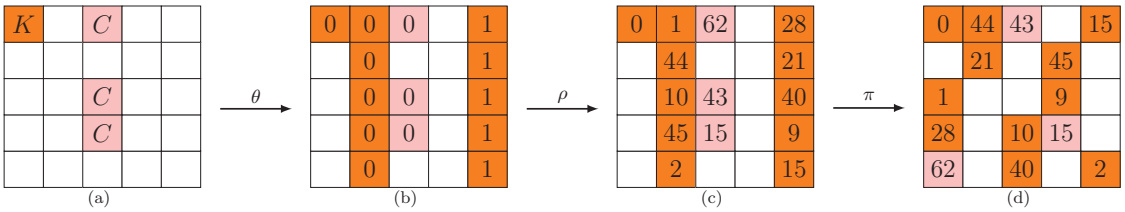

Fig. 4. *The diffusion of cube variables and key bits in one round*

Due to ρ, every lane of the state is rotated with a different offset , the state after ρ is shown in Figure 4 (c). The function of π is to reorder the lanes, thus, the state after π is shown as Figure 4 (d).

It worth noting that only three lanes ($A[0][2]$, $A[3][3]$ and $A[0][4]$ in Figure 4 (d)) contain cube variables. From Figure 4, we know that $v_i = A[2][2][3 \cdot i]$ (the initial state) will multiply with $k_{3\cdot i+62-2} = k_{3\cdot i-4}$ (Here the substraction and addition operations are done modulo 64). Similarly, we could obtain all the effective key bits, see Table 3.

Hence, for C, its effective key bits are $k_2, k_5, k_8, k_{11}, k_{14}, k_{17}, k_{20}, k_{23}, k_{26}, k_{29}, k_{32}, k_{35}, k_{38}$, $k_{41}, k_{44}, k_{47}, k_{60}$ and k_{63}, namely the number of its effective key bits is eighteen. □

Table 3 The key bits multiplied with cube variables in each lane

Lane	Key bits
$A[2][0]$	$k_2, k_5, k_8, k_{11}, k_{14}, k_{17}, k_{20}, k_{23}, k_{26}, k_{29}, k_{32}, k_{35}, k_{38}, k_{41}, k_{44}, k_{60}, k_{63}$
$A[2][2]$	$k_2, k_5, k_8, k_{11}, k_{14}, k_{17}, k_{20}, k_{23}, k_{26}, k_{29}, k_{32}, k_{35}, k_{38}, k_{41}, k_{44}, k_{47}, k_{63}$
$A[2][3]$	$k_5, k_8, k_{11}, k_{14}, k_{17}, k_{20}, k_{23}, k_{26}, k_{29}, k_{32}, k_{35}, k_{38}, k_{41}, k_{44}, k_{47}$

Similarly, we could construct cubes with eighteen effective key bits when recovering $A[1][0]$.

Theorem 2. For $A[1][0]$, let $C = \{v_i = A[4][2][19 \cdot i] | 0 \leq i \leq 16\} \cup \{v_{i+17} = A[4][0][19 \cdot i] | 0 \leq i \leq 14\}$, and set

$$A[4][1][19 \cdot i] = \begin{cases} v_i \oplus v_{i+17} \ (0 \leq i \leq 14) \\ v_i \ (15 \leq i \leq 16) \end{cases}$$

to keep the column parities of $A[4][*]$ constant, then the effective key bits of C are $k_{64}, k_{66}, k_{71}, k_{73}$, $k_{78}, k_{80}, k_{85}, k_{90}, k_{92}, k_{97}, k_{99}, k_{104}, k_{109}, k_{111}, k_{116}, k_{118}, k_{123}, k_{125}$, namely, C has eighteen effective key bits.

