IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 7, 2020, accepted January 12, 2020, date of publication January 17, 2020, date of current version February 4, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2967070*

# Improved Cube Attacks on Some Authenticated Encryption Ciphers and Stream Ciphers in the Internet of Things

**YU HE[1], GAOLI WANG[1], WENSHAN LI[1], AND YIZHI REN[2]**
[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
[2]Cyberspace School, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Gaoli Wang (glwang@sei.ecnu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802300, in part by the National Cryptography Development Fund under Grant MMJJ20180201, in part by the International Science and Technology Cooperation Projects under Grant 61961146004, and in part by the National Natural Science Foundation of China under Grant 61572125.

**ABSTRACT** With technical development and internet popularization, Internet of Things (IoT) technology is gaining a wider application in various fields. Key challenges in the growth of IoT are related to privacy and security. To avoid any possible malicious attacks, employing cryptosystems is widely recognized as one of the most effective approaches to implement confidentiality, integrity, and message authentication for the security of IoT. In this work, we investigate the security of Authenticated Encryption ciphers and stream cipher by using the improved cube attack. Firstly, we introduce a method to identify good cubes, which leads to the largest-round distinguisher. Our idea is based on the greedy algorithm of finding cubes and the numeric mapping method for estimating the algebraic degree of the NFSR-based cryptosystem. By using this method, we can efficiently explore useful cubes from a large search space. Further, we evaluate the security of several cryptographic primitives against the cube attack by using the SAT model of division property and flag technique, which can make the propagation of division property more accurately. Experiments show that we can obtain some new or improved cryptanalysis on MORUS-640-128, TRIAD, Quartet, TriviA-ck-v2, and Enhanced-bivium. The attacks can improve the number of attacking rounds and efficiency, and provide a vital reference for security analysis of other Authenticated Encryption ciphers and lightweight stream ciphers.

**INDEX TERMS** IoT, CAESAR, NIST, SAT, authenticated cipher, stream cipher, cube attack, division property.

## I. INTRODUCTION

With the rapid development and the wide use of IoT, there is a huge amount of data transferred over a network in order to communicate and share information. However, the IoT is a network with high dynamic topology, and their connections are vulnerable to attacks. As the benefits of IoT are numerous, the need for ensuring the security and the reliability of the overall system is growing. Nowadays, protecting the overall system against malicious attacks is one of the greatest challenges for deploying IoT technology.

In order to improve cryptographic defenses against cyber-attacks, the U.S. National Institute of Standards and Technology (NIST) announced an open competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) in January 2013 [1] to encourage the design and analysis of AE ciphers. Specifically, authenticated ciphers combine the cryptographic services of confidentiality, integrity, and authentication into one algorithmic construct. Moreover, IoT applications ranging from smart locks to wearable technology to home automation and healthcare, they often require constrained devices, and these innumerable tiny trustworthy devices are required to provide services such as encryption\decryption, authentication, digital signature and so on. To create worthy solutions to the problem of securing data in the IoT-like constrained environments, NIST published the call in August 2018 [2] for proposals for a new lightweight cryptography (LWC) standard process for lightweight applications.

The associate editor coordinating the review of this manuscript and approving it for publication was Shagufta Henna.

MORUS family [3] is one of the finalists of the CAESAR competition and has received extensive attention. TriviA-ck-v2 [4] is one of the Round 2 candidates in the CAESAR competition. TRIAD [5] and Quartet [6] are two lightweight symmetric-key schemes based on a stream cipher, and both of them are selected as LWC authenticated cipher candidates in April 2019. Cube attack is one of the most powerful techniques in cryptanalysis of cryptographic primitives, and this motivates us to study the security of these above ciphers against the cube attack.

### A. RELATED WORK

Cube attack is a powerful technique to analyze symmetric-key ciphers proposed by Dinur and Shamir [7] in 2009. Cube attack can be seen as a generalization of higher-order differential attack [8] and chosen IV statistical attack [9], [10]. The core idea of the cube attack is to simplify the polynomial by summing the output of the cryptosystem over a subset of public variables, so-called cube. The target of the cube attack is to recover secret variables from the simplified polynomial named superpoly. Cube attack has been successfully used to various ciphers, including stream ciphers [11]–[15], hash functions [16]–[18], and authenticated encryptions [19], [20]. Originally, stream cipher can be treated as a blackbox polynomial. We can recover the structure of the superpoly with a linearity test [7] or a quadraticity test [21]. Later, cube attacks develop into different types, such as dynamic cube attack [12], conditional cube attack [17], correlation cube attack [15], deterministic cube attack [22], and division property based cube attack.

Todo [23] introduces division property at EUROCRYPT 2015, which is a generalization of integral property. It can be used to construct the integral distinguisher against block ciphers with non-linear components. After that, Todo and Morii [24] present the bit-based division property at FSE 2016, which is a more robust method because it can trace the division property at the bit level. However, it is evident that the memory and computing complexities significantly increase. The complexity of utilizing bit-based division property was roughly equal to $2^n$ for $n$-bit primitives, so the considerable complexity restricted the wide applications of bit-based division property. To overcome this limit, at ASIACRYPT 2016, Xiang *et al.* [25] apply an auxiliary Mixed-Integer Linear Programming (MILP) tool to search integral distinguishers with conventional division property. They innovatively introduce the method to apply inequalities to model the basic bitwise operations COPY, AND, XOR, and an S-box operation. The larger integral distinguishers can be found efficiently through this MILP aided technique. Sun *et al.* [26] proposed the automatic search method that based on SAT\SMT to search integral distinguishers for ARX-based ciphers.

Further, Todo *et al.* [27] put cube attacks in the setting of non-blackbox polynomials and apply the division property to the cube attack at CRYPTO 2017. By utilizing this new cube attack technique, they explore cubes with a larger size, and successfully evaluate the involved secret variables on stream ciphers TRIVIUM, Grain128a, and authenticated cipher ACORN. Soon after, Wang *et al.* [28] present several techniques (flag technique, degree evaluation, and term enumeration) to enhance the performance of division property based cube attack at CRYPTO 2018. With the help of bit-based division property using three subsets, Wang *et al.* [29] apply their improved cube attack technique on Trivium in practice and propose a theoretical attack that can recover the superpoly of Trivium up to 842 rounds. Very recently, Yang *et al.* [30] apply a new method of finding cube testers to ACORN, which is based on the greedy algorithm of finding cubes [31], [32] and the numeric mapping technique [33] for iteratively estimating the upper bound of the algebraic degree of the NFSR-based cryptosystem.

Salam *et al.* [14] recover the key with the cube attack on the initialization of 4-step MORUS-640-128 in 2017. In 2019, Li and Wang [34] apply the MILP tool to analyze its 5.5-step variant by utilizing the technique of cube attacks based on division property. TriviA-ck-v2 [4] is one of the Round 2 candidates in the CAESAR competition. In 2017, Sarkar *et al.* [35] obtain distinguishers till 950 rounds. Liu [33] provides a distinguishing attack for its 1047-round variant with a cube of size 61 in CRYPTO 2017. It is noted that cube attacks in our paper are key recovery attacks, which aim to guess at least one bit of secret key with complexity less than the brute-force attack. However, cube attack by utilizing such large cube is unfeasible under current limited computing resources. Zhang *et al.* [36] apply the cube attack on 464-round Enhanced-bivium and claim that the breaking complexity is $2^{55}$ after 464 rounds. Quartet and TRIAD are Round 1 submissions of the ongoing NIST Lightweight Cryptography Standardization Project in April 2019. To the best of our knowledge, none of the key recovery attacks with the cube technique has been conducted on them.

Moreover, in the previous attacks on MORUS-640-128 and Enhanced-bivium, they randomly select a subset of public variables as the cube from a large search space. Aiming to extend the number of attacked rounds and reduce computing complexity, we introduce an algorithm to find expected cubes. Then we apply the improved division property based cube attack on these target ciphers.

### B. OUR CONTRIBUTIONS

In this paper, we introduce a new greedy method to discover cubes from a wide range of initialization vectors efficiently, and we also use the numeric mapping method to iteratively estimate the upper bound of the algebraic at a linear computational complexity. We model the division property propagations of five target ciphers and translate the division trail finding problem into an SAT problem. Further, for the bitwise AND operation, the effect of constant 0 bits is taken into account. We use the algorithm combined with the SAT model of the division property and the flag technique to improve the accuracy of the propagation of division properties.

