Practical Key-recovery Attacks on Round-Reduced Ketje Jr, Xoodoo-AE and Xoodyak

Haibo Zhou[1], Zheng Li[1], Xiaoyang Dong[2⋆], Keting Jia[3⋆], and Willi Meier[4]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, School of Mathematics, Shandong University, Jinan 250100, China
{zhouhaibo, lizhengcn}@mail.sdu.edu.cn
[2] Institute for Advanced Study, Tsinghua University, P. R. China
[3] Department of Computer Science and Technology, Tsinghua University, P. R. China
{xiaoyangdong, ktjia}@tsinghua.edu.cn
[4] FHNW, Switzerland
willi.meier@fhnw.ch

Abstract. Conditional cube attack was proposed by Huang et al. at EUROCRYPT 2017 to attack Keccak keyed mode. Inspired by dynamic cube attack, they reduce the degree by appending key bit conditions on the initial value (IV). Recently, Li et al. proposed new conditional cube attacks on Keccak keyed mode with extremely small degrees of freedom. In this paper, we find a new property on Li et al.'s method, and modify the new conditional cube attack for lightweight encryption algorithms using a 8-2-2 pattern, and apply it to 5-round Ketje Jr, 6-round Xoodoo-AE and Xoodyak, where Ketje Jr is among the 3rd round CAESAR competition candidates and Xoodyak is a Round 1 submission of the ongoing NIST lightweight cryptography project. Then we give the updated conditional cube attack analysis. All our results are of practical time complexity with negligible memory cost and our test codes are given in this paper. Notably, it is the first third-party cryptanalysis result for Xoodyak.

Keywords: Conditional Cube Attack, Keccak, Ketje Jr, Xoodoo, Xoodyak

1 Introduction

Authenticated encryption (AE) can provide confidentiality, integrity and authenticity for messages simultaneously. CAESAR competition [Com14], launched in 2014, aimed to find AE schemes with Security, Applicability, and Robustness. Totally, 57 candidates have been submitted to CAESAR in the first round competition. After three rounds of competition, only 6 authenticated encryption algorithms survived in the final portfolio.

⋆ Corresponding authors

Recently, NIST has initiated a process to solicit, evaluate, and standardize lightweight authenticated encryption algorithms with associated data (AEAD) and hashing, that are suitable for use in constrained environments where the performance of current NIST cryptographic standards is not acceptable. Till April 18, 2019, NIST has received 57 submissions, out of which 56 were selected as Round 1 Candidates.

Ketje [BDP+16] is one of the third round candidates of CAESAR competition [Com14], whose lightweight version is Ketje Jr, which leverages Keccak-p permutation with 200-bit internal state. Using a cube-attack-like method [DMP+15], Dong et al. [DLWQ17] give the first third party cryptanalysis on Ketje Jr with initialization phase reduced to 5 rounds. Then, Bi et al. [BDL+18] and Song et al. [SG18] improved the cube-attack-like cryptanalysis by MILP method, and obtained better attacks [SG18] on 5-round Ketje Jr. Song et al. [SG18] also give 6-round attacks when the recommended 96-bit key is reduced to 72 bits for Version 1 and 80 bits for Version 2. When targeting the encryption phase of Ketje Jr, Fuhr et al. [FNR18] described key recovery attacks on Ketje Jr with a rate extended to 32 or 40 bits (instead of the nominal 16 bits).

Xoodoo is another permutation proposed by Daemen et al. [DHAK18] at ToSC 2018. Song et al. [SG18] gave the first key-recovery attack on 6-round Xoodoo-AE, which is an artificial AE by using Xoodoo in Ketje style. In addition, the official AEAD scheme Xoodyak [DHAK18] based on Xoodoo is included in the Round 1 candidates of NIST lightweight cryptography competition. Besides the nonce-respecting security claim, the authors [DHAK18] also clarify the nonce-misuse case that: "*Nonce violation and release of unverified decrypted ciphertext have no consequences for integrity and do not put the key in danger for* Xoodyak".

Our Contribution. In this paper, we investigate three lightweight AEAD schemes, namely Ketje Jr (v1 and v2), Xoodoo-AE and Xoodyak, with Li et al.'s [LDB+19] new conditional cube attack. Conditional cube attack was first introduced by Huang et al. [HWX+17] at EUROCRYPT 2017 to attack Keccak keyed modes. However, Huang et al.'s attack becomes invalid when applied to targets with very small degrees of freedom even with the help of MILP models [LDW17, SGSL18]. Ketje Jr (v1 and v2), Xoodoo-AE and Xoodyak are exactly such schemes. In fact, the previous attacks [DLWQ17, SG18] on reduced Ketje Jr or Xoodoo-AE are mainly based on Dinur et al.'s cube-attack-like cryptanalysis [DMP+15].

Recently, Li et al. [LDB+19] proposed a new conditional cube attack, which could work even on targets with extremely small degrees of freedom. They introduced the so-called kernel quadratic term to replace the Huang et al.'s conditional cube variables, which makes sure that no cubic term appears in the output of 2nd round.

In this paper, we study Li et al.'s method and discover some new properties, such as using a 8-2-2 pattern for the lightweight algorithm can control the conditional cube variables' diffusion better; we also can adapt the 8-2-2 pattern to the 7-2-2 pattern for some algorithms whose state has short columns. Then we apply it and get improved key-recovery attacks on reduced Ketje Jr (v1 and v2), Xoodoo-AE, as well as the first attack on reduced Xoodyak. The advantages of our attacks are summarized in Table 1, and described as follows:

- For 5-round initialization of Ketje Jr v1 with recommended key length, we could recover the 96-bit key in $2^{26.6}$ time with negligible memory cost, while the best previous attack needs $2^{36.86}$ time and 2^{18} memory.
- For 5-round initialization of Ketje Jr v2 with recommended key length, we could recover the 96-bit key in $2^{27.5}$ time with negligible memory cost, while the best previous attack needs $2^{34.91}$ time and 2^{15} memory.
- For 6-round Xoodoo-AE, we could recover the 128-bit key in $2^{40.5}$ time with negligible memory cost, while the best previous attack needs 2^{89} time and 2^{55} memory.
- For 6-round Xoodyak in nonce-misuse settings, we could recover the 128-bit key in $2^{43.8}$ time with negligible memory cost.

Table 1: Summary of Key-recovery Attacks

Target	**b**	—K—	DF	Rounds	T	M	Source	Type
Ketje Jr v1	200	96	86	5/13	2^{56}	2^{38}	[DLWQ17]	T1
					$2^{36.86}$	2^{18}	[SG18]	T1
					$2^{26.6}$	–	Sect. 5.1	T2
Ketje Jr v2	200	96	86	5/13	$2^{50.32}$	2^{32}	[DLWQ17]	T1
					$2^{34.91}$	2^{15}	[SG18]	T1
					$2^{27.5}$	–	Sect. 5.1	T2
Xoodoo-AE	384	128	238	6/-	2^{89}	2^{55}	[SG18]	T1
					$2^{40.5}$	–	Sect. 5.2	T2
Xoodyak†	384	192	192	6/12	$2^{43.8}$	–	Sect. 5.3	T2

†: The attack on Xoodyak works in nonce-reuse setting.
T1: Cube-attack-like attack
T2: Conditional cube attack
DF: Degrees of freedom

2 Preliminaries

2.1 Notations

Some notations for Keccak variants are as follows:

S_0	the initial state of Keccak-p permutation,
$S_{i-1,\theta}$	the internal state after θ in i-th round of Keccak-p, $i \geq 1$,
$S_{i-1,\pi}$	the internal state after π in i-th round, $i \geq 1$,
S_i	the output state of the i-th round, $i \geq 1$,

Thus the internal states of i-th round Keccak are as follows:

$$S_{i-1} \xrightarrow{\theta} S_{i-1,\theta} \xrightarrow{\rho} S_{i-1,\rho} \xrightarrow{\pi} S_{i-1,\pi} \xrightarrow{\chi} S_{i-1,\chi} \rightarrow S_i. \tag{1}$$

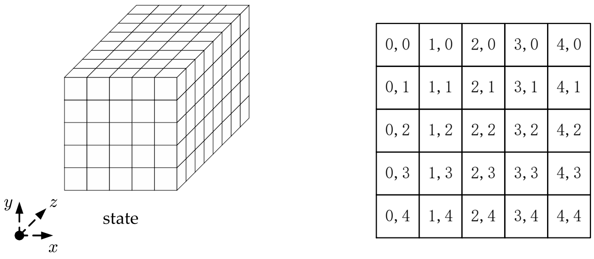

Fig. 1: (a) The Keccak State [BDPVA09], (b) State A In 2-dimension

Similarly, the internal states of i-th round Xoodoo are as follows:

$$S_{i-1} \xrightarrow{\theta} S_{i-1,\theta} \xrightarrow{\rho_{west}} S_{i-1,\rho_{west}} \xrightarrow{\iota} S_{i-1,\iota} \xrightarrow{\chi} S_{i-1,\chi} \xrightarrow{\rho_{east}} S_{i-1,\rho_{east}} \rightarrow S_i. \quad (2)$$

$(*, j, k)$	the index of row,
$(i, *, k)$	the index of column,
$(i, j, *)$	the index of lane,
(i, j, k)	the index of bit,
$A[i][j]$	the lane indexed by $(i, j, *)$ of state A,
$A[i][j][k]$	the bit indexed by (i, j, k) of state A.