By sliding the cube C in Theorem 1 five times (the effective key bits are slid as well), we get another five cubes, and so we could recover all the sixty-four bits of $A[0][0]$. Similarly, by sliding the cube C in Theorem 2, we could get five cubes to recover all the sixty-four bits of $A[1][0]$. In the preprocessing phase, for each of the chosen cubes, the cube sums are calculated and stored for all the possible values of its effective key bits. Thus, in this phase, the time complexity is

$2^{18} \cdot 11 \cdot 2^{32} \approx 2^{54}$ and the memory complexity is $2^{18} \cdot 11 \approx 2^{22}$. In the online phase, we need to calculate the cube sums of the eleven chosen cubes respectively. Since the each cube contains thirty-two variables, the time complexity of the online phase is $2^{32} \cdot 11 \approx 2^{36}$. Compared with the basic attack presented in [11], our attack reduces both the time and memory complexity in the preprocessing phase by a factor of 2^{42}.

Fortunately, the auxiliary variables could also be used to balance our attack. We shall take the case of recovering $A[0][0]$ as an example to illustrate the impact of auxiliary variables. For recovering $A[0][0]$, if we use 9 auxiliary variables in $A[0][1]$, then the time complexity of the preprocessing phase is $2^9 \cdot 6 \cdot 2^{32} \approx 2^{44}$ and the memory complexity is $2^9 \cdot 6 \approx 2^{12}$. While the time complexity of the online phase is $2^9 \cdot 2^{32} \cdot 6 \approx 2^{44}$. Hence, the total time complexity of the preprocessing phase is about 2^{45}, the memory complexity is about 2^{13}, and the total time complexity of the online phase is about 2^{45}.

3.2. New results on the 7-round Keccak-MAC

It is known that the algebraic degrees of output bits will reach 128 after seven rounds. Even if the column parities are kept constant, we need 64-variable cubes to implement divide-and-conquer attacks. However, we could still find cubes like those in Theorem 1 and 2.

Theorem 3. For $A[0][0]$, let $C = \{v_i = A[2][2][3 \cdot i] | 0 \leq i \leq 32\} \cup \{v_{i+33} = A[2][3][3 \cdot i] | 0 \leq i \leq 30\}$ and set

$$A[2][0][3 \cdot i] = \begin{cases} v_i \oplus v_{i+33} \ (0 \leq i \leq 30) \\ v_i \ (31 \leq i \leq 32) \end{cases}$$

to keep the column parities of $A[2][*]$ constant, then the effective key bits of C are $k_{60}, k_{63}, k_2, k_5, k_8, \ldots, k_{62}$, $k_1, k_4, \ldots, k_{31}$, namely C has thirty-four effective key bits.

Theorem 4. For $A[1][0]$, let $C = \{v_i = A[4][2][19 \cdot i] | 0 \leq i \leq 32\} \cup \{v_{i+33} = A[4][0][19 \cdot i] | 0 \leq i \leq 30\}$, and set

$$A[4][1][19 \cdot i] = \begin{cases} v_i \oplus v_{i+33} \ (0 \leq i \leq 30) \\ v_i \ (31 \leq i \leq 32) \end{cases}$$

to keep the column parities of $A[4][*]$ constant, then the effective key bits of C are $k_{64+((19 \cdot i+16) \mod 64)} (0 \leq i \leq 33)$, namely, C has thirty-four effective key bits.

By sliding cubes in Theorem 3 and 4 (the effective key bits are slid as well), we get enough cubes to recover the 128-bit key. With the help of auxiliary variables, the time complexity in the preprocessing phase is about 2^{84}, the memory complexity is about 2^{67}, and the time complexity in the online phase is about 2^{84}.

4. Lower bounds of complexities of divide-and-conquer attacks based on cubes

It can be seen that the complexity of divide-and-conquer attacks on Keccak-MAC is largely determined by the number of effective key bits, i.e., key bits multiplied with cube variables in the first round of Keccak-MAC. Small number of effective key bits is desirable in divide-and-conquer attacks. Thus, in this section, we discuss how small portions the key could be divided into in two different scenarios.