**TABLE 1. Summarization of key recovery attack results.**

| Target | Attack length | Cube size | Complexity | References |
|---|---|---|---|---|
| MORUS-640-128 | 5.5-step | 22 | $2^{67}$ | [34] |
| | 5.9-step | 24 | $2^{32.06}$ | This paper |
| | 4-step | 3 | $2^{9.98}$ | [14] |
| | 5-step | 3 | $2^{4}$ | This paper |
| | 6-step | 22 | $2^{35.55}$ | This paper |
| TRIAD | 521 | 16 | $2^{24.81}$ | This paper |
| | 565 | 30 | $2^{48}$ | This paper |
| Quartet | 3-step | 2 | $2^{4}$ | This paper |
| | 4-step | 18 | $2^{23.81}$ | This paper |
| TriviA-ck-v2 | 867 | 12 | $2^{20.46}$ | This paper |
| | 874 | 16 | $2^{22.48}$ | This paper |
| Enhanced-bivium | 464 | 11 | $2^{55}$ | [36] |
| | 502 | 12 | $2^{18.19}$ | This paper |

Finally, we evaluate the security of MORUS-640-128, TRIAD, Quartet, Trivia-ck-v2, and Enhanced-bivium by using the improved cube attack based on the division property with the SAT method, and we give some new cryptanalysis results in key recovery attack.

For MORUS-640-128 with keystream generation function, we give the first 5.9-step key recovery attack with a cube of size 24, which achieves 0.4-step (two rounds) more than the previous best work in [34], and the complexity for superpoly recovery is $2^{32.06}$. We also study reduced versions of the initialization of MORUS-640-128 and provide the 6-step key recovery attack with a cube of size 22, which improves two steps comparing with [14]. Also, we find better cubes of the same size as [14], and then we gain one more step key recovery attack with these cubes. For TRIAD, we obtain a zero-sum distinguisher for 559-round with a cube of size 35 and a 568-round distinguisher with a cube of size 45. We find a cube of size 16, which leads to a key recovery attack for its 521 variant, the complexity is $2^{24.81}$. We provide a key recovery attack for 565-round TRIAD with the 30-dimensional cube, the complexity is $2^{48}$. Also, the theoretical key recovery attack is applied to its 572-round variant. We obtain cubes of size 2 and 18, resulting in key recovery attacks for 3 and 4 rounds of Quartet. We provide cube attacks on TriviA-ck-v2 up to 874 rounds with a cube of size 16. We also provide the first key recovery attack for 502-round Enhanced-bivium with a cube of size 12. Our results are summarized in Table 1, and the comparisons with former key recovery attacks are also included.

### C. ORGANIZATION

The remainder of this paper is organized as follows: Some basic definitions and theories are introduced in Section II. In Section III we briefly describe MORUS-640-128, TRIAD, Quartet, Enhanced-bivium and TriviA-ck-v2. Section IV shows the method of finding good cubes and an algorithm of using the SAT model combined with the flag technique. Section V presents several applications. Section VI provides conclusions and future work.

## II. PRELIMINARIES

### A. CUBE ATTACK

Cube attack is an analysis tool to recover the secret key of a cryptosystem, which was proposed by Dinur and Shamir [7].

Almost any cryptosystem can be seen as a Boolean function. For a stream cipher with n secret key bits $x = (x_0, x_1, \ldots, x_{n-1})$ and m public initialization vector (IV) bits $v = (v_0, v_1, \ldots, v_{m-1})$, the algebraic normal form (ANF) of the given output bit can be regarded as a Boolean polymonial $f(x, v)$ over the field $GF(2)$. For a specific subset of indexes $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{0, 1, \ldots, m-1\}$, the terms $t_I = v_{i_1} \cdots v_{i_{|I|}}$ in the polynomial, and the $2^{|I|}$ values of indexes are referred as cube $C_I$. Then the ANF of $f(x, v)$ can be decomposed as

$$f(x, v) = t_I \cdot p(x, v) + q(x, v), \tag{1}$$

where $p(x, v)$ is called the superpoly of $I$ in $f(x, v)$ and it is irrelevant to $\{v_{i_1}, v_{i_2} \cdots v_{i_{|I|}}\}$. Obviously, the value of $p(x, v)$ can merely be affected by the secret key bits $x$ and the assignment to the non-cube $IV$s. Furthermore, every term in $q(x, v)$ at least misses one variable from the set $I$. A major observation in cube attacks is that the sum of $f(x, v)$ over all values of the cube $C_I$ is

$$\oplus_{C_I} f(x, v) = \oplus_{C_I} t_I \cdot p(x, v) + \oplus_{C_I} q(x, v) = p(x, v). \tag{2}$$

### B. DIVISION PROPERTY AND DIVISION TRAIL

Division property is a generalization of integral property, proposed by Todo [23] at EUROCRYPT 2015. In this subsection, we will recall the definitions of bit-based division property [24] and division trail [25].

*Definition 1 (Bit-Based Division property [24]): Let $\mathbb{X}$ be a multiset whose elements take values from $\mathbb{F}_2^n$, and let $\mathbb{K}$ be a set of m-dimensional vectors whose i-th element takes 0 or 1. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, the parity of $\boldsymbol{\pi}_u(x)$ over all $x \in \mathbb{X}$ satisfies the following conditions:*

$$\bigoplus_{x \in \mathbb{X}} \boldsymbol{\pi}_u(x) = \begin{cases} unknown, & if\ there\ is\ k \in \mathbb{K},\ u \geq k \\ 0, & otherwise. \end{cases} \tag{3}$$

There is $u \geq k$, if $u[j] \geq k[j]$ for all $j$.

*Definition 2 (Division trail [25]): Let $f$ be the round function of an iterated cipher and $\mathbb{K}_0$ be the initial division property state. After r rounds, the division properties denoted by $\mathbb{K}_r$, can be deduced through $f$. Thus, the propagation of the division property from $\mathbb{K}_0$ to $\mathbb{K}_r$ is evaluated as $\{k\} \overset{def}{=} \mathbb{K}_0 \xrightarrow{f} \mathbb{K}_1 \xrightarrow{f} \cdots \xrightarrow{f} \mathbb{K}_i \cdots \xrightarrow{f} \mathbb{K}_r$, where $\mathbb{K}_i$ denotes the internal states division property after i rounds $(0 \leq i \leq r)$. For any vector $k_i \in \mathbb{K}_i$, there must exist a vector $k_{i-1} \in \mathbb{K}_{i-1}$ can propagate to $k_i$ for $i \in \{1, \cdots, r\}$, then we call $k_0 \rightarrow k_1 \rightarrow \cdots \rightarrow k_r$ is a r-round division trail.*

Moreover, if there is no division trail $k_0 \xrightarrow{f} k_r = e_j$, where $e_j$ is an $n$-bit unit vector whose $j$th bit is 1 and others are 0, then the $j$th output bit after $r$-round $f$ functions is balanced. Otherwise, the $j$th bit may be balanced or not.

### C. SAT-AIDED BIT-BASED DIVISION PROPERTY

Sun *et al.* [26] proposed the automatic search method that based on SAT\SMT to search integral distinguishers for ARX-based ciphers. They model the bit-based division property propagations of the three basic operations (COPY, AND, and XOR) in Conjunctive Normal Form (CNF). The concrete equations are depicted as follow.