2.2 The Keccak-p permutations

The Keccak-p permutations are derived from the Keccak-f permutations [BDPVA09] and have a tunable number of rounds. A Keccak-p permutation is defined by its width $b = 25 \times 2^l$, with $b \in \{25, 50, 100, 200, 400, 800, 1600\}$, and its number of rounds n_r, denoted as Keccak-$p[b, n_r]$. The round function $\mathtt{R}$ consists of five operations, denoted as $\mathtt{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$, and the details are as follows:

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

Keccak-$p[b, n_r]$ works on a state A of size b, which can be represented as 5×5 $\frac{b}{25}$-bit lanes, as depicted in Figure 1, $A[i][j]$ with i for the index of column and j for the index of row. In what follows, indexes i and j are in set $\{0, 1, 2, 3, 4\}$ and they are working modulo 5 without other specification.

2.3 Ketje

Ketje [BDP+16] is a submission by the Keccak team. It is a sponge-like construction. In Ketje v1, two instances are proposed, Ketje Sr and Jr with

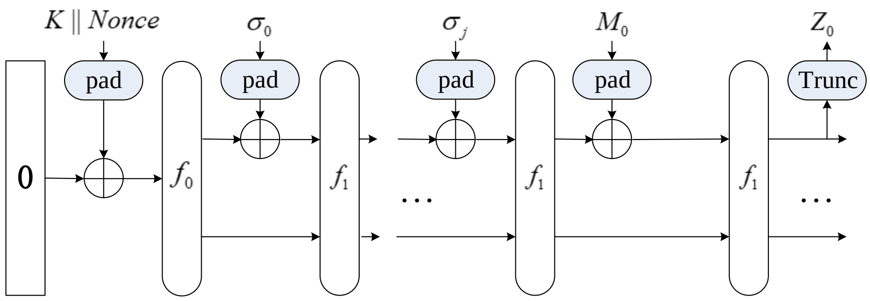

Fig. 2: KETJE, where the finalization is omitted.

400-bit and 200-bit state sizes, respectively. In the latest KETJE v2, another two instances KETJE MINOR and MAJOR are added to the family, with 800-bit and 1600-bit state sizes, respectively. KETJE SR is the primary recommendation. The four concrete instances of KETJE v2 are shown in Table 2. In the following, we give a brief overview about KETJE v2. For a complete description, we refer to the design document [BDP+16].

The structure of KETJE is an authenticated encryption mode MonkeyWrap, which is based on MonkeyDuplex [BDPA11]. It consists of four parts: initialization, processing associated data, processing the plaintext, finalization. Figure 2 illustrates the scheme of KETJE v2, where the finalization is omitted. In KETJE v2, *the twisted permutations*, KECCAK-$p^*[b]$=$\pi\circ$ KECCAK-$p[b] \circ \pi^{-1}$, are introduced to effectively re-order the bits in the state. $\pi^{-1} : A[x + 3y][x] = A[x][y]$ is the inverse of π, shown in Figure 3. Specially, f_0=KECCAK-$p^*[b, 12]$ and f_1=KECCAK-$p^*[b, 1]$.

Table 2: Four Instances in KETJE v2

Name	f	ρ	Main use case
KETJE JR	KECCAK-$p^*[200]$	16	lightweight
KETJE SR	KECCAK-$p^*[400]$	32	lightweight
KETJE MINOR	KECCAK-$p^*[800]$	128	lightweight
KETJE MAJOR	KECCAK-$p^*[1600]$	256	high performance

2.4 Xoodoo

At ToSC 2018, Daemen et al. [DHAK18] proposed a 384-bit permutation, called XOODOO, whose design is similar to KECCAK. The state is presented as a three-dimension matrix of bits $A[4][3][w]$, where $w = 32$. The round function of

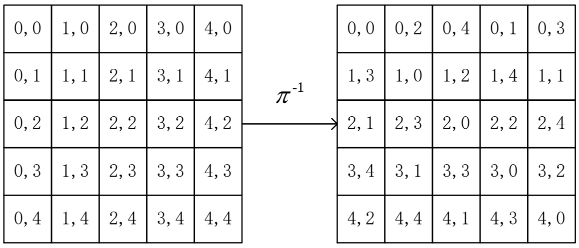

Fig. 3: π^{-1}

XOODOO has five operations, denoted as $\mathtt{R} = \rho_{east} \circ \chi \circ \iota \circ \rho_{west} \circ \theta$, and the details are as follows:

$$
\begin{aligned}
\theta &: A[x][y][z] = A[x][y][z] \oplus \textstyle\sum_{j=0}^{2} (A[x-1][j][z-5] \oplus A[x-1][j][z-14]). \\
\rho_{west} &: A[x][1][z] = A[x-1][1][z],\ A[x][2][z] = A[x][2][z-11]. \\
\iota &: A[0][0] = A[0][0] \oplus RC_i. \\
\chi &: A[x][y][z] = A[x][y][z] \oplus ((A[x][y+1][z] \oplus 1) \wedge A[x][y+2][z]). \\
\rho_{east} &: A[x][1][z] = A[x][1][z-1],\ A[x][2][z] = A[x-2][2][z-8].
\end{aligned}
$$

As pointed out in [DBH$^+$], XOODOO could be used as an AE scheme in KETJE style.

2.5 Xoodyak

As a Round 1 candidate of NIST lightweight cryptography competition, XOODYAK [DHAK18] includes an AEAD scheme and a hashing scheme. Both of them are based on XOODOO permutation.

XOODYAK-AEAD uses a new mode of operation, called *Cyclist*. As shown in Figure 4, f is the 12-round permutation XOODOO[12]. XOODYAK-AEAD first absorbs a 128-bit key into the 384 state, then applies XOODOO[12] to the state; second, absorbs a 128-bit nonce; then, the 192-bit associated data blocks; then absorb 192-bit plaintext blocks and output ciphertext blocks; then go to the finalization phase. Please refer to [DHAK18] for detailed information.

Besides the nonce-respecting security claim, the authors [DHAK18] also clarify the nonce-misuse case that: "*Nonce violation and release of unverified decrypted ciphertext have no consequences for integrity and do not put the key in danger for* XOODYAK". In this paper, we assume the nonce is reused and we try to recover the key of reduced XOODYAK. We apply our conditional cube attack in phase of absorbing M_0, and select cube variables in M_0 with 192-bit degrees of freedom. Our target is to recover the other 192-bit unknown state. Once we recover it, the full 384-bit state is known and we could compute the key inversely.

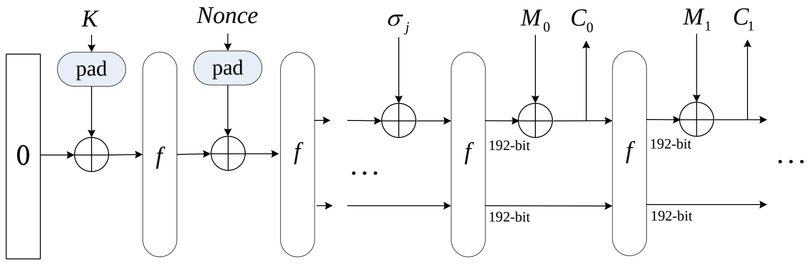

Fig. 4: Framework of Xoodyak-AEAD, where the finalization is omitted.

3 Related Works

3.1 Cube Attack

The cube attack [DS09] was introduced by Dinur and Shamir at EUROCRYPT 2009. It assumes that the output bit of a symmetric cryptographic scheme can be regarded as a polynomial over $GF(2)$.

Theorem 1. *([DS09]) Let $f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1})$ be a polynomial over $GF(2)$, where $k_0, ..., k_{n-1}$ are secret variables, and $v_0, ..., v_{m-1}$ are public variables.*

For a set $I = \{i_1, i_2, ..., i_{|I|}\} \subset \{0, ..., m-1\}$, $f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1})$ can be represented uniquely as

$$f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}) = T_I \cdot P + Q(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}), \tag{3}$$

where $T_I = v_{i_1} \cdots v_{i_{|I|}}$. The polynomial P only relates to v_s's ($s \notin I$) and the secret variables, and $Q(k_0, ..., k_{n-1}, v_0, ..., v_{m-1})$ misses at least one variable in T_I. T_I is called maxterm and P is called superpoly.