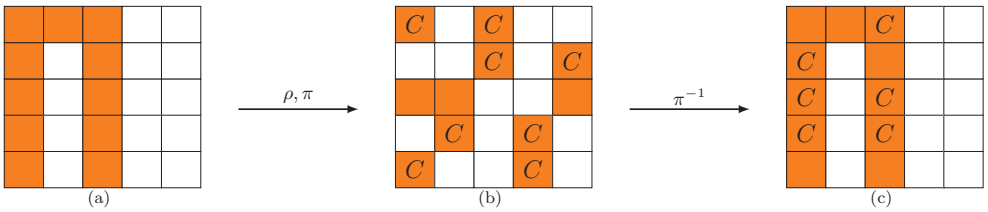

Fig. 5. *Overview of how to choose cube variables for recovering* $A[0][0]$

4.1. Scenario I: cubs with a reasonable restriction

Inspired by the attacking idea in [11], in this subsection, based on the cubes whose cube sums are independent on the value of $A[1][0]$ (resp. $A[0][0]$), we shall discuss how small portions $A[0][0]$ (resp. $A[1][0]$)could be divided into.

In the case of recovering $A[0][0]$, cubes whose cube sums do not depend on the value of the value of $A[1][0]$ after six (seven) rounds are wanted. Figure 5 (a) and (b) show the diffusion of the key bits in $A[1][0]$. In Figure 5, the orange squares represent the lanes containing key bits in bits in $A[1][0]$. For a cube, it is easy to verify that only when cube variables are in the orange lanes with "C" inside (see Figure 5 (b)) after π, its cube sums would not depend on the value of $A[1][0]$ after six (seven) rounds. Since $c = 256$ and the 128-bit key is placed in $A[0][0]$ and $A[1][0]$ in this paper, we could not choose cube variables from $A[0][0]$, $A[1][0]$, and $A[i][4]$ $(1 \leq i \leq 4)$. Hence, only cubes chosen from $A[2][i](i \in \{0, 2, 3\})$ and $A[0][i]$ $(i \in \{1, 2, 3\})$ (see Figure 5) could satisfy that its cube sums do not depend on the value of $A[1][0]$ after six (seven) rounds.

Similarly, in the case of recovering $A[1][0]$, for a cube, in order to make its cube sums do not depend on the value of $A[0][0]$ after six (seven) rounds, we could only choose cube variables from $A[4][i]$ $(0 \leq i \leq 3)$ and $A[1][i]$ $(1 \leq i \leq 2)$. There are still six lanes that we could choose cube variables.

For the 6-round Keccak-MAC, we need 32-variable cubes to complete the divide-and-conquer attack. In the case of recovering $A[0][0]$, for a cube, in order to make its cube sums do not depend on the value of $A[1][0]$, we could only choose cube variables from four lanes of $A[2][i](i \in \{0, 2, 3\})$ and $A[0][i]$ $(i \in \{1, 2, 3\})$ (two lanes are used to keep the column parities constant). Similarly, in the case of recovering $A[1][0]$, for a cube, in order to make its cube sums do not depend on the value of $A[0][0]$, we could choose cube variables from four lanes. Hence, in both cases, there would be at least one lane containing at least eight cube variables. Since the key bits in the same lane are pairwise different, for such cubes, there would be at least eight effective key bits. However, with the consideration of the rotation constants used in the ρ, we could get more precise results. Firstly, we would introduce a helpful lemma.

Lemma 1. For two different positive integers m_1 and m_2, let $t = \frac{64}{\gcd(64,(m_1-m_2))}$, and let $I = \{i_j | 0 \leq j \leq n-1\}$. Denote $I_1 = \{i_0 + m_1, \cdots, i_{n-1} + m_1\}$ and $I_2 = \{i_0 + m_2, \cdots, i_{n-1} + m_2\}$. If $I_1 = I_2$, then n is divisible by t. Here all the addition and subtraction operations are done modulo 64.