*Model 1 (SAT Model for COPY [26]): Let a* $\xrightarrow{COPY}$ $(b_0, b_1)$ *be a division trail of COPY function. The following CNFs are sufficient to describe the propagation of the division property for COPY.*

$$\begin{cases} \neg b_0 \vee \neg b_1 = 1 \\ a \vee b_0 \vee \neg b_1 = 1 \\ a \vee \neg b_0 \vee b_1 = 1 \\ \neg a \vee b_0 \vee b_1 = 1 \end{cases} \tag{4}$$

*Model 2 (SAT Model for XOR [26]): Let* $(a_0, a_1) \xrightarrow{XOR} b$ *be a division trail of XOR function. The following CNFs are sufficient to describe the propagation of the division property for XOR.*

$$\begin{cases} \neg a_0 \vee \neg a_1 = 1 \\ a_0 \vee a_1 \vee \neg b = 1 \\ a_0 \vee \neg a_1 \vee b = 1 \\ \neg a_0 \vee a_1 \vee b = 1 \end{cases} \tag{5}$$

*Model 3 (SAT Model for AND [26]): Let* $(a_0, a_1) \xrightarrow{AND} b$ *be a division trail of AND function. The following CNFs are sufficient to describe the propagation of the division property for AND.*

$$\begin{cases} \neg a_1 \vee b = 1 \\ a_0 \vee a_1 \vee \neg b = 1 \\ \neg a_0 \vee b = 1 \end{cases} \tag{6}$$

### D. CUBE ATTACK BASED ON DIVISION PROPERTY

In cube attack, attackers aim to recover the superpoly $p(x, v)$ for a cube and get some secret key bits information. Todo *et al.* [27] proposed the improved cube attack technique based on division property with MILP. They can identify the secret key bits excluded from the superpoly and explore larger-size cubes. The technique is based on the following proposition.

*Proposition 1 (Identify the involved keys by Division Property [27]): Let* $f(x, v)$ *be a polynomial over the field GF* (2)*, where* $x$ *and* $v$ *denote the secret and public variables respectively. For a subset of indexes* $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{0, 1, \ldots, m-1\}$*, let* $C_I$ *be a set of* $2^{|I|}$ *values where the variables in* $\{v_{i_1} \cdots v_{i_{|I|}}\}$ *are taking all possible combinations of values. Let* $k_I$ *be an* $n$*-dimensional vector such that* $v^{k_I} = t_I = v_{i_1} v_{i_2} \cdots v_{i_{|I|}}$*, i.e.* $k_i = 1$ *if* $i \in I$ *and* $k_i = 0$ *otherwise. Assuming there is no division trail such that* $(e_j, k_I) \xrightarrow{f} 1$*,* $x_j$ *is not involved in the superpoly of the cube* $C_I$*.*

*Flag Technique:* Sun *et al.* [37] firstly present the MILP model for division property of the bitwise AND operation between a variable and one constant. Later, Todo *et al.* take into account the impact of constant 0 bits on the propagation of division property, and add several additional constraints in [27]. Afterwards, Wang et al. [28] find out that constant 0 bits could be generated from some XOR operations, which involve an even number of constant 1 bits. For this reason, Wang et al. propose the flag technique to make their MILP model more precisely. The operation $\oplus$ follows the rules:

$$\begin{cases} 1_c \oplus 1_c = 0_c \\ 0_c \oplus x = x \oplus 0_c = x \\ \delta \oplus x = x \oplus \delta = \delta \end{cases} \quad \text{for arbitrary } x \in \{1_c, 0_c, \delta\}. \tag{7}$$

The operation $\times$ follows the rules:

$$\begin{cases} 1_c \times x = x \times 1_c = x \\ 0_c \times x = x \times 0_c = 0_c \\ \delta \times \delta = \delta \end{cases} \quad \text{for arbitrary } x \in \{1_c, 0_c, \delta\}. \tag{8}$$

After getting the cube index $I$ and index set $J$ of probably involved secret variables by the above-mentioned algorithms, Wang *et al.* [28] evaluate the algebraic degree and enumerate all possible terms of the superlopy to recover the superpoly efficiently.

*Complexity for Superpoly Recovery:* With cube indices $I$, key bits $J$ and degree $d$, enumerates all $t$-degree mononials that may appear in the superpoly. Let $J_i$ denotes the set contains all possilbe monomials of degree $i$, for any $i \in (1, 2, \cdots, d)$. The comlexity for superpoly recovery is

$$2^{|I|} \times (1 + \sum_{t=1}^{d} |J_t|). \tag{9}$$

## III. A BRIEF DESCRIPTION OF FIVE TARGET CIPHERS

### A. THE DESCRIPTION OF MORUS-640-128

MORUS is an authenticated encryption cipher submitted to the CAESAR competition, and it is of great concern recently.

The design of MORUS is based on the stream cipher, and four parts include: initialization, associated data processing, encryption and finalization. For MORUS-640-128, the internal state is 640 bits and divided into five 128-bit state elements denoted as $S_{0,0}, \cdots, S_{0,4}$. The initialization starts by loading the 128-bit key $K_{128}$ and the 128-bit initialization vector $IV_{128}$ into the state together with constants $const_0$, $const_1$. The value of above mentioned constants and rotation constants can be referred to [3]. The initial state bits are reprented as follow.

$$\begin{aligned} S_{0,0}^{-16} &= IV_{128}; \\ S_{0,1}^{-16} &= K_{128}; \\ S_{0,2}^{-16} &= 1^{128}; \\ S_{0,3}^{-16} &= const_0; \\ S_{0,4}^{-16} &= const_1. \end{aligned} \tag{10}$$

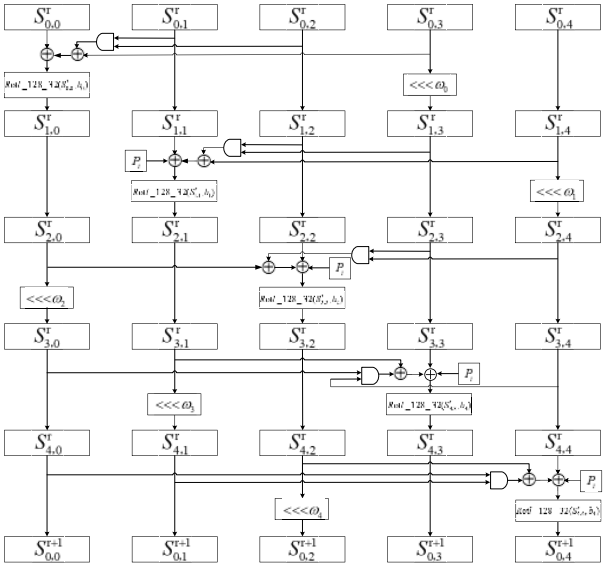


**FIGURE 1.** The state update function of MORUS-640-128.

The state $S$ is updated by calling the round function StateUpdate for 16 steps. There are 5 rounds with similar bitwise operations at each step as illustrated in Figure 1. The update function StateUpdate($S^i$, $P_i$) is given as follow.

$$
\begin{aligned}
&\textit{Round } 0:\\
&S^i_{1,0} = Rotl\left(S^i_{0,0} \oplus \left(S^i_{0,1} \S^i_{0,2}\right) \oplus S^i_{0,3}, b_0\right);\\
&S^i_{1,3} = S^i_{0,3} \lll \omega_0;\\
&S^i_{1,1} = S^i_{0,1};\\
&S^i_{1,2} = S^i_{0,2};\\
&S^i_{1,4} = S^i_{0,4};\\
&\qquad \textit{For } 1 \le j \le 4,\\
&\textit{Round } j:\\
&S^i_{j+1,j} = Rotl\left(S^i_{j,j} \oplus \left(S^i_{j,j+1} \S^i_{j,j+2}\right) \oplus S^i_{j,j+3} \oplus P_i, b_j\right);\\
&S^i_{j+1,j+3} = S^i_{j,j+3} \lll \omega_j;\\
&S^i_{j+1,j+1} = S^i_{j,j+1};\\
&S^i_{j+1,j+2} = S^i_{j,j+2};\\
&S^i_{j+1,j+4} = S^i_{j,j+4}.
\end{aligned}
\tag{11}
$$

Only two of the state elements are updated in every round: one is modified by rotation $\lll$, and the other by a set of ANDs, XORs, and rotation $Rotl$. The $Rotl\left(x, b_j\right)$ for $b \in (0, \cdots, 4)$ indicates that a 128-bit block $x$ is divided into four 32-bit words and each word is rotated left by $b_j$ bits, the values of $b_j$ can be referred to [3]. The addition between subscripts is actually addition modulo 5.

The plaintext block $P_i$ is encrypted to the ciphertext $C_i$ by the exclusive OR operation with the key stream $S_0 \oplus (S_1 \lll 96) \oplus (S_2 \& S_3)$ in the encryption phase.