Denote by C_I the structure, called cube, consisting of all $2^{|I|}$ different vectors with $v_i, i \in I$ being active (traversing all 0-1 combinations) and non-cube indices $v_s, s \notin I$ being static constants.

Then the sum of f over all values of the cube C_I (cube sum) is

$$\sum_{v_{i_1}, ..., v_{i_{|I|}} \in C_I} f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}) = P. \tag{4}$$

The basic idea is to find enough T_I's P is linear and not a constant. This enables the key recovery through solving a system of linear equations.

3.2 Conditional Cube Attack

Conditional cube attack [HWX+17] was proposed by Huang et al. at EUROCRYPT 2017 to attack Keccak keyed mode. Then it soon was applied to many cryptanalysis, such as [LDW17, LBDW17, SG18, BLD+18]. Inspired by dynamic cube attack [DS09], which reduces the degree of output polynomials of cube variables by adding some bit conditions on the initial value (IV), they reduce the degree by appending key bit conditions. The techniques are similar to message modification technique [WY05, WYY05] and conditional differential cryptanalysis [KMN10] which used bit conditions to control differential propagation.

Definition 1. *([HWX+17]) Cube variables that have propagation controlled in the first round and are not multiplied with each other after the second round of* Keccak *are called* ***conditional cube variables****. Cube variables that are not multiplied with each other after the first round and are not multiplied with any conditional cube variable after the second round are called* ***ordinary cube variables****.*

Theorem 2. *([HWX+17]) For* $(n+2)$*-round* Keccak *sponge function* $(n > 0)$*, if there are p* $(0 \leq p < 2^n + 1)$ *conditional cube variables* $v_0, ..., v_{p-1}$*, and* $q = 2^{n+1} - 2p + 1$ *ordinary cube variables,* $u_0, ..., u_{q-1}$ *(If* $q = 0$*, we set* $p = 2^n + 1$*), the term* $v_0 v_1 ... v_{p-1} u_0 ... u_{q-1}$ *will not appear in the output polynomials of* $(n+2)$*-round* Keccak *sponge function.*

Actually, in the previous conditional cube attacks [HWX+17, LBDW17, SGSL18], they only use the special case of the above theorem when $p = 1$. We describe it as a corollary for clearness.

Corollary 1. *For* $(n+2)$*-round* Keccak *sponge function* $(n > 0)$*, if there is one conditional cube variable* v_0*, and* $q = 2^{n+1} - 1$ *ordinary cube variables,* $u_0, ..., u_{q-1}$*, the term* $v_0 u_0 ... u_{q-1}$ *will not appear in the output polynomials of* $(n+2)$*-round* Keccak *sponge function.*

3.3 New Conditional Cube Attack

Recently, Li et al. [LDB+19] proposed new conditional cube attacks on Keccak keyed mode with extremely small degrees of freedom. In Huang et al.'s attack [HWX+17], all cube variables must not be multiplied together in the first round. However, Li et al. [LDB+19] relaxed this constraint by introducing the so-called *kernel quadratic term*.

Definition 2. *([LDB+19]) Suppose all the* $(q+2)$ *cube variables are* $v_0, v_1, u_0, ...u_{q-1}$*, and constraints are as follows:*

- *After 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****.*

Then the following corollary is deduced.

Corollary 2. *[LDB+19] For $(n+2)$-round* Keccak *sponge function $(n>0)$, if there is one kernel quadratic term v_0v_1, and $q = 2^{n+1}-1$ ordinary cube variables, $u_0, u_1, ..., u_{q-1}$, the term $v_0v_1u_0u_1...u_{q-1}$ will not appear in the output polynomials of $(n+2)$-round* Keccak *sponge function under certain bit conditions.*

So the distinguisher is approached as follows:

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

According to Definition 2, in Li et al.'s [LDB+19] new conditional cube attack, we only care that the *kernel quadratic term* (i.e. v_0v_1) should not be multiplied with ordinary cube variables in the second round. So they devised the so-called 6-2-2 pattern to reduce the diffusion of v_0v_1. Therefore, they could get more degrees of freedom to find ordinary cube variables.

Concretely, in the initial state S_0, Li et al. selected v_0 and v_1, such that $S_0[2][0][0] = S_0[2][1][0] = S_0[3][0][34] = S_0[3][1][34] = v_0$, are in CP-kernel, $S_0[0][1][60] = S_0[1][1][1] = v_1$ are not in CP-kernel. Then, before the first χ, the distribution of v_0 and v_1 is shown in Figure 5. Moreover, after χ, *kernel quadratic term* v_0v_1 is also in CP-kernel. Thus, the second θ becomes the identity for v_0v_1. Therefore, before the second χ, only two bit positions are related to v_0v_1. So, it is called 6-2-2 pattern. Li et al. also gave a method to derive the 6-2-2 pattern. For more details, please refer to [LDB+19].

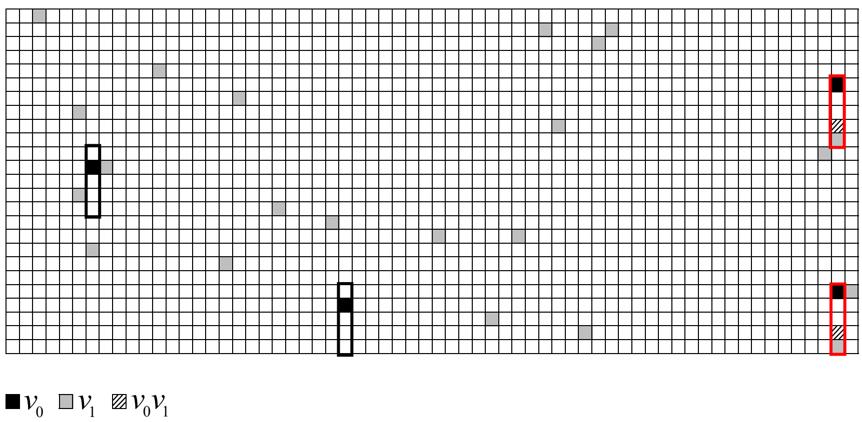

Fig. 5: 6-2-2 Pattern: Generation of Kernel Quadratic Terms in the First χ [LDB+19].

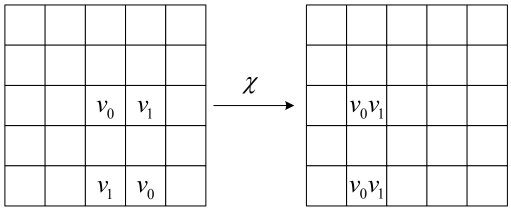

Fig. 6: Slice $(*, *, z)$: from $S_{0,\pi}$ to S_1

3.4 MILP Model of Conditional Cube Attack

At ASIACRYPT 2017, Li et al. [LBDW17] for the first time applied MILP method to cube attacks on keyed Keccak. Later, the MILP model was improved by [SGSL18], and also applied to cube-attack-like method by Bi et al. [BDL+18] and Song et al. [SG18]. In the previous MILP model of conditional cube attack, no cube variables multiply with each other in the first round, and the conditional cube variable of degree one is considered not to multiply with any ordinary cube variables in the second round. To limit the diffusion of conditional cube variables, conditions are added in the first round. Actually, as described in Corollary 1, $(2^{n+1}-1)$ ordinary cube variables are needed to perform a $(n+2)$-round attack. To obtain enough ordinary cube variables for the attack, the objective of Li et al.'s MILP model [LBDW17] at ASIACRYPT 2017 was to maximize the number of ordinary cube variables. To reduce the attack complexity further, Song et al. [SGSL18] proposed a new MILP model to minimize the number of conditions at ASIACRYPT 2018.

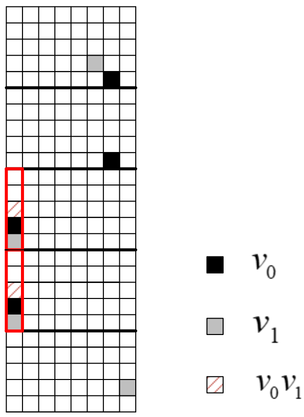

Fig. 7: 8-2-2 Pattern: Generation of Kernel Quadratic Terms in the First χ.

4 8-2-2 Pattern

Based on Li et al.'s [LDB+19] new discovery, we introduce the 8-2-2 pattern instead of the 6-2-2 pattern to attack targets with fewer degrees of freedom, namely Ketje Jr, Xoodoo-AE and Xoodyak-AEAD.