Proof. Suppose that $I_1 = I_2$. Denote

$$I^j = \{i_j + (m_1 - m_2) \cdot l | 0 \leq l \leq t-1\} \ (0 \leq j \leq t-1).$$

It is obvious that

$$I \subseteq \bigcup_{j=0}^{n-1} I^j.$$

Firstly, we shall prove that

$$I^l \subseteq I \ (0 \leq l \leq n-1).$$

Since $I_1 = I_2$, for any i_l $(0 \leq l \leq n-1)$ there exists $i_s \in I$ such that $i_l + m_1 - m_2 = i_s$ for some s. In a similar way, $i_s + m_1 - m_2 = i_l + (m_1 - m_2) \cdot 2 = i_u \in I$, for some u. Hence, $i_l + (m_1 - m_2) \cdot j \in I$ $(0 \leq j \leq t-1)$, namely $I^l \subseteq I$ $(0 \leq l \leq n-1)$. Therefore $\bigcup_{j=0}^{n-1} I^j \subseteq I$. This indicates that $\bigcup_{j=0}^{n-1} I^j = I$.

Secdonly, we shall prove that for two sets I^l and I^s, if they have a common element, then $I^l = I^s$. For the sake of convenience, we denote $i_j + (m_1 - m_2) \cdot p$ by $I^j[p]$. Without loss of generality, we assume that $I^l[0] = I^s[q]$. On one hand, for $0 \leq r \leq t - q - 1$,

$$I^l[r] = I^s[r+q].$$

On the other hand, for $t - q \leq r \leq t - 1$, since $t \cdot (m_1 - m_2)$ is divisible by 64, we have

$$I^l[r] = i_l + r \cdot (m_1 - m_2) = i_s + (r + q - t) \cdot (m_1 - m_2) = I^s[r + q - t].$$

Hence, $I^l = I^s$.

Therefore, $I = \bigcup_j I^j$,where I^j are pairwise different. Since $|I^j| = t$ holds for $0 \leq j \leq n-1$, it follows that $|I| = |\bigcup_j I^j| = \sum_j t$ is divisible by t. □

Using Lemma 1, we could obtain the lower bounds of complexities of the divide-and-conquer attacks on the round-reduced Keccak-MAC.

Theorem 5. In the case of recovering $A[0][0]$ (resp.$A[1][0]$) of the 6-round Keccak-MAC, for a 32-variable cube whose cube sums do not depend on the value of $A[1][0]$(resp. $A[0][0]$) after six rounds, the number of its effective key bits is at least nine.

Proof. We would prove that in the case of recovering $A[0][0]$, for a 32-variable cube whose cube sums do not depend on the value of $A[1][0]$ after six rounds, the number of its effective key bits is at least nine. The case of recovering $A[1][0]$ could be proved similarly.

Let C be a 32-variable cube whose cube sums do not depend on the value of $A[1][0]$ after six rounds. Assume the number of effective key bits of C is eight. Then, there would be six lanes containing eight cube variables (two lanes are used to keep the column parities constant). Since the parities of the columns containing cube variables are kept constant, the locations of cube variables in lanes of the same sheet are the same. Let $I = \{i_j | 0 \leq j \leq 7\}$ be the set of locations of cube variables in $A[2][i]$ $(i \in \{0, 2, 3\})$. Let I_i be the set of key bits which would multiply with the cube variables in $A[2][i]$ $(i \in \{0, 2, 3\})$ in the first round. According to the proof of Theorem 1, $I_i = \{i_j + m_i\}$ $(i \in \{0, 2, 3\})$, where m_i $(i \in \{0, 2, 3\})$ is determined by θ and the rotation constants used in ρ. In this case, we could obtain that $m_0 = 60,\ m_2 = 63$ and $m_3 = 5$, see Figure 6. Since the number of effective key bits is eight, we have $I_0 = I_2 = I_3$. According to Lemma 1, $|I|$ could be divisible by $\frac{64}{gcd(64,(m_i - m_j))}$ $(i \neq j, i, j \in \{0, 2, 3\})$. However, $\frac{64}{gcd(64,(m_0 - m_2))} = 64$, that is, we get a contradiction. Hence, for a 32-variable cube whose cube sums do not depend on the value of $A[1][0]$ after six rounds, the minimum number of its effective key bits would be at least nine. □