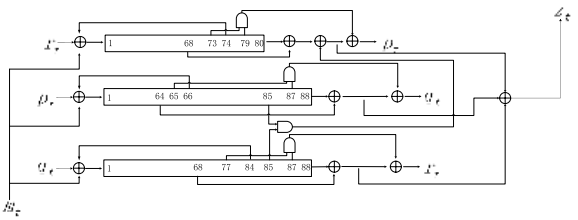


**FIGURE 2.** The state update function of TRIAD.

### B. THE DESCRIPTION OF TRIAD

TRIAD is a family of lightweight symmetric-key schemes based on a stream cipher. The 256-bit internal state of TRIAD denoted by $\boldsymbol{a} = (a_1 \| a_2 \| \cdots \| a_{80})$, $\boldsymbol{b} = (b_1 \| b_2 \| \cdots \| b_{88})$ and $\boldsymbol{c} = (c_1 \| c_2 \| \cdots \| c_{88})$. It accepts a 128-bit secret key $K$ and a 96-bit nonce $N$ represented by byte arrays as follow.

$$
\begin{aligned}
K &= (K[0], K[1], K[2], \cdots, K[15]);\\
N &= (N[0], N[1], N[2], \cdots, N[11]).
\end{aligned}
\tag{12}
$$

The initialization of TRIAD loads the secret key, nonce, and some constants into the state, and please refer to [5] for more details. The internal state is updated by Algorithm 1 for 1024 rounds. The input is 256-bit internal state $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ and 1-bit message $msg$ and the output is the updated internal state and 1-bit key stream $z$. Figure 2 shows the state update function of TRIAD.

**Algorithm 1** Update Function of TRIAD

$t_1 \leftarrow a_{68} \oplus a_{80} \oplus b_{85} \cdot c_{85}$
$t_2 \leftarrow b_{64} \oplus b_{88}$
$t_3 \leftarrow c_{68} \oplus c_{88}$
$z \leftarrow t_1 \oplus t_2 \oplus t_3$
$t_1 \leftarrow t_1 \oplus a_{73} \cdot a_{79} \oplus b_{66} \oplus msg$
$t_2 \leftarrow t_2 \oplus b_{65} \cdot b_{87} \oplus c_{84} \oplus msg$
$t_3 \leftarrow t_3 \oplus c_{77} \cdot c_{87} \oplus a_{74} \oplus msg$
$(a_1, a_2, \cdots, a_{80}) \leftarrow (t_3, a_1, \cdots, a_{79})$
$(b_1, b_2, \cdots, b_{88}) \leftarrow (t_1, b_1, \cdots, b_{87})$
$(c_1, c_2, \cdots, c_{88}) \leftarrow (t_2, c_1, \cdots, c_{87})$

### C. THE DESCRIPTION OF TRIVIA-CK-V2

TriviA-ck-v2 is an authenticated encryption algorithm submitted to the CAESAR competition.

TriviA-ck-v2 has the 384-bit internal state and uses three Non-linear feedback registers $A$, $B$, $C$ of size 132 bit, 105 bit, and 147 bit respectively. It is loaded with the 128-bit key and the 128-bit initialization vector IV. Algorithm 2 describes the state update function of TriviA-ck-v2.

After the state updation, keystream bits can be generated. The keystream bit generation function is defined as

$$
A_{66} \oplus A_{132} \oplus B_{69} \oplus B_{105} \oplus C_{66} \oplus C_{147} \oplus (A_{102} \cdot B_{66}). \tag{13}
$$

**Algorithm 2** Update Function of TriviA-ck-v2

$t_1 \leftarrow A_{66} \oplus A_{132} \oplus (A_{130} \cdot A_{131}) \oplus B_{96}$
$t_2 \leftarrow B_{69} \oplus B_{105} \oplus (B_{103} \cdot B_{104}) \oplus C_{120}$
$t_3 \leftarrow C_{66} \oplus C_{147} \oplus (C_{145} \cdot C_{146}) \oplus A_{75}$
$(A_1, A_2, \cdots, A_{132}) \leftarrow (t_3, A_1, \cdots, A_{131})$
$(B_1, B_2, \cdots, B_{105}) \leftarrow (t_1, B_1, \cdots, B_{104})$
$(C_1, C_2, \cdots, C_{147}) \leftarrow (t_2, C_1, \cdots, C_{146})$

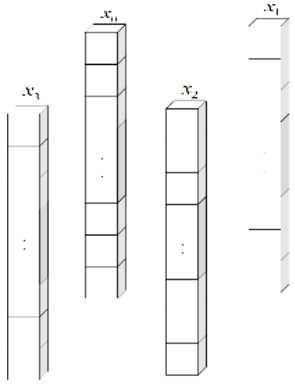


**FIGURE 3.** Four 64-bit lanes in Quartet.

### D. THE DESCRIPTION OF ENHANCED-BIVIUM

Enhanced-bivium is an NFSR-based stream cipher [38]. The 174-bit internal state of Enhanced-bivium can be denoted as $s = (s_0, s_1, \cdots, s_{173})$. The 80-bit key $K$ and the 80-bit intialization vector IV are loaded as follow. It requires 628 cycles in the initialization phase.

$$\begin{aligned}(s_0, s_1, \cdots, s_{89}) &= (K_0, K_1, \cdots, K_{79}, 0, \cdots, 0)\,; \\ (s_{90}, s_{91}, \cdots, s_{173}) &= (IV_0, IV_1, \cdots, IV_{79}, 0, 1, 1, 1)\,. \end{aligned} \quad (14)$$

The update function is given in Algorithm 3. The internal state is updated for $R$ rounds, then produce the 1-bit keystream denoted as $z$.

**Algorithm 3** Update Function of Enhanced-bivium

**for** $i = 1$ to $R$ **do**
  $t_1 \leftarrow s_{62} + s_{89}$
  $t_2 \leftarrow s_{92} + s_{173}$
  $z_i \leftarrow t_1 + t_2$
  $t_1 \leftarrow t_1 + s_{87} \cdot s_{88} + s_{137}$
  $t_2 \leftarrow t_2 + s_{171} \cdot s_{172} + s_{74}$
  $(s_0, s_1, \cdots, s_{89}) \leftarrow (t_2, s_0, \cdots, s_{88})$
  $(s_{90}, s_{91}, \cdots, s_{173}) \leftarrow (t_1, s_{90}, \cdots, s_{172})$
**end for**

### E. THE DESCRIPTION OF QUARTET

Quartet is one of the submissions of the ongoing NIST Lightweight Cryptography Standardization Project. As depicted in Figure 3, there are four 64-bit lanes involved in the algorithm: $x_0, x_1, x_2$ and $x_3$.

The state updating of Quartet has four functions. The only non-linear function $\chi$ is defined as

$$x_i \leftarrow x_i \oplus (x_{i+2} \oplus 1) \cdot x_{i+1}. \quad (15)$$

It operates on the 64-bit lanes, which updates $x_i$ by taking $x_i$, $x_{i+1}$ and $x_{i+2}$ as inputs with the index addition being the addition modulo 4. The $\rho$ function is a linear function, which divides the 64-bit lane $x_i$ into two 32-bit words and rotates each 32-bit word left by the same number of bits $n_i$. The $\lambda$ function is defined as bellow, which operates on one 64-bit lane $x_i$.

$$x_i \leftarrow x_i \oplus (x_i \ggg_{64} r_{i,1}) \oplus (x_i \ggg_{64} r_{i,2}). \quad (16)$$

The $\tau$ function is used to add the round constant $c_r$ to the lane $x_3$, for $0 \le r \le 23$. The round constants $c_r$ and the rotation parameters $n_i$, $r_{i,1}$ and $r_{i,2}$ can be referred to [6].