According to the new conditional cube attack given in Sect. 3.3, we only care that the *kernel quadratic term* (i.e. v_0v_1) should not be multiplied with ordinary cube variables in the second round. Li et al.'s 6-2-2 pattern allows v_1 to be not in the CP-kernel. Thus, it will distribute to 22 bit positions before the first χ as shown in Figure 5. This reduces the degrees of freedom to select ordinary cube variables u_i, since according to Definition 2, u_i has to be prevent from multiplying with v_i in the first round.

Hence, we introduce the so-called 8-2-2 pattern, i.e. both v_0 and v_1 are set in CP-kernel in the initial state. For example, in the initial state S_0 of Ketje Jr, v_0 occupies 4 bits of S_0, i.e. $A[4][3][0]$=$A[4][4][0]$=$A[2][3][1]$=$A[2][4][1]$=v_0, and v_1 also occupies 4 bits, i.e. $A[0][4][6]$=$A[0][3][6]$=$A[3][4][0]$=$A[3][3][0]$=v_1. Before the first χ operation, v_0 only appears in the 4 bits, while v_1 also appears in 4 bits. Similar to Li et al.'s work [LDB+19], only 2 S-boxes are related to v_0 and v_1 simultaneously and the quadratic term v_0v_1 is generated in two bit positions after χ operation, as shown in Figure 7. Moreover, the two bits containing v_0v_1 in S_1 are also in CP-kernel. Similar to Li et al. [LDB+19], we also give the method to find the 8-2-2 pattern.

Table 3: Related Indexes of Bits Containing v_0, v_1 and v_0v_1

Index	v_0v_1	
S_1	(x, y_0, z) (x, y_1, z)	
Index	v_0	v_1
$S_{0,\pi}$	$(x+1, y_0, z)$ $(x+1, y_1, z)$	$(x+2, y_0, z)$ $(x+2, y_1, z)$
$S_{0,\rho}$	$(x+3y_0+1, x+1, z)$ $(x+3y_1+1, x+1, z)$	$(x+3y_0+2, x+2, z)$ $(x+3y_1+2, x+2, z)$
$S_{0,\theta}$	$(x+3y_0+1, x+1, z-\rho[x+3y_0+1, x+1])$ $(x+3y_1+1, x+1, z-\rho[x+3y_1+1, x+1])$	$(x+3y_0+2, x+2, z-\rho[x+3y_0+2, x+2])$ $(x+3y_1+2, x+2, z-\rho[x+3y_1+2, x+2])$
S_0	$(x+3y_0+1, x+1, z-\rho[x+3y_0+1, x+1])$ $\boldsymbol{(x+3y_0+1, x_1, z-\rho[x+3y_0+1, x+1])}$ $(x+3y_1+1, x+1, z-\rho[x+3y_1+1, x+1])$ $\boldsymbol{(x+3y_1+1, x_2, z-\rho[x+3y_1+1, x+1])}$	$(x+3y_0+2, x+2, z-\rho[x+3y_0+2, x+2])$ $\boldsymbol{(x+3y_0+2, x_3, z-\rho[x+3y_0+2, x+2])}$ $(x+3y_1+2, x+2, z-\rho[x+3y_1+2, x+2])$ $\boldsymbol{(x+3y_1+2, x_4, z-\rho[x+3y_1+2, x+2])}$

The method to find 8-2-2 pattern. The most important constraint is that v_0v_1 has to be in a CP-kernel in S_1. So we start with setting v_0v_1 in a CP-kernel. Denote the two bits containing v_0v_1 as (x, y_0, z), (x, y_1, z). According to the expression of χ, v_0v_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_0v_1 in $S_1[x][y_1][z]$. So there will be 4 cases to determine the bit positions for v_0 and v_1 in reverse. For example, in Figure 6, v_0v_1 appears in $S_1[1][2][z]$ by multiplication of v_0 in $S_{0,\pi}[2][2][z]$ and v_1 in $S_{0,\pi}[3][2][z]$, and similarly v_0v_1 appears in $S_1[1][4][z]$ by multiplication of v_1 in $S_{0,\pi}[2][4][z]$ and v_0 in $S_{0,\pi}[3][4][z]$.

Under one of the 4 cases, Table 3 describes the bit positions of v_0v_1, v_0 and v_1 inversely from S_1 to S_0, while the other cases are similar. In Table 3, in order to reduce the diffusion of v_0 and v_1, the 4 bit positions containing v_0 and v_1 in S_0 are set in CP-kernel, where $x_1, x_2 \neq x+1$, $x_3, x_4 \neq x+2$. At last, all the 8 bits in S_0 should be selected in free space for ordinary cube variables. Accordingly, we can determine 8-2-2 patterns.

5 Applications to Ketje Jr, Xoodoo-AE and Xoodyak-AEAD

5.1 5-Round Attack against Ketje Jr

For Ketje Jr with 200-bit state, the recommended key size is 96 bits, while the shortest padding occupies 18 bits. A 5-round attack can be performed with 17 cube variables.

According to the new conditional cube attack illustrated in Sect. 3.3, we find a 8-2-2 pattern at first, and then search for the minimal number of key bit conditions and the 15 ordinary cube variables satisfying the corresponding rules by previous MILP models [LBDW17, SGSL18, LDB+19].

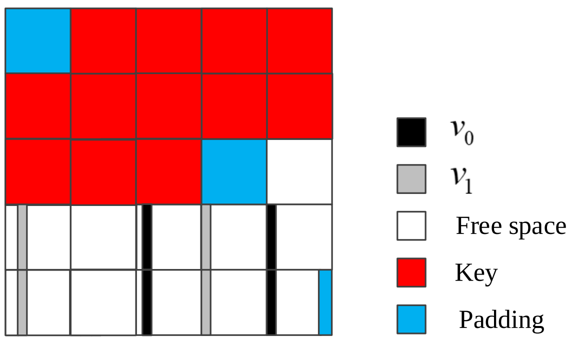

Fig. 8: The Initial State of Ketje Jr v1

5-round Attack on Ketje Jr *v1* In the procedure of attack on 5-round Ketje Jr v1, we select v_0 and v_1 following the 8-2-2 pattern. As shown in Figure 8, the 96-bit key is located at the red parts, while the padding part is shown in blue. v_0 is set in CP-kernel filled with black. i.e. $A[4][3][0]=A[4][4][0]=A[2][3][1]=A[2][4][1]=v_0$. And v_1 is located at 4 grey bits, i.e. $A[0][4][6]=A[0][3][6]=A[3][4][0]=A[3][3][0]=v_1$. The white bits represent free space to be selected as ordinary cube variables.

The cube variables and bit conditions to attack 5-round initialization phase of Ketje Jr v1 are given in Table 4.

Table 4: Parameters set for attack on 5-round Ketje JR v1

kernel quadratic term
$A[4][3][0]=A[4][4][0]=A[2][3][1]=A[2][4][1]=v_0$, $A[0][4][6]=A[0][3][6]=A[3][4][0]=A[3][3][0]=v_1$
Bit Conditions
$A[3][3][6] = k_7 + k_{22} + k_{47} + k_{62} + k_{87} + n_{71} + n_{78}$ $A[0][3][2] = k_9 + k_{34} + k_{49} + k_{74} + k_{89} + n_{18} + n_{50} + 1$ $A[3][4][4] = k_{12} + k_{27} + k_{52} + k_{67} + k_{92}$ $A[1][3][1] = k_1 + k_{26} + k_{41} + k_{66} + k_{81} + n_{50} + n_{57}$ $A[1][3][3] = k_3 + k_{28} + k_{43} + k_{68} + k_{83} + n_{52} + n_{59}$ $A[4][2][5] = k_{14} + k_{29} + k_{54} + k_{62} + k_{69} + k_{94} + n_{30} + n_{45} + n_{70} + n_{85} + 1$
Ordinary Cube Variables
$A[4][2][4]=u_0, A[4][3][4]=u_1, A[4][4][4]=u_0 + u_1, A[1][3][0]=u_2$, $A[1][4][0]=u_2, A[1][3][6]=u_3, A[1][4][6]=u_3, A[1][3][7]=u_4, A[1][4][7]=u_4$, $A[2][3][2]=u_5, A[2][4][2]=u_5, A[2][3][3]=u_6, A[2][4][3]=u_6, A[2][3][4]=u_7$, $A[2][4][4]=u_7, A[2][3][5]=u_8, A[2][4][5]=u_8, A[4][2][1]=u_9, A[4][3][1]=u_9$, $A[4][2][2]=u_{10}, A[4][3][2]=u_{11}, A[4][4][2]=u_{10} + u_{11}, A[4][2][3]=u_{12}$, $A[4][3][3]=u_{13}, A[4][4][3]=u_{12} + u_{13}, A[4][2][6]=u_{14}, A[4][3][6]=u_{14}$

Note that the number of the key bits to be guessed and assigned is 6. If the key guessing is right, it is expected to output zero cube sums. The time complexity to recover the 6-bit key is $2^6 \times 2^{17}$. According to the property of the permutation, it is totally symmetric in z-axis. Thus we can obtain corresponding parameter sets with any i-bit rotation ($0 \leq i < 8$) in z-axis. Therefore, the related key bits rotated by i bits can be recovered. Besides, according to Sect. 4, we can use other kernel quadratic terms not only different in z-axis to recover more key bits.