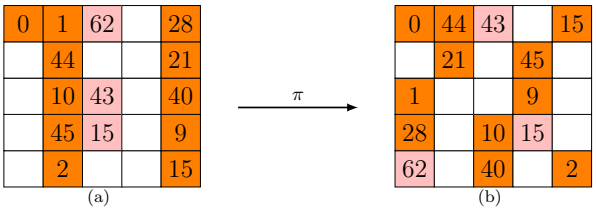

Fig. 6. *The state after ρ and π*

In the case of the 7-round Keccak-MAC, we obtain the similar conclusion.

Theorem 6. In the case of recovering $A[0][0]$ (resp.$A[1][0]$) of the 7-round Keccak-MAC, for a 64-variable cube whose cube sums do not depend on the value of $A[1][0]$ (resp. $A[0][0]$) after seven rounds, the number of its effective key bits is at least seventeen.

According to Theorem 5 and 6, we could obtain the lower bounds of the complexities of the divide-and-conquer attack on the round-reduced Keccak-MAC, see Table 4.

Table 4 The lower bounds of the complexities

Rounds	TC in PP	MC in PP	TC in OP
6	2^{41}	2^{12}	2^{40}
7	2^{76}	2^{20}	2^{75}

4.2. Scenario II: cubes with no restriction and not considering specific rotation constants used in ρ

Since there is no proof that an optimal cube for the divide-and-conquer attack should satisfy such restriction in the above subsection. In this subsection, we shall consider the lower bounds of complexities of divide-and-conquer attacks based on cubes without any restriction, and we even neglect the specific rotation constants used in ρ. That is to say, we could choose cube variables from nineteen lanes ($c = 256$ and the 128-bit key is placed in $A[0][0]$ and $A[1][0]$). Since we need to the keep the parities of columns containing cube variables constant, we could only choose cube variables from at most fourteen lanes, see Figure 7. In Figure 7, the squares with "cp" represent lanes used to keep the column parities constant, the orange squares mean lanes containing key bits, the pink squares represent lanes padded zeros (the capacity part) and the blank squares means lanes we could choose cube variables from.

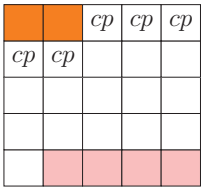

Fig. 7. *lanes which could be chosen cube variables from*

In the case of the 6-round Keccak-MAC, we need 32-variable cubes to complete the divide-and-

conquer attack. We could choose cube variables from lanes in five sheets at most. For example, if we choose cube variables from lanes in the same sheet, there are at most three lanes could be used (one lane is used to keep the column parities constant), then there would be at least one lane containing $\lceil 32/3 \rceil = 11$ cube variable. Hence, there would be eleven bits of the fourth lane containing cube variables for keeping the column parities constant. In other words, there would be $32 + 11 = 43$ state bits containing cube variables. We summarize the result of each case in Table 5.

Table 5 Summary of each case

Number of sheets where cube variables chosen	Maximum number of cube variables in each lane after π	Number of state bits containing cube variables after π
1	11	43
2	6	43
3	4	43
4	3	43
5	3	44

As we can see in Table 5, there would be at least forty-three state bits containing cube variables after π. In other words, there would be at least forty-three key bits multiplying with cube variables (some of these forty-three key bits may be the same). Due to θ, one key bit would diffuse to eleven state bits after θ in the first round. For a 32-variable cube (parities of columns containing cube variables are kept constant), there would be at least $\lceil 43/11 \rceil = 4$ effective key bits.