In the initialization phase, one round $R_{ini}$ of the state updating consists of a series of operations done on the four 64-bit lanes, where $\circ$ is the composition operation of the different mappings.

$$\begin{aligned} R_{ini} = {} & \tau \circ \lambda(x_2, r_{2,1}, r_{2,2}) \circ \rho(x_1) \circ \chi(x_3, x_0, x_1) \circ \\ & \lambda(x_1, r_{1,1}, r_{1,2}) \circ \rho(x_0) \circ \chi(x_2, x_3, x_0) \circ \\ & \lambda(x_0, r_{0,1}, r_{0,2}) \circ \rho(x_3) \circ \chi(x_1, x_2, x_3) \circ \\ & \lambda(x_3, r_{3,1}, r_{3,2}) \circ \rho(x_2) \circ \chi(x_0, x_1, x_2). \end{aligned} \quad (17)$$

## IV. METHODS TO SEARCH GOOD CUBES AND IMPROVED CUBE ATTACK WITH SAT

The goal of the cube attack is to recover the secret key of a cryptosystem. The main observation of the cube attack is that summing the outputs of the cryptosystem over a cube might cancel out all the higher degree terms except terms associated with the monomials involving the cube variables. Hence, exploring useful cubes are vital to cube attack. However, the number of possible cube choices will increase significantly with the increase of cube sizes during the cube attack. In this section, we will discuss how to find expected cubes more efficiently by utilizing the numeric mapping technique. Moreover, this section describes an algorithm to combine the SAT model of the division property with the flag technique.

### A. THE NUMERIC MAPPING TECHNIQUE

The numeric mapping technique is proposed at CRYPTO 2017 to iteratively estimate the upper bound on the algebraic degree of the NFSR-based cryptosystem [33]. In the following, we will shortly review the technique.

Let $f(x) = \bigoplus_{u=(u_1,\cdots,u_n)\in\mathbb{F}_2^n} a_u^f \prod_{i=1}^n x_i^{u_i}$, and $\mathbb{B}_n$ denotes the set of all $n$-variable Boolean functions. We denote coefficients of the ANF of $f$ as $a_u^f$. The numeric mapping denoted by $DEG$ is defined as below, where $D = (d_1, d_2, \cdots, d_n)$.

$$\begin{aligned} DEG:\ & \mathbb{B}_n \times \mathbb{Z}^n \to \mathbb{Z}, \\ & (f, D) \to \max_{a_u^f \neq 0} \left\{ \sum_{i=1}^n u_i d_i \right\}. \end{aligned} \quad (18)$$

Suppose $g_i (1 \leq i \leq n)$ are Boolean functions on $m$ variables, and denote $deg(G) = (deg(g_1), deg(g_2), \cdots, deg(g_n))$ for $G = (g_1, g_2, \cdots, g_n)$. The numeric degreee of the composite function $h = f \circ G$ is defined as $DEG(f, deg(G))$, denoted by $DEG(h)$ for short. We call $DEG(f, D)$ a super numeric degree of $h$ if $d_i \geq deg(g_i)$ for all $1 \leq i \leq n$, where $D = (d_1, d_2, \cdots, d_n)$. We can check that the algebraic degree of $h$ is always less than or equal to the numeric degree of $h$, *i.e.*,

$$deg(h) \leq DEG(h) = \max_{a_u^f \neq 0} \left\{ \sum_{i=1}^{n} u_i deg(g_i) \right\}. \quad (19)$$

### B. A NEW GREEDY ALGORITHM TO FIND EXPECTED CUBES

Before reaching the peak, the number of distinguishable rounds generally becomes bigger as the size of the cube grows larger, then it will turn small gradually. In fact, when the cube has a relatively large size, the number of rounds we can attack becomes larger. Inspired by this, we present a new greedy algorithm to find good cubes by discarding several bits from the candidate list iteratively. In addition, we give the time complexity analysis in this section.

In our greedy algorithm, all of the IV bits are selected as the initial candidate bits, namely $s_0$ is equal to the length of IV. We search expected cubes by iteratively discarding $k_i$ bits from the previous candidate list. Further, to reduce trivial operations, we set the size of candidates to a fixed value, denoted by $n$. The main process is as follow:

1) Choose a list of $s_{i-1}$-bit candidate cubes from an initial starting set, or from the results of the previous iteration. This size of the list is denoted by $n$.
2) Drop $k_i$ bits from each one of the candidate cubes. By using the numeric mapping technique, select the best candidates of size $s_{i-1} - k_i$, which leads to distinguishers or nonrandomness detectors with the largest number of rounds.
3) For all original candidate cubes of size $s_{i-1}$, their corresponding new best candidates of size $s_{i-1} - k_i$ are stored in a new list together. According to the numbers of the distinguishable round, sort the new list from largest to smallest.
4) Reduce the size of the sorted list to $n$ by pruning some of the elements with the smaller distinguishable round.
5) Repeat steps 1 to 4 until the cube of the expected size has been found.

Obviously, the whole complexity of both the accelerated greedy algorithm [30] and our new greedy algorithm mainly depends on the size of IV, discarding or adding size $k_i$, and the complexity of finding the largest distinguishable round. Similarly, we calculate the lower bound of the maximum number of distinguishable rounds by using a degree estimation algorithm. Especially for NFSR-based cryptosystems, we use the numeric mapping technique [33] to estimate the lower bound of the maximum number of distinguishable rounds for the test cube. The time complexity of the degree estimation algorithm is about $O(N)$, where $N$ is the number of initialization rounds.

### C. COMBINATION OF SAT MODEL AND FLAG TECHNIQUE

In the previous model of cube attack based on division property, the initial division properties of cube IV bits are set to active, denoted by 1. And the other bits are initialized to constant bits, denoted by 0. Wang *et al.* [28] introduce the flag technique, which takes the effect of constant bits on the propagation of division property into consideration, then the model can be more precise.

For specific ciphers, the flags of the internal states can be derived from update functions iteratively without the automatic solver SAT. Algorithm 4 shows the combination of the SAT model and the flag technique. Hence the accuracy of the SAT model of the division property propagation will be increased.

**Algorithm 4** SAT Model Combined With Flag Technique

Declare $X^j$ as the division property of $j$th round internal state
Declare $operation_m^j$ as the $m$th operation in $j$th round state update function or keystream generation function
Declare $f^j$ as the flag of $j$th round state
**while** $operation_m^j(X_{a_0}^{j-1}, \cdots, X_{a_n-1}^{j-1})$ is not NULL **do**
  **if** $operation_m^j$ is AND **then**
    **if** arbitrary $X_i^{j-1}$ for any $i \in (a_0, \cdots, a_{n-1})$, the corresponding flag $f_i^{j-1}$ is $0_c$ **then**
      $0 \leftarrow operation_m\left(X_{a_0}^{j-1}, \cdots, X_{a_n-1}^{j-1}\right)$
    **else**
      follow the traditional SAT propagation rules
    **end if**
  **else**
    follow the traditional SAT propagation rules
  **end if**
**end while**
**return** $X^j$

## V. APPLICATIONS

With the methods proposed in Section 4, we are able to find good cubes from a large search space efficiently. As an illustration, we apply our greedy algorithm to explore expected cubes for MORUS-640-128, TRIAD, Quartet, TriviA-ck-v2, and Enhanced-bivium, then we apply SAT-aided bit-based division property to conduct cube attacks in this section. As authenticated encryption primitives, we only focus on the process of the initialization phase because the number of rounds we can attack is smaller than the whole initialization rounds.

In the experiments, the numeric mapping technique is used to estimate the algebraic degree of the output key bit more precisely. Note that, regardless of the dimension of the cube, the estimation can give an upper bound of the real algebraic degree of the output key bit. If the estimated degree of the

output bits is smaller than the size of the cube, then we obtain a zero-sum distinguisher. Finally, several key recovery attacks with suitable cubes are implemented in this section. We implement the attacks on the Linux system with 8 GB RAM and use Cyptominisat solver with Python interface to solve SAT problems.

### A. APPLICATIONS TO MORUS-640-128

In this section, we practically evaluate the performance of the cube attack against reduced versions of MORUS-640-128. There are 5 rounds with similar operations at each step in the state update function, so one of five rounds will be called 0.2-step in this paper. Besides, the keystream generation function is named 0.5-step.