Totally, 12 iterations could recover the 72-bit key. Then guess 96-72=24 bits to determine the full key. The time complexity is $12 \times 2^6 \times 2^{17} = 2^{26.58}$.

To support our theory, we have run more than 1000 experiments and obtained correct key recovery of 5-round Ketje Jr v1 with 100 percent success rate. For saving space,we give an example here for intuition, in which the key is generated randomly and all the controllable nonce bits are set to zero. The program is run in Visual Studio 2012 with x64 platform Release. The time is less than 8 seconds for recovery of 6 key bits using one CPU core (Intel i7 3.6GHz), and parallelism can reduce time. Using the test code that we provided, one can verify it easily.

We calculate the cube sum at lane $(0,0),(1,0)$. As the probability, for which the cube sum on these two lanes is zero, is 2^{-16} for a random function. Therefore, if the 17-dimension cube sums of 5-round output is zero, we declare that the key guess is correct with high probability. Actually, we also calculate other lanes' cube sum, and all of them turn out to be zero. The test code is given in `https://github.com/alicebobb/aabb/tree/alicebobb-patch-1`.

96-bit key K:

`101000001101011001110100110111000111001000011101`

`110111001011011011111100100111010010110001010101`

The correct value for the guessed key bits in Table 4 is `001110`.

......

guessed value: `010110`, cube sums: `0x88, 0x52`

guessed value: `110110`, cube sums: `0xd5, 0x44`

guessed value: **001110**, cube sums: **0x0, 0x0**

guessed value: `101110`, cube sums: `0xef, 0xa5`

guessed value: `011110`, cube sums: `0x1b, 0x49`

......

To see the full key guess results, we put them in Appendix A.

5-round Attack on KETJE JR *v2* As Figure 9 shows us, after π^{-1} transformation, the 96-bit key is located at the red parts, while the padding part is shown in blue. And v_0 is set in CP-kernel as $A[4][2][0]=A[4][0][0]=A[0][1][3]=A[0][4][3]=v_0$ in black, and v_1 is located at 4 grey bits, i.e. $A[3][1][0]=A[3][4][0]=A[1][2][5]=A[1][4][5]=v_1$. The white bits represent free space to be selected as ordinary cube variables.

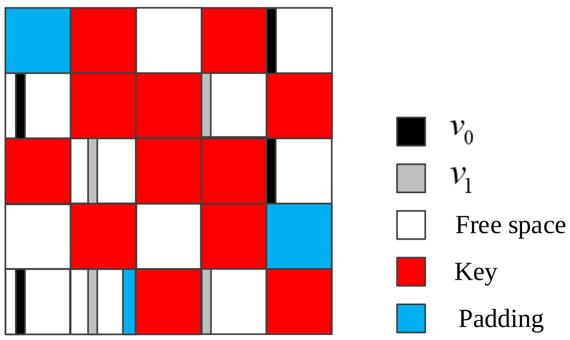

Fig. 9: The Initial State of KETJE JR v2

The cube variables and conditions to attack 5-round initialization phase of KETJE JR v2 are given in Table 5.

The number of bit conditions related to key is 7, hence, we have to guess a 7-bit key to assign conditions. The time complexity to recover the 7-bit key is

Table 5: Parameters set for attack on 5-round KETJE JR v2

kernel quadratic term
$A[4][2][0]{=}A[4][0][0]{=}A[0][1][3]{=}A[0][4][3]{=}v_0$, $A[3][1][0]{=}A[3][4][0]{=}A[1][2][5]{=}A[1][4][5]{=}v_1$
Bit Conditions
$A[1][2][4] = k_{11} + k_{52} + k_{67} + k_{83} + n_{35} + n_{51} + 1$
$A[1][2][6] = k_{13} + k_{54} + k_{69} + k_{85} + n_{37} + n_{53} + n_{78}$
$A[2][3][2] = k_{10} + k_{51} + k_{66} + k_{82} + n_{50} + n_{75} + n_{83}$
$A[0][4][5] = k_{22} + k_{38} + k_{53} + k_{94} + n_{14} + n_{46} + n_{62} + 1$
$A[4][2][7] = k_{23} + k_{39} + k_{54} + k_{95} + n_{78} + 1$
$A[4][0][4] = k_{13} + k_{28} + k_{44} + k_{69} + k_{85} + n_{37} + n_{45} + n_{53} + n_{68}$
$A[1][2][0] = k_0 + k_{23} + k_{39} + k_{56} + k_{72} + k_{95} + n_{32} + n_{80} + 1$
Ordinary Cube Variables
$A[3][1][4]{=}u_0, A[3][4][4]{=}u_0, A[3][1][7]{=}u_1, A[3][4][7]{=}u_1, A[0][1][0]{=}u_2$, $A[0][4][0]{=}u_2, A[0][1][1]{=}u_3, A[0][3][1]{=}u_4, A[0][4][1]{=}u_3 + u_4, A[0][1][2]{=}u_5$, $A[0][3][2]{=}u_6, A[0][4][2]{=}u_5 + u_6, A[0][1][4]{=}u_7, A[0][3][4]{=}u_8$, $A[0][4][4]{=}u_7 + u_8, A[0][1][5]{=}u_9, A[0][3][5]{=}u_9, A[0][1][6]{=}u_{10}$, $A[0][4][6]{=}u_{10}, A[2][0][4]{=}u_{11}, A[2][3][4]{=}u_{11}, A[2][0][6]{=}u_{12}, A[2][3][6]{=}u_{12}$, $A[2][0][7]{=}u_{13}, A[2][3][7]{=}u_{13}, A[3][1][2]{=}u_{14}, A[3][4][2]{=}u_{14}$

$2^7 \times 2^{17}$. According to the property of the permutation, it is totally symmetric in z-axis. Thus we can obtain corresponding parameter sets with any i-bit rotation ($0 \leq i < 8$) in z-axis. Therefore, the guessed key bits rotated by i bits can be recovered. 11 iterations of the above process could recover a 77-bit key and the remaining 19 key bits are recovered by exhaustive search. The total time complexity is $11 \times 2^7 \times 2^{17} = 2^{27.46}$.

Similarly, we have run more than 1000 experiments and obtained correct key recovery of 5-round KETJE JR v2 with 100 percent success rate. We also give an example here for intuition, in which the key is generated randomly and all the controllable nonce bits are set to zero. The program is run in Visual Studio 2012 with x64 platform Release. The time is less than 16 seconds for recovery of 7 key bits using one CPU core (Intel i7 3.6GHz), and parallelism can reduce time. Using the test code that we provided, one can verify it easily.

We calculate the cube sum at lane $(0, 0), (1, 1)$. The probability, for which the cube sum on these two lanes the cube sum is zero, is 2^{-16} for a random function. Therefore, if the 17-dimension cube sums of 5-round output is zero, we declare that the key guess is correct with high probability. Actually, we also calculate other lanes' cube sum, and all of them turn out to be zero. The test code is given in `https://github.com/alicebobb/aabb/tree/alicebobb-patch-1`.

96-bit key K:

`100101100000100110001010010101101010111011011001`

`110010011101101000111111101011011010011101111001`

The correct value for the guessed key bits in Table 5 is `0000110`.

......
guessed value: `0111010`, cube sums: `0xc, 0xca`
guessed value: `1111010`, cube sums: `0xa4, 0xe5`
guessed value: **0000110**, cube sums: **0x0, 0x0**
guessed value: `1000110`, cube sums: `0x2e, 0xea`
guessed value: `0100110`, cube sums: `0xc8, 0x7f`
......
To see the full key guess results, we put them in Appendix A.

5.2 6-Round Attack against Xoodoo-AE

As pointed out in [DBH+], Xoodoo [DHAK18] could be used as an AE scheme in Ketje style. We assume that the Xoodoo-AE has a 128-bit key and follows the Ketje's packing. As shown in Figure 10, the 128-bit key K is located at the 5 red lanes, and the padding parts are blue. The white part represents nonce bits. The operations θ and χ of Xoodoo are very similar to those of Keccak-p and ρ_{west} just reorders the state bits which is similar to ρ and π. So it is easy to modify the attack strategy of Keccak-p to Xoodoo.