In fact, after θ, there are only seven lanes which are next to only one lane containing key bits. What is more, these seven lanes are in two sheets, see the pink squares in Figure 8. When we choose cube variables from lanes in two sheets, there would be at least one lane containing six cube variables. Since the key bits in the same lane are pairwise different, there would be at least six effective key bits. When we choose cube variables from at least three sheets, it is possible that the number of effective key bits is less than 6. Since cube variables come from at least three different sheets, there exist some cube variables which do not belongs to the above seven lanes. Hence, there exist at least two cube variables multiplying with two key bits. Namely, in this case, there would be at least $\lceil (41 + 2 \cdot 2)/11 \rceil = 5$ effective key bits.

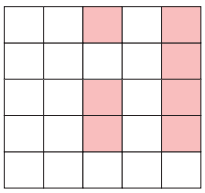

Fig. 8. *Lanes which would be next to only one lane containing key bits after π*

Similarly, in the case of the 7-round Keccak-MAC, for a 64-variable cube, we obtian that the number of the effective cube variables would be at least ten effective key bits.

Even if the technique of auxiliary variables could be used, the lower bounds are still not better than the complexities of the conditional cube tester in [12]. We summarize the lower bounds of complexities of divide-and-conquer attacks on the 6-round and the 7-round Keccak-MAC in Table 6.

Table 6 Lower bounds of the complexities based on cubes without any restriction

Rounds	TC in PP	MC in PP	TC in OP
6	2^{40}	2^{8}	2^{39}
7	2^{73}	2^{9}	2^{73}

5. Conclusion

In this paper we study divide-and-conquer attack against Keccak-MAC. Improved divide-and-conquer attacks against 6- and 7-round Keccak-MAC are given by exploiting the property of ρ which is not paid enough attention by the previous attackers. Furthermore, we theoretically analyse the lower bounds of complexities of divide-and-conquer attacks on the round-reduced Keccak-MAC. It is shown that the Keccak-MAC can resist the divide-and-conquer attack better than the conditional cube tester no matter which rotation constants are used in the ρ mapping.

6. References

[1] Bertoni Guido, Daemen Joan, Peeters Michaël, and Gilles Van Assche. Keccak sponge function family main document. http://Keccak.noekeon.org/Keccak-main-2.1.pdf.

[2] Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Preimage attacks on the round-reduced keccak with the aid of differential cryptanalysis. *IACR Cryptology ePrint Archive*, 2013:561, 2013.

[3] Daniel J. Bernstein. Second preimages for 6 (7 (8??)) rounds of keccak. NIST mailing list (2010).

[4] María Naya-Plasencia, Andrea Röck, and Willi Meier. Practical analysis of reduced-round keccak. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *Progress in Cryptology - INDOCRYPT 2011 - 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011. Proceedings*, volume 7107 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2011.

[5] Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced keccak. *J. Cryptology*, 27(2):183–209, 2014.

[6] Sourav Das and Willi Meier. Differential biases in reduced-round keccak. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT*, volume 8469 of *Lecture Notes in Computer Science*, pages 69–87. Springer, 2014.

[7] Christina Boura and Anne Canteaut. Zero-sum distinguishers for iterated permutations and application to keccak-f and hamsi-256, 2010.

[8] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of keccak and *Luffa*. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 2011.

[9] Ming Duan and Xuejia Lai. Improved zero-sum distinguisher for full round keccak-f permutation. *IACR Cryptology ePrint Archive*, 2011:23, 2011.

[10] Bertoni Guido, Daemen Joan, Peeters Michaël, and Gilles Van Assche. Keyak. http://Keyak.noekeon.org.

[11] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced keccak sponge function. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 733–761. Springer, 2015.

[12] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional cube attack on reduced-round keccak sponge function. *IACR Cryptology ePrint Archive*, 2016:790, 2016.

[13] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[14] Gregory V. Bard, Nicolas Courtois, Jorge Nakahara Jr., Pouyan Sepehrdad, and Bingsheng Zhang. Algebraic, aida/cube and side channel analysis of KATAN family of block ciphers. In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, volume 6498 of *Lecture Notes in Computer Science*, pages 176–196. Springer, 2010.