**Algorithm 5** Evaluate Secret Variables by SAT Method

```
Declare cube indexes I
Let x be n SAT variables of M corresponding to secret variables.
Let v be m SAT variables of M corresponding to public variables.
v_i = 1 for all i ∈ I.
v_i = 0 for all i ∈ {0, 1, ··· , m − 1} − I.
x_i = 0 for all i ∈ {0, 1, ··· , n − 1}.
for j in range(m) do
    x_j = 1
    solve SAT model M
    if M is satisfied then
        J = J ∪ {j}
    end if
    x_j = 0
end for
return J
```

To conduct the practical attack, we use an SAT-based algorithm, which is a division property based cube attack framework. According to Proposition 1, for a polynomial $f(x, y)$, if there is no division trail such that $(e_j, k_I) \xrightarrow{f} 1$, then the observed bit $x_j$ does not belong to the set $J$, whose elements are indexes of secret variables may be involved to the superpoly. Consequently, we can evaluate which secret variables may be involved in the superpoly of a given cube based on this algorithm. The specific progress is shown in Algorithm 5. And this algorithm is supported by the SAT solver. The input $\mathcal{M}$ is an SAT model, where the target stream cipher is represented by using the division property.

During the attack, we implement the SAT model for the propagation of the division property on MORUS-640-128 for the first step. Then we evaluate involved secret variables by using Algorithm 5. To verify our attack and implementation, cubes identical to [34] are directly used in our testing experiments for 2.5-step to 5.5-step, and the results we obtained are consistent with the previous results. Then, we find some better cubes of the same size as [14], as shown in Table 2, and with these cubes we gain one more step key recovery attack on MORUS-640-128 in the initialization phase.

**TABLE 2.** Example of cubes for 5-step MORUS-640-128 with 3-dimensional cubes.

| Cube Indexes | Output Index | Involved secret bits $J$ |
|---|---|---|
| 4 93 115 | $z_0$ | 79 |
| 84 92 121 | $z_0$ | 109 |
| 53 89 123 | $z_2$ | 15 |
| 66 90 118 | $z_6$ | 60 |
| 35 58 71 | $z_7$ | 31 |
| 23 70 109 | $z_9$ | 113 |
| 29 82 100 | $z_0$ | 46 91 |
| 60 66 113 | $z_5$ | 11 37 |
| 58 94 101 | $z_7$ | 20 24 |
| 10 44 113 | $z_{13}$ | 6 61 |

**TABLE 3.** Some of cube attacks on MORUS-640-128 with various cubes.

| Steps | Cube Indexes | Output bit | Involved secret variables $J$ |
|---|---|---|---|
| 5.9 | 1 11 14 19 31 35 37 40 50 51 56 60 63 67 76 82 90 95 107 112 115 116 117 126 | $z_2$ | 2 8 10 16 17 20 24 25 28 29 30 39 43 49 75 90 93 96 99 100 104 113 119 121 |
| 5.9 | 3 5 12 23 30 41 46 52 53 70 71 74 77 79 83 85 86 93 94 96 101 107 118 125 | $z_4$ | 6 8 17 24 31 34 39 44 45 50 56 61 70 78 80 87 88 91 92 96 97 104 107 127 |
| 5.9 | 1 9 18 27 28 37 44 47 50 54 61 70 76 78 79 82 87 98 100 101 104 108 113 118 | $z_7$ | 6 9 14 16 23 24 26 27 33 38 44 45 49 57 64 68 71 88 92 93 96 99 106 118 119 127 |
| 6 | 3 16 23 40 43 45 46 47 56 60 67 68 80 87 91 92 93 97 113 116 117 126 | $z_2$ | 5 7 15 21 30 32 36 48 64 70 72 75 89 109 110 122 |
| 6 | 2 6 8 13 20 22 26 35 38 42 49 50 52 70 76 80 94 95 96 107 121 127 | $z_2$ | 3 4 9 17 18 24 25 32 40 51 56 60 66 89 105 112 114 119 |

Further, we find better cubes with the greedy algorithm and select different output keystream bits for 5.7-step and 5.9-step. Moreover, we find a cube of a practical size 24, which can lead to the first 5.9-step key recovery attack, and we gain 0.4-step (two rounds) more than the previous best work in [34]. For example, if the cube index is $I = \{1, 11, 14, 19, 31, 35, 37, 40, 50, 51, 56, 60, 63, 67, 76, 82, 90, 95, 107, 112, 115, 116, 117, 126\}$, then the key variables with index in $\{2, 8, 10, 16, 17, 20, 24, 25, 28, 29, 30, 39, 43, 49, 75, 90, 93, 96, 99, 100, 104, 113, 119, 121\}$ may be involved in the corresponding superpoly, and the complexity of recovering the superpoly is $2^{32.06}$. Also, we find some good cubes of size 22 for 6-step MORUS-640-128 in the initialization phase, which improves two more steps comparing with [14]. We provide some selected cubes and summarize the corresponding key recovery attack results in Table 3.

### B. APPLICATIONS TO TRIAD

The security of TRIAD in the initialization process is evaluated using the same approach as before.

Firstly, we conduct the algebraic degree estimation algorithm to calculate the lower bound of the largest distinguishable round for TRIAD. To compare the performance

**TABLE 4.** Some cubes of TRIAD found by our greedy algorithm.

| Cube Size | #Rounds | Cube Indexes |
|---|---|---|
| 20 | 523 | 0 1 2 5 6 7 8 9 10 11 12 13 14 24 25 26 27 28 29 30 |
| 25 | 555 | 0 1 2 5 6 7 8 9 10 11 12 13 14 21 23 24 25 26 27 28 29 30 37 56 72 |
| 30 | 561 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 17 21 22 23 24 25 26 27 28 29 30 37 46 56 72 |
| 45 | 568 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 37 41 44 46 47 56 59 63 67 72 73 74 75 79 |

between the accelerated greedy algorithm and our new greedy algorithm, we explore cube testers using the accelerated greedy algorithm and our new greedy algorithm, respectively. During the experiments with the accelerated greedy algorithm, we exhaust all possible cubes of size 4 at the first iteration. After that, we add 4 to 6 new bits at each iteration. As for our greedy algorithm, we exhaust all possible cubes of size 92 at the first iteration and drop 4 to 6 bits at each iteration. Experiments show that both methods can find the same good cube tester efficiently. For example, we find the same cube tester with size of 30, and the cube index set is $I = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 17, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 37, 46, 56, 72)$, which can lead to a 561-round zero-sum distinguisher. We also obtain the 568-round distinguisher with the cube of size 45. We summarize the results in Table 4.

Therefore, to reduce computational complexity, our new algorithm is recommended when the size of the cube is larger than half the length of IV; the accelerated greedy algorithm is recommended when the size of the cube is smaller than half the length of IV. Moreover, we construct the SAT model of division property for TRIAD. In the initial input of the division property, the division properties of constant bits are fixed to 0. Set the division properties of cube IV bits to 1, and non-cube IVs to 0. In our experiments, we set the non-cube IV bits to be zeros, so let the flags of non-cube IV bits equal to $0_c$.

We obtain a cube of size 16, which leads to a key recovery attack for 521-round TRIAD. The cube index set is $I = \{0, 1, 5, 6, 7, 8, 9, 10, 11, 24, 25, 26, 27, 28, 29, 30\}$, only 12 secret key bits with index in {39,43,47,48, 51, 57, 58, 67, 72, 119, 121, 122} may exist in the superpoly corresponding to this cube. The complexity of recovering the superpoly is $2^{24.81}$. With the cube of size 30 and the cube index in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 17, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 37, 46, 56, 72\}$, we provide a key recovery attack for its 565-round variant. The key bits with index in {6, 14, 15, 18, 40, 41, 43, 51,53,54, 55,56,60,72,74, 118, 120, 124} may be involved in the corresponding superpoly. For 572-round TRIAD, then the following secrect bits with index in $J = \{0, 1, 2, \cdots, 127\} \setminus \{2, 3, 6, 32, 33, 34, 36, 39, \cdots, 42, 48, 49, 62, 81, 100, 101, 104, 105, 106, 109, \cdots, 127\}$ may be involved in the corresponding superpoly of the output key.