However, Xoodoo's state is 3×4, not Keccak-p's 5×5 state, which means columns in Xoodoo are shorter than those in Keccak-p. Another different feature is the S-box, which is applied to every 3-bit column. Moreover, the first row and part of the second row are occupied by the key and padding bits in Xoodoo, which means, there are fewer degrees of freedom for us to search ordinary cube variables in CP-kernel. All those features will affect our decisions for choosing conditional cube variables and ordinary cube variables. Consequently, we modify the 8-2-2 pattern into a 7-2-2 pattern, which means that v_0 appears in 4 bits and v_1 appears in 3 bits in the initial state.

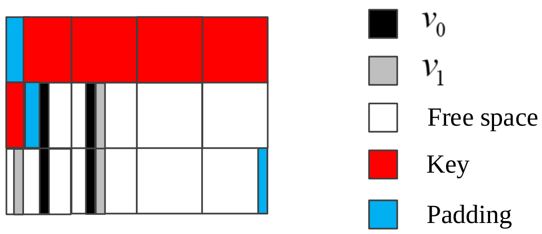

Fig. 10: The Initial State of Xoodoo-AE

Assume that v_0v_1 is the *kernel quadratic term*. As shown in Figure 10, v_0 is set in CP-kernel as $S_0[0][1][16] = S_0[0][2][16] = S_0[1][1][4] = S_0[1][2][4] = v_0$ in

black, and v_1 is located at 3 grey bits, i.e. $S_0[1][1][5] = S_0[1][2][5] = S_0[0][2][1] = v_1$. After operation θ and ρ_{west},
$S_{0,\rho_{west}}[1][1][16] = S_{0,\rho_{west}}[1][2][15] = v_0$
as well as $S_{0,\rho_{west}}[0][2][27] = S_{0,\rho_{west}}[2][1][4] = v_0$, and
$S_{0,\rho_{west}}[1][2][16] = S_{0,\rho_{west}}[1][0][15] = v_1$
as well as $S_{0,\rho_{west}}[0][2][12] = S_{0,\rho_{west}}[1][2][17] = S_{0,\rho_{west}}[1][0][6] = S_{0,\rho_{west}}[2][1][6] = S_{0,\rho_{west}}[1][2][26] = S_{0,\rho_{west}}[1][0][15] = S_{0,\rho_{west}}[2][1][5] = v_1$.

Table 6: Parameters set for attack on 6-round Xoodoo-AE

kernel quadratic term
$A[0][1][16]$=$A[0][2][16]$=$A[1][2][4]$=$A[1][1][4]$=v_0, $A[1][2][5]$=$A[1][1][5]$=$A[0][2][1]$=v_1
Bit Condition
$A[2][1][10]$=k_{66} + k_{75} + k_{112} + n_{67} + n_{186} + n_{195} + 1
Ordinary Cube Variables
$A[3][1][25]$=u_0,$A[3][2][25]$=u_0,$A[3][1][28]$=u_1,$A[3][2][28]$=u_1,$A[0][1][25]$=u_2, $A[0][2][25]$=u_2,$A[0][1][27]$=u_3,$A[0][2][27]$=u_3,$A[0][1][28]$=u_4,$A[0][2][28]$=u_4, $A[1][1][1]$=u_5,$A[1][2][1]$=u_5,$A[1][1][2]$=u_6,$A[1][2][2]$=u_6,$A[1][1][6]$=u_7, $A[1][2][6]$=u_7,$A[1][1][7]$=u_8,$A[1][1][8]$=u_9,$A[1][2][8]$=u_9,$A[1][1][9]$=u_{10}, $A[1][2][9]$=u_{10},$A[1][1][10]$=u_{11},$A[1][2][10]$=u_{11},$A[1][1][13]$=u_{12}, $A[1][2][13]$=u_{12},$A[1][1][18]$=u_{13},$A[1][2][18]$=u_{13},$A[1][1][26]$=u_{14},$A[1][2][26]$=u_{14}, $A[1][1][28]$=u_{15},$A[1][2][28]$=u_{15},$A[1][1][30]$=u_{16},$A[1][2][30]$=u_{16},$A[2][1][0]$=u_{17}, $A[2][2][0]$=u_{17},$A[2][2][3]$=u_{18},$A[2][1][6]$=u_{19},$A[2][2][6]$=u_{19},$A[2][1][12]$=u_{20}, $A[2][2][12]$=u_{20},$A[2][1][18]$=u_{21},$A[2][2][18]$=u_{21},$A[2][1][21]$=u_{22},$A[2][2][21]$=u_{22}, $A[2][1][24]$=u_{23},$A[2][2][24]$=u_{23},$A[3][1][2]$=u_{24},$A[3][2][2]$=u_{24},$A[3][1][3]$=u_{25}, $A[3][2][3]$=u_{25},$A[3][1][8]$=u_{26},$A[3][2][8]$=u_{26},$A[3][1][9]$=u_{27},$A[3][2][9]$=u_{27}, $A[3][1][11]$=u_{28},$A[3][2][11]$=u_{28},$A[3][1][17]$=u_{29},$A[3][2][17]$=u_{29},$A[3][1][19]$=u_{30}, $A[3][2][19]$=u_{30}

The cube variables and bit conditions are shown in Table 6. With 31 ordinary cube variables, there is only one bit condition related to key. We guess the key bit $k_{66} + k_{75} + k_{112}$ to assign the condition. The time complexity of one recovery is $2^1 \times 2^{33}$. According to the property of the permutation, it is totally symmetric in z-axis. Thus we can obtain corresponding parameter sets with any i-bit rotation ($0 \le i < 32$) in z-axis. Therefore, the guessed key bits rotated by i bits can be recovered. 90 iterations could recover a 90-bit key and the remaining key bits could be recovered by exhaustive search. Totally, it consumes $90 \times 2^1 \times 2^{33} + 2^{38} = 2^{40.5}$.

Similarly, we have run more than 1000 experiments and obtained correct key recovery of 6-round Xoodoo-AE with 100 percent success rate. we also give an example here for intuition, in which the key is generated randomly and all the controllable nonce bits are set to zero. The program is run in Visual Studio 2012 with x64 platform Release. The time is about 2 hours for recovery of one key bit using one CPU core (Intel i7 3.6GHz), and parallelism can reduce time. Using the test code that we provided, one can verify it easily.

We just list the cube sum at lane $(0,0),(1,0)$, and the probability that the cube sum for these two lanes cube sum is zero is 2^{-64} for a random function. Therefore, if the 33-dimension cube sums of 5-round output is zero, we declare that the key guess is correct with high probability. Actually, we also calculate other lanes' cube sum, and all of them turn out to be zero. The test code is given in `https://github.com/alicebobb/aabb/tree/alicebobb-patch-1`.

128-bit key K:
`1010000011010110011101001101110001110010000111011101110010110110`
`1111110010011101001011000101010100010111101000111100101100000101`
The correct value for the guessed key bit in Table 6 is `1`.
right key: `1`
guessed value: `0`, cube sums: `0x9dff359, 0xc614c263`
guessed value: `1`, cube sums: `0x0, 0x0`

5.3 6-Round Attack against Xoodyak

We attack reduced Xoodyak-AEAD in nonce-reuse setting. The targeted part of the Xoodyak-AEAD is the absorbing plaintext phase. As shown in Figure 4, we select cube variables from the 192-bit M_0, and we are going to recover the other 192-bit unknown state, which we denote as 192-bit equivalent key. We reduced the Xoodoo to 6 rounds, i.e., M_0 is processed by 6-round Xoodoo, then the corresponding ciphertexts outputs are used to compute the cube sums. As shown in Table 11, the 192-bit key K is located at the 6 red lanes. The white part are tweakable bits and could be selected as cube variables.

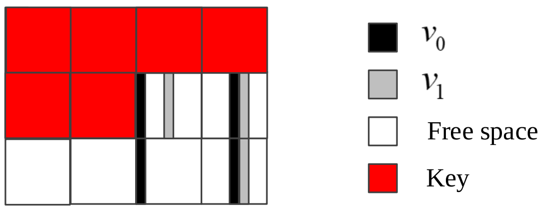

Fig. 11: The Initial State of Xoodyak

Similarly, assume that v_0v_1 is the *kernel quadratic term.* v_0 is set in CP-kernel as $S_0[2][1][0] = S_0[2][2][0] = S_0[3][1][20] = S_0[3][2][20] = v_0$ in black, and

v_1 is located at 3 grey bits, i.e. $S_0[3][1][21] = S_0[3][2][21] = S_0[2][1][17] = v_1$. After operation θ and ρ_{west}, $S_{0,\rho_{west}}[3][1][0] = S_{0,\rho_{west}}[3][2][31] = v_0$ as well as $S_{0,\rho_{west}}[2][2][11] = S_{0,\rho_{west}}[0][1][20] = v_0$, $S_{0,\rho_{west}}[3][2][0] = S_{0,\rho_{west}}[3][0][31] = v_1$ as well as $S_{0,\rho_{west}}[3][1][17] = S_{0,\rho_{west}}[0][1][22] = S_{0,\rho_{west}}[3][2][1] = S_{0,\rho_{west}}[3][0][22] = S_{0,\rho_{west}}[0][1][31] = S_{0,\rho_{west}}[3][2][10] = S_{0,\rho_{west}}[0][1][21] = v_1$. The white bits represent free space to be selected as ordinary cube variables.

To prevent ordinary cube variables to multiply with other cube variables at the first round, we try to select the ordinary cube variables in CP-kernel. With 31 ordinary cube variables, the number of bit conditions related to key is 6. Then a 6-round attack on Xoodyak can be performed. Both the cube variables and conditions are listed in Table 7. The 6 equivalent key bits to be guessed are as follows: $k_{66}+k_{89}$, $k_{75}+k_{89}$, $k_8+k_{31}+k_{99}+k_{113}$, $k_{33}+k_{42}+k_{79}+k_{161}+k_{170}$, $k_{49}+k_{58}+k_{95}+k_{177}+k_{186}$ and $k_{42}+k_{56}+k_{70}+k_{79}+k_{170}+k_{184}$. The time complexity to recover the 6-bit key is $2^6 \times 2^{33}$. Similar to the attack on Xoodoo-AE, it is totally symmetric in z-axis. Thus we can obtain corresponding parameter sets with any i-bit rotation ($0 \le i < 32$) in z-axis. We need 27 iterations of the above procedures to recover $27 \times 6 = 162$ bits key, and leave the other 30 bits key to exhaustive search. The total time complexity is $27 \times 2^6 \times 2^{33} + 2^{30} = 2^{43.8}$.

Table 7: Parameters set for attack on 6-round Xoodyak

kernel quadratic term
$A[2][1][0]=A[2][2][0]=A[3][2][20]=A[3][1][20]=v_0$, $A[3][2][21]=A[3][1][21]=A[2][1][17]=v_1$
Bit Conditions
$A[1][2][1] = k_{33} + k_{42} + k_{79} + k_{161} + k_{170} + n_{106}$ $A[1][2][24] = k_{42} + k_{56} + k_{70} + k_{79} + k_{170} + k_{184} + n_{106}$ $A[3][1][3] = k_8 + k_{31} + k_{99} + k_{113} + n_{163}$ $A[2][1][2] = k_{66} + k_{89} + n_{25} + n_{130} + n_{153} + n_{167}$ $A[3][2][16] = k_{75} + k_{89} + n_{25} + n_{153} + n_{167}$ $A[1][2][26] = k_{49} + k_{58} + k_{95} + k_{177} + k_{186} + n_{113}$
Ordinary Cube Variables
$A[3][1][26]=u_0$,$A[3][2][26]=u_0$,$A[3][1][29]=u_1$,$A[3][2][29]=u_1$,$A[0][2][7]=u_2$, $A[0][2][16]=u_3$,$A[0][2][30]=u_4$,$A[1][2][4]=u_5$,$A[1][2][9]=u_6$,$A[1][2][13]=u_7$, $A[1][2][22]=u_7$,$A[3][1][24]=u_8$,$A[3][2][24]=u_8$,$A[2][1][4]=u_9$,$A[2][2][4]=u_9$, $A[2][1][6]=u_{10}$,$A[2][2][6]=u_{10}$,$A[2][1][15]=u_{10}$,$A[2][2][15]=u_{10}$,$A[2][2][8]=u_{11}$, $A[2][1][9]=u_{12}$,$A[2][2][9]=u_{12}$,$A[2][1][11]=u_{13}$,$A[2][2][11]=u_{13}$,$A[2][1][14]=u_{14}$, $A[2][2][14]=u_{14}$,$A[3][1][23]=u_{15}$,$A[3][2][23]=u_{15}$,$A[2][1][18]=u_{16}$,$A[2][2][18]=u_{16}$, $A[2][1][20]=u_{17}$,$A[2][2][20]=u_{17}$,$A[2][1][23]=u_{18}$,$A[2][2][23]=u_{18}$,$A[2][1][27]=u_{19}$, $A[2][2][27]=u_{19}$,$A[2][1][29]=u_{20}$,$A[2][2][29]=u_{20}$,$A[3][1][1]=u_{21}$,$A[3][2][1]=u_{21}$, $A[3][1][2]=u_{22}$,$A[3][1][8]=u_{23}$,$A[3][2][8]=u_{23}$,$A[3][1][31]=u_{23}$,$A[3][2][31]=u_{23}$, $A[3][1][10]=u_{24}$,$A[3][2][10]=u_{24}$,$A[3][1][11]=u_{25}$,$A[3][1][13]=u_{26}$,$A[3][2][13]=u_{26}$, $A[3][1][14]=u_{27}$,$A[3][2][14]=u_{27}$,$A[3][1][17]=u_{28}$,$A[3][2][17]=u_{28}$,$A[3][1][19]=u_{29}$, $A[3][2][19]=u_{29}$,$A[3][1][22]=u_{30}$,$A[3][2][22]=u_{30}$

Similarly, we also have run some experiments and obtained correct key recovery of 6-round Xoodyak with 100 percent success rate. In our verification experiments the key is generated randomly and all the controllable nonce bits are set to zero. The program is run in Visual Studio 2012 with x64 platform Release. Recovery a 6-bit key need about 60 hours using one CPU core (Intel i7 3.6GHz), and parallelism can reduce time. Using the test code that we provided, one can verify it easily.

We calculate the cube sum at lane $(0,0),(1,0),(2,0),(3,0),(0,1),(1,1)$, and the probability, for which the cube sum on these six lanes is zero, is 2^{-192} for a random function. Therefore, if the 33-dimension cube sums of 6-round output is zero, we declare that the key guess is correct with high probability. Actually, we also calculate other lanes' cube sum, and all of them turn out to be zero. The test code is given in `https://github.com/alicebobb/aabb/tree/alicebobb-patch-1`.

192-bit key K:

```
1010000011010110011101001101110001110010000111011101110010110110
1111110010011101001011000101010100010111101000111100101100000101
1110101001000010000111010101001100111000110100110101010001001010
```

The correct value for the guessed key bit in Table 6 is `001000`.

right key: `001000`

guessed value:001000, cube sums: `0x0, 0x0, 0x0, 0x0, 0x0, 0x0`

guessed value:000100, cube sums: `0xd2c47032,0x456c766f,0xd74569ed,`
`0x4a96a204,0xd6f503b8,0x6f6e9541`

guessed value:010000, cube sums: `0xa6dd98e3,0xdf935915,0xafbc25d9,`
`0x939c401d,0x95e04808,0x3caecd13`

6 Conclusion

In this paper, we give several practical key-recovery attacks on 5-round initialization of Ketje Jr v1 and v2, 6-round Xoodoo-AE in nonce-respecting setting and 6-round Xoodyak in nonce-reuse setting, whose time complexities are $2^{26.6}$, $2^{27.5}$, $2^{40.5}$ and $2^{43.8}$ with negligible memory cost. All the attacks are practically implemented.

References

BDL+18. Wenquan Bi, Xiaoyang Dong, Zheng Li, Rui Zong, and Xiaoyun Wang. MILP-aided Cube-Attack-Like Cryptanalysis on Keccak Keyed Modes. *Designs, Codes and Cryptography*, pages 1–26, 2018.

BDP+16. Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Ketje v2. *Submission to the CAESAR Competition*, 2016.

BDPA11. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 320–337, 2011.

BDPVA09. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak Sponge Function Family Main Document. *Submission to NIST (Round 2)*, 3(30), 2009.

BLD+18. Wenquan Bi, Zheng Li, Xiaoyang Dong, Lu Li, and Xiaoyun Wang. Conditional cube attack on round-reduced river keyak. *Des. Codes Cryptography*, 86(6):1295–1310, 2018.

Com14. The CAESAR Committee. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness, 2014. `http://competitions.cr.yp.to/caesar.html`.

DBH+. Joan Daemen, Guido Bertoni, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Innovations in permutation-based crypto. 21st Workshop on Elliptic Curve Cryptography, 2017. `https://ecc2017.cs.ru.nl/slides/ecc2017-daemen.pdf`.