**TABLE 5.** Some of cube attacks on TriviA-ck-v2 with various cubes.

| Rounds | Cube Indexes | Involved secret variables $J$ |
|---|---|---|
| 804 | 5 11 20 26 35 66 71 86 | 24 27 36 42 45 61 62 63 66 91 92 93 106 107 108 125 126 127 |
| 845 | 2 4 10 25 31 40 46 55 121 | 7 18 28 29 35 37 50 51 52 54 61 71 72 73 92 93 94 101 102 103 |
| 857 | 5 6 11 20 26 35 36 41 50 66 71 86 | 0 4 13 38 39 40 43 44 45 49 54 55 59 60 61 64 65 66 70 73 119 120 121 |
| 867 | 5 11 20 22 26 35 37 41 50 66 71 86 | 13 32 33 34 43 49 53 54 55 64 73 |
| 874 | 1 3 5 6 11 18 20 22 26 35 36 41 50 66 71 86 | 13 19 20 21 30 51 60 73 115 116 117 |

### C. APPLICATIONS TO TRIVIA-CK-V2

We conduct experiments searching for cubes on the reduced version of TriviA-ck-v2 to find the existence of lower dimension cubes.

We initialize the algebraic degrees of cube variables to 1, and set the degrees of any non-zero constant bits are 0. Since secret key bits are constant with respect to cube variables, the degrees of secret key bits are set to 0, too. We set non-cube IV bits to constant 0, so the degrees of both non-cube IV bits and other constant 0 bits are set to $-\infty$. When evaluating the degree of $A_{130}A_{131}$, $B_{103}B_{104}$, and $C_{145}C_{146}$, we make further improvement of the degree evaluation by utilizing the numeric mapping technique in [33]. We can make the algebraic degree estimation tending towards real value due to its NFSR-based structure.

We apply the model described in Section 4, and we obtain cubes resulting in key recovery attacks up to 874 rounds of TriviA-ck-v2, as shown in Table 5. For TriviA-ck-v2, an 874-round key recovery attack can be performed with 16 cube variables. If the cube index set is $I = \{1, 3, 5, 6, 11, 18, 20, 22, 26, 35, 36, 41, 50, 66, 71, 86\}$, then the secret key bits with index set in {13, 19, 20, 21, 30, 51, 60, 73, 115, 116, 117} may be involved in the corresponding superpoly of the output key bit, and the complexity of superpoly recovery is $2^{22.48}$.

### D. APPLICATIONS TO ENHANCED-BIVIUM

For Enhanced-bivium, we obtain some better cryptanalytic results at a lower computational complexity. Combining the method to select cube variables for key recovery attack, we can find some good cubes, shown in Table 6.

We obtain a cube with a size 68, and the number of the round we can attack theoretically is 575. The cube index set is $I = \{0, 1, 2, \cdots, 79\} \setminus \{0, 1, 2, 31, 42, 44, 46, 54, 56, 58, 66, 75\}$. Also, we find a cube of size 35, and we can get a 559-round practical zero-sum distinguisher. We also propose a key recovery attack by using cube attack based on division property with the SAT method on reduced Enhanced-bivium. For 502-round Enhanced-bivium in initialization phase, if the cube index set of size 12 is $I = \{3, 5, 9, 12, 15, 18, 20, 25, 48, 51, 53, 69\}$, the secret key bits with index in {7, 21, 23, 24, 25, 30, 46, 47, 48, 71, 72, 73} may be involved in the corresponding superpoly of output

**TABLE 6.** Some cubes of Enhanced-bivium found by our greedy algorithm.

| Cube Size | #Rounds | Cube Indexes |
|---|---|---|
| 20 | 535 | 3 5 7 9 12 15 18 20 23 25 27 29 31 33 36 38 48 51 53 69 |
| 25 | 547 | 0 1 3 5 6 7 9 12 15 18 20 23 25 27 29 31 33 36 38 48 51 53 56 58 69 |
| 35 | 559 | 0 1 3 4 5 6 7 8 9 10 11 12 13 15 18 20 23 25 27 29 31 33 36 38 42 44 46 48 51 53 56 58 60 62 69 |
| 46 | 562 | 8 9 10 11 12 14 15 16 17 20 21 22 23 24 26 27 29 30 32 33 34 36 38 39 41 43 45 47 48 50 51 53 55 57 59 61 63 64 65 67 69 71 72 74 77 79 |
| 68 | 575 | 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 32 33 34 35 36 37 38 39 40 41 43 45 47 48 49 50 51 52 53 55 57 59 60 61 62 63 64 65 67 68 69 70 71 72 73 74 76 77 78 79 |

**TABLE 7.** Some of cube attacks on Quartet with various cubes.

| Steps | Cube Indexes | Involved secret variables $J$ |
|---|---|---|
| 4 | 65 71 74 81 82 94 97 104 110 111 116 118 158 160 163 164 188 191 | 38 39 43 44 45 205 208 230 |
| 4 | 66 80 87 98 104 107 109 110 119 120 121 123 125 154 158 163 167 176 | 8 13 14 38 44 48 49 226 229 251 |
| 4 | 68 74 85 86 87 99 110 111 115 116 126 153 158 166 181 184 186 191 | 1 2 9 14 15 19 24 25 60 194 217 220 233 236 242 |

key bit. The time complexity of recovering the superpoly is $2^{18.19}$. So far, all of the attacks on Enhanced-bivium are the best attacks in terms of the number of attacked rounds and the feasibility of attack.

### *E. APPLICATIONS TO QUARTET*

We use the technique illustrated in Section 4 on the modified version of Quartet to find good cubes. We take the first keystream bit as the output bit. The appropriate cubes found for the 3-step and 4-step Quartet are of size 2 and 18, respectively.

We denote the 256-bit state of Quartet by $s$, where $s[i]$ represents the $i$-th bit of $s$. For 3-step Quartet in the initialization process, if the cube index set is $I = \{180, 181\}$, then the secret key bits with index set in $\{201, 240, 243\}$ may be involved in the corresponding superpoly of the output bit $z_0$, and the complexity of superpoly recovery is $2^4$. We do similar experiments on Quartet with an initialization phase of 4 steps, the attack on 4-step Quartet sequentially utilizes several 18-dimensional cubes, as shown in Table 7. For example, if the cube index set is $I = \{65, 71, 74, 81, 82, 94, 97, 104, 110, 111, 116, 118, 158, 160, 163, 164, 188, 191\}$, then the secret key variables with index in $\{38, 39, 43, 44, 45, 205, 208, 230\}$ may be involved in the corresponding superpoly of the output key bit $z_0$, and the complexity to recover the superpoly is $2^{23.81}$. However, the degree of the output function grows significantly with the increase in the number of steps. Our further experiments reveal that cube attacks failed when the number of attack step is increased from 4 to 5.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we present a new method for finding good cubes for cryptographic primitives with optimal complexity. In our division property based cube attacks, the operations with a constant are considered, and we use the flag technique on our SAT model to propagate the division property more precisely. We give some new or improved cube cryptanalysis on MORUS-640-128, TRIAD, Quartet, TriviA-ck-v2, and Enhanced-bivium. As an important type of cryptanalysis, the cube attack is flexible in applying in the IoT since the attacking rounds are improved.

In the future, we will study how to estimate the algebraic degree of non-NFSR based cryptosystems more precisely.

## REFERENCES

[1] (2013). *CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness*. [Online]. Available: http://competitions.cr.yp.to/caesar.html

[2] (2018). *NIST Lightweight Cryptography Standardization Process*. [Online]. Available: https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates

[3] H. Wu and T. Huang. (2016). *The Authenticated Cipher MORUS (v2)*. CAESAR Additional Finalists. [Online]. Available: http://competitions.cr.yp.to/round3/morusv2.pdf

[4] A. Chakraborti and M. Nandi. (2015). *TriviA-ck-v2*. CAESAR Round 2 Candidates. [Online]. Available: http://competitions.cr.yp.to/round2/triviackv2.pdf

[5] B. Subhadeep, I. Takanori, M. Willi, T. Yosuke, and B. Zhang. (2019). *Triad v1-A Lightweight AEAD and Hash Function Based on Stream Cipher*. NIST LWC Round 1 Submission. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TRIAD-spec.pdf

[6] B. Zhang. (2019). *Quartet: A Lightweight Authenticated Cipher (v1)*. NIST LWC Round 1 Submission. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/Quartet-spec.pdf

[7] I. Dinur and A. Shamir, "Cube attacks on tweakable black box polynomials," in *Proc. EUROCRYPT*, Berlin, Germany, vol. 5479, Apr. 2009, pp. 278–299.