DHAK18. Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of xoodoo and xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

DLWQ17. Xiaoyang Dong, Zheng Li, Xiaoyun Wang, and Ling Qin. Cube-like Attack on Round-Reduced Initialization of Ketje Sr. *IACR Trans. Symmetric Cryptol.*, 2017(1):259–280, 2017.

DMP+15. 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 *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, 2015.

DS09. Itai Dinur and Adi Shamir. Cube Attacks on Tweakable Black Box Polynomials. In *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, 2009.

FNR18. Thomas Fuhr, María Naya-Plasencia, and Yann Rotella. State-recovery attacks on modified ketje jr. *IACR Trans. Symmetric Cryptol.*, 2018(1):29–56, 2018.

HWX+17. Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional Cube Attack on Reduced-Round Keccak Sponge Function. In *Advances in Cryptology - EUROCRYPT 2017 - 36th*

Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II, volume 10211 of *Lecture Notes in Computer Science*, pages 259–288, 2017.

KMN10. Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145, 2010.

LBDW17. Zheng Li, Wenquan Bi, Xiaoyang Dong, and Xiaoyun Wang. Improved Conditional Cube Attacks on Keccak Keyed Modes with MILP Method. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 99–127, 2017.

LDB+19. Zheng Li, Xiaoyang Dong, Wenquan Bi, Keting Jia, Xiaoyun Wang, and Willi Meier. New conditional cube attack on keccak keyed modes. Cryptology ePrint Archive, Report 2019/392, 2019. `https://eprint.iacr.org/2019/392`.

LDW17. Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. Conditional cube attack on round-reduced ASCON. *IACR Trans. Symmetric Cryptol.*, 2017(1):175–202, 2017.

SG18. Ling Song and Jian Guo. Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP. *IACR Trans. Symmetric Cryptol.*, 2018(3):182–214, 2018.

SGSL18. Ling Song, Jian Guo, Danping Shi, and San Ling. New MILP Modeling: Improved Conditional Cube Attacks on Keccak-Based Constructions. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 65–95, 2018.

WY05. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

WYY05. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16, 2005.

A Experimental results on Ketje Jr v1, Ketje Jr v2

Table 8: Guess key and cube sum for 5-round KETJE JR v1

Guess key	**Cube sum**	**Guess key**	**Cube sum**	**Guess key**	**Cube sum**	**Guess key**	**Cube sum**
000000	0x91,0xe5	000010	0xc4,0xfe	000001	0xe9,0x49	000011	0xf4,0xfa
100000	0x9d,0x9b	100010	0xa6,0xfd	100001	0x22,0x71	100011	0xc7,0x19
010000	0xa9,0xa5	010010	0xf8,0x19	010001	0x9c,0x54	010011	0xdc,0x4d
110000	0x9c,0x69	110010	0xe2,0x43	110001	0x8c,0x38	110011	0x27,0x53
001000	0xf8,0xf1	001010	0x85,0x3c	001001	0xfc,0x95	001011	0xee,0x33
101000	0x1,0xa7	101010	0x2e,0x1c	101001	0x5b,0x2d	101011	0x21,0x81
011000	0x15,0xbb	011010	0x8d,0x7b	011001	0x31,0x46	011011	0xd7,0xae
111000	0x48,0x4e	111010	0xb2,0x96	111001	0xfb,0x2b	111011	0xed,0xaf
000100	0xa7,0xd6	000110	0xcb,0xb9	000101	0x81,0xfe	000111	0x3b,0x4e
100100	0x45,0x60	100110	0xf0,0x81	100101	0xe,0xd	100111	0x37,0xa3
010100	0x15,0x38	010110	0x88,0x52	010101	0x9f,0xd9	010111	0xcf,0x11
110100	0x3,0x9c	110110	0xd5,0x44	110101	0x40,0x1a	110111	0x43,0xba
001100	0x38,0x1e	**001110**	**0x0,0x0**	001101	0x45,0x67	001111	0x4b,0x31
101100	0x37,0xc9	101110	0xef,0xa5	101101	0x4a,0x18	101111	0x4,0x2f
011100	0xde,0xa7	011110	0x1b,0x49	011101	0xd9,0x21	011111	0x23,0x58
111100	0x0,0xce	111110	0xba,0x49	111101	0x8b,0x75	111111	0x44,0x8

Table 9: Guess key and cube sum for 5-round Ketje Jr v2

Guess key	**Cube sum**	**Guess key**	**Cube sum**	**Guess key**	**Cube sum**	**Guess key**	**Cube sum**
0000000	0x55,0xfd	0000010	0x7b,0xa5	0000001	0xcd,0x85	0000011	0x6b,0x9b
1000000	0x19,0x48	1000010	0xef,0x4b	1000001	0xab,0xfb	1000011	0x2d,0x5f
0100000	0x58,0x52	0100010	0x2a,0x4	0100001	0x87,0x19	0100011	0x43,0xed
1100000	0x86,0x20	1100010	0xb5,0x32	1100001	0x40,0x60	1100011	0x19,0x33
0010000	0x8c,0xaf	0010010	0x48,0xca	0010001	0x61,0x8a	0010011	0xce,0x61
1010000	0x8d,0x62	1010010	0xb0,0x8b	1010001	0x1a,0x23	1010011	0xf,0x2d
0110000	0x78,0x17	0110010	0x8f,0xf2	0110001	0x8b,0x89	0110011	0xc,0x6c
1110000	0x1c,0xec	1110010	0xcb,0x33	1110001	0x5e,0xa0	1110011	0xd4,0x73
0001000	0xa9,0x73	0001010	0xe4,0x82	0001001	0x0,0x83	0001011	0x59,0x92
1001000	0x25,0x39	1001010	0x38,0xb2	1001001	0xd5,0x29	1001011	0x57,0xff
0101000	0x43,0xc8	0101010	0xff,0x88	0101001	0xa7,0xb3	0101011	0x52,0xb4
1101000	0x55,0xa3	1101010	0x60,0x15	1101001	0x64,0x62	1101011	0x51,0x6d
0011000	0x29,0x1d	0011010	0x7d,0x3e	0011001	0x85,0xf	0011011	0xcb,0xc6
1011000	0xc7,0xbc	1011010	0xae,0x52	1011001	0xc2,0x27	1011011	0x64,0xd5
0111000	0xec,0xad	0111010	0xc,0xca	0111001	0xe3,0x71	0111011	0x7c,0xa6
1111000	0xe7,0x48	1111010	0xa4,0xe5	1111001	0x22,0x29	1111011	0xd1,0x77
0000100	0xc2,0xb9	**0000110**	**0x0,0x0**	0000101	0xf7,0x23	0000111	0x1a,0x37
1000100	0x23,0xdb	1000110	0x2e,0xea	1000101	0x2d,0xb9	1000111	0xe4,0x8b
0100100	0x65,0x33	0100110	0xc8,0x7f	0100101	0xc3,0x37	0100111	0x32,0xcc
1100100	0x90,0x87	1100110	0xcc,0xe7	1100101	0x4d,0xb2	1100111	0xa9,0xca
0010100	0x5d,0x0	0010110	0xed,0xad	0010101	0x3f,0xf1	0010111	0x97,0xd2
1010100	0x60,0xac	1010110	0xb6,0xe4	1010101	0xcf,0x9e	1010111	0xf7,0x25
0110100	0x54,0xd7	0110110	0x47,0x11	0110101	0xca,0x59	0110111	0x1c,0x4c
1110100	0x8e,0x6b	1110110	0xc6,0x50	1110101	0xed,0xb7	1110111	0x1d,0x6f
0001100	0x51,0x72	0001110	0x44,0xed	0001101	0x5,0x41	0001111	0xd2,0xfb
1001100	0xef,0x2a	1001110	0x2d,0xe	1001101	0x2a,0xf4	1001111	0x11,0x3a
0101100	0x28,0x68	0101110	0x15,0x41	0101101	0xe2,0xcc	0101111	0xa,0xe8
1101100	0xa2,0x1c	1101110	0x11,0x56	1101101	0x43,0xce	1101111	0x9a,0x79
0011100	0xe7,0xb9	0011110	0x42,0xc9	0011101	0x8f,0x26	0011111	0x88,0x4
1011100	0xef,0x89	1011110	0x66,0x67	1011101	0x6,0x9b	1011111	0xa2,0x9c
0111100	0x56,0xcd	0111110	0xea,0x43	0111101	0x6d,0x3f	0111111	0xac,0x85
1111100	0xc6,0x4b	1111110	0xc2,0x21	1111101	0x8b,0xff	1111111	0x20,0xec