[8] X. Lai, "Higher order derivatives and differential cryptanalysis," in *Communications and Cryptography*, vol. 276. Boston, MA, USA: Springer, 1994, pp. 227–233.

[9] H. Englund, T. Johansson, and M. S. Turan, "A framework for chosen IV statistical analysis of stream ciphers," in *Proc. INDOCRYPT*, Heidelberg, Germany, vol. 4859, Dec. 2007, pp. 268–281.

[10] S. Fischer, S. Khazaei, and W. Meier, "Chosen IV statistical analysis for key recovery attacks on stream ciphers," in *Proc. AFRICACRYPT*, Berlin, Germany, vol. 5023, Jun. 2008, pp. 236–245.

[11] J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir, "Cube testers and key recovery attacks on reduced-round MD6 and Trivium," in *Proc. FSE*, Berlin, Germany, Feb. 2009, pp. 1–22.

[12] I. Dinur and A. Shamir, "Breaking grain-128 with dynamic cube attacks," in *Proc. FSE*, Berlin, Germany, vol. 6733, 2011, pp. 167–187.

[13] P.-A. Fouque and T. Vannet, "Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks," in *Proc. FSE*, Heidelberg, Germany, vol. 8424, Mar. 2014, pp. 502–517.

[14] I. Salam, L. Simpson, H. Bartlett, E. Dawson, and K. K. Wong, "Investigating cube attacks on the authenticated encryption stream cipher MORUS," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 961–966.

[15] M. Liu, J. Yang, W. Wang, and D. Lin, "Correlation cube attacks: From weak-key distinguisher to key recovery," in *Proc. EUROCRYPT*, Cham, Switzerland, vol. 10821, May. 2018, pp. 715–744.

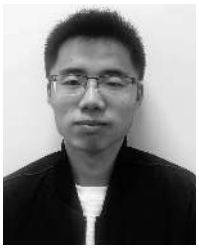
[16] I. Dinur, P. Morawiecki, J. Pieprzyk, M. Srebrny, and M. Straus, "Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function," in *Proc. EUROCRYPT*, vol. 9056, Berlin, Germany, Apr. 2015, pp. 733–761.

[17] S. Huang, X. Wang, G. Xu, M. Wang, and J. Zhao, "Conditional cube attack on reduced-round Keccak sponge function," in *Proc. EUROCRYPT*, Cham, Switzerland, vol. 10211, 2017, pp. 259–288.

[18] Z. Li, W. Bi, X. Dong, and X. Wang, "Improved conditional cube attacks on Keccak keyed modes with MILP method," in *Proc. ASIACRYPT*, Cham, Switzerland, vol. 10624, Dec. 2017, pp. 99–127.
[19] Z. Li, X. Dong, and X. Wang, "Conditional cube attack on round-reduced ASCON," *IACR Trans. Symmetric Cryptol.*, vol. 2017, pp. 175–202, Mar. 2017.
[20] X. Dong, Z. Li, X. Wang, and L. Qin, "Cube-like attack on round-reduced initialization of Ketje Sr," *IACR Trans. Symmetric Cryptol.*, vol. 2017, pp. 259–280, Mar. 2017.
[21] P. Mroczkowski and J. Szmidt, "The cube attack on stream cipher Trivium and quadraticity tests," *Fundamenta Informaticae*, vol. 114, nos. 3–4, pp. 309–318, 2012.
[22] C. Ye and T. Tian, "An algebraic method to recover superpolies in cube attacks," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 1082, 2018.
[23] Y. Todo, "Structural evaluation by generalized integral property," in *Proc. EUROCRYPT*, Berlin, Germany, vol. 9056, Apr. 2015, pp. 287–314.
[24] Y. Todo and M. Morii, "Bit-based division property and application to simon family," in *Proc. FSE*, Berlin, Germany, vol. 9783, Mar. 2016, pp. 357–377.
[25] Z. Xiang, W. Zhang, Z. Bao, and D. Lin, "Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers," in *Proc. ASIACRYPT*, Berlin, Germany, vol. 10031, Dec. 2016, pp. 648–678.
[26] L. Sun, W. Wang, and M. Wang, "Automatic search of bit-based division property for ARX ciphers and word-based division property," in *Proc. ASIACRYPT*, Cham, Switzerland, vol. 10624, Dec. 2017, pp. 128–157.
[27] Y. Todo, T. Isobe, Y. Hao, and W. Meier, "Cube attacks on non-blackbox polynomials based on division property," in *Proc. CRYPTO*, vol. 10403, 2017, pp. 250–279.
[28] Q. Wang, Y. Hao, Y. Todo, C. Li, T. Isobe, and W. Meier, "Improved division property based cube attacks exploiting algebraic properties of superpoly," in *Proc. CRYPTO*, Aug. 2018, pp. 275–305.
[29] S. Wang, B. Hu, J. Guan, K. Zhang, and T. Shi, "A practical method to recover exact superpoly in cube attack," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 259, 2019.
[30] J. Yang, M. Liu, and D. Lin, "Cube cryptanalysis of round-reduced ACORN," in *Proc. Int. Conf. Inf. Secur.*, Cham, Switzerland, Sep. 2019, pp. 44–64.
[31] L. Karlsson, M. Hell, and P. Stankovski, "Improved greedy nonrandomness detectors for stream ciphers," in *Proc. ICISSP*, vol. 2017, 2017, pp. 225–232.
[32] P. Stankovski, "Greedy distinguishers and nonrandomness detectors," in *Proc. INDOCRYPT*. Berlin, Germany: Springer, Dec. 2010, pp. 210–226.
[33] M. Liu, "Degree evaluation of NFSR-based cryptosystems," in *Proc. CRYPTO*, Cham, Switzerland, Aug. 2017, pp. 227–249.
[34] Y. Li and M. Wang, "Cryptanalysis of MORUS," *Des., Codes, Cryptogr.*, vol. 87, pp. 1035–1058, May. 2019.
[35] S. Sarkar, S. Maitra, and A. Baksi, "Observing biases in the state: Case studies with Trivium and Trivia-SC," *Des., Codes Cryptogr.*, vol. 82, nos. 1–2, pp. 351–375, 2017.
[36] S. Zhang, G. Chen, and J. Li, "Cube attack on reduced-round enhanced-bivium," in *Proc. SSIC*, Paris, France, Jul. 2016.
[37] L. Sun, W. Wang, and M. Wang, "MILP-aided bit-based division property for primitives with non-bit-permutation linear layers," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 811, 2016.
[38] S. Zhang, G. Chen, Y. Zhou, and J. Li, "Enhanced-bivium algorithm for RFID system," *Math. Problems Eng.*, vol. 2015, pp. 1–6, Oct. 2015.

**YU HE** received the bachelor's degree in computer science and technology from Henan Agricultural University, in 2018. She is currently pursuing the master's degree with East China Normal University, Shanghai, China. Her research interests include symmetric cryptography, cube attack, and network security.

**GAOLI WANG** received the B.S. degree in fundamental mathematics and the Ph.D. degree in information security from Shandong University, Jinan, China. She is currently a Professor with the Software Engineering Institute, East China Normal University. Her research interests include cryptography, computer, and network security.



**WENSHAN LI** received the bachelor's degree in software engineering from Yantai University, in 2017. He is currently pursuing the master's degree with East China Normal University, Shanghai, China. His research interests include division property, cube attack, stream ciphers, and network security.

**YIZHI REN** received the Ph.D. degree in computer software and theory from the Dalian University of Technology, China, in 2011. From 2008 to 2010, he was a Research Fellow with Kyushu University, Japan. He is currently an Associate Professor with the School of Cyberspace, Hangzhou Dianzi University, China. He has published more than 60 research articles in refereed journals and conferences. His current research interests include network security, complex networks, and trust management. He won the IEEE Trustcom2018 Best Paper Award, the CSS2009 Student Paper Award, and the AINA2011 Best Student Paper Award.