A New General Method of Searching for Cubes in Cube Attacks

Lin Ding[1,2(✉)], Lei Wang[2,3], Dawu Gu[2], Chenhui Jin[1], and Jie Guan[1]

[1] PLA SSF Information Engineering University, Zhengzhou 450001, China
dinglin_cipher@163.com
[2] Shanghai Jiao Tong University, Shanghai 200240, China
[3] Westone Cryptologic Research Center, Beijing 100000, China

Abstract. Cube attack, proposed by Dinur and Shamir at EUROCRYPT 2009, is one of general and powerful cryptanalytic techniques against symmetric-key cryptosystems. However, it is quite time consuming to search for large cubes using the existing techniques, e.g., random walk, and practically infeasible to execute the cube attack when the size of cube exceeds an experimental range, e.g., 50. Thus, how to find favorite cubes is still an intractable problem. In this paper, a new general method of searching for cubes in cube attacks, called *iterative walk*, is proposed. Iterative walk takes the technique numeric mapping proposed at CRYPTO 2017 as a tool, which is used to test cubes and find out the best cubes among them. This new method consists of two concrete techniques, called *incremental iterative walk* and *decremental iterative walk*, respectively. Both of them split the process of searching for cubes with large size into several iterative processes, each of which aims at searching for a 'best' set of input variables with small size. After each iterative process, the input variables in the obtained 'best' set are added to (or dropped from) the cube in incremental (or decremental) iterative walk. As illustrations, we apply it to the authenticated encryption cipher ACORN v3, which was selected as one of seven finalists of CAESAR competition. Some new distinguishing attacks on round reduced variants of ACORN v3 are obtained.

Keywords: Cube attack · Distinguishing attack · ACORN v3 · Numeric mapping

1 Introduction

Cube attack on tweakable black box polynomials was introduced by Dinur and Shamir [1] at EUROCRYPT 2009 and can be seen as a generalization of higher-order differential attack [2,3] and chosen IV statistical attacks [4,5]. The idea of cube attack is to tweak the multivariate master polynomial by assigning chosen values for the public variables, which results in derived polynomials. The set of assigned public variables is denoted as a *cube*, and the sum of corresponding derived polynomials over all values of the cube, denoted as a *superpoly*, is evaluated. The target of cube attacks is to find a number of linear superpolys in terms

W. Meng et al. (Eds.): ICICS 2020, LNCS 12282, pp. 369–385, 2020.
https://doi.org/10.1007/978-3-030-61078-4_21

of the common secret variables and recover the secret variables by solving the resultant system of linear equations. The possibility that a cube yields a linear equation depends on both its size and the algebraic properties of the cipher. Since the seminal work of Dinur and Shamir, several variants of cube attacks, including cube tester [6], dynamic cube attack [7], conditional cube attack [8] and correlation cube attack [9] were put forward.

Previous Works on Searching for Cubes. A key step to a successful cube attack is searching for good cubes and the corresponding superpolys during the offline phase. However, how to find favorite cubes is still an intractable problem. In the original paper of the cube attack [1], the cryptosystems were regarded as black-box, and the authors proposed a new technique, which is a variant of the *random walk* proposed in [5], to search for cubes experimentally. The basic idea is to start from a random subset and iteratively test the linearity of superpoly to decide whether the size of tested subset should be increased or decreased. In this technique, the authors introduced a linearity test to reveal the structure of the superpoly. If the linearity test always passes, the Algebraic Normal Form (ANF) of the superpoly is recovered by assuming that the superpoly is linear. Moreover, a quadraticity test was introduced in [10], and the ANF of the superpoly is similarly recovered. Note that they are experimental cryptanalysis, and it is possible that cube attacks do not actually work. For example, if the superpoly is highly unbalanced function for specific variables, we cannot ignore the probability that the linearity and quadraticity tests fail.

In [11], an simple evolutionary algorithm was proposed by Aumasson et al. to find good cubes. By introducing the well known greedy heuristic, a strategy called Greedy Bit Set Algorithm was presented by Stankovski in [12] to find cubes. The authors of [13] and [14] both used the union of two subcubes to generate larger cube candidates. In all these works, the size of a cube is limited to the experimental range because the attacker has to make 2^d encryptions under the fixed key to compute the sum over a cube of size d. Thus, searching for large cubes is time consuming, and it is practically infeasible to execute the cube attack when the size of cube exceeds an experimental range, e.g., 50. This restricts the capability of the attacker for better cubes.

Numeric Mapping. Recently two works on cube attacks using large cubes of size greater than 50 were presented in [15,16]. Both of them treat the cryptosystems as non-blackbox polynomials. One is introducing the bit-based division property into cube attacks on non-blackbox polynomials by Todo et al. [15] at CRYPTO 2017. More recently, Wang et al. [17,18] further investigated this attack and presented better key recovery attacks on some NFSR-based stream ciphers. Nevertheless, in these two works, the recovered secret variables are generally smaller than 1 bit, while the time complexities are significantly high and the success probabilities of key recovery are difficult to estimate as their attacks are based on some assumptions. Another is exploiting a new technique, called *numeric mapping*, to present a general framework of iterative estimation of algebraic degree for NFSR-based cryptosystems by Liu [16] at CRYPTO 2017. The key idea of Liu's work is based on a simple fact. Its advantage is that it has linear time complexity and needs a negligible amount of memory. Furthermore,

it is deterministic rather than statistical. As pointed out by Todo et al. [19], Liu's method is more efficient, since cube attacks based on division property need to ask for the help of solvers, e.g., the MILP solver. The high efficiency of numeric mapping makes it possible to test a large number of large cubes with limited computational resources. It is important to note that numeric mapping can give an upper bound on algebraic degree of the output of a given NFSR-based cryptosystem when the cube is given. However, how to search for cubes using numeric mapping is not explored in [16]. Later, Zhang et al. [20] further investigated Liu's work, and presented some attacks on two variants of Trivium stream cipher.

Previous Attacks on ACORN v3. ACORN v3 [21] is an authenticated encryption stream cipher, and was selected as one of seven finalists of CAESAR competition [22] at March 2018. Up to now, several attacks on ACORN v3 had been published in [23–26]. However, there are no attacks better than exhaustive key search on ACORN v3 so far. In [27], Ghafari and Hu proposed a new attack framework based on cube testers and d-monomial test, and gave a distinguishing attack on 676 initialization rounds of ACORN v3 with a time complexity of 200×2^{33}.[1] In [29], Ding et al. proposed distinguishing attacks on 647, 649, 670, 704, and 721 initialization rounds of ACORN v3, which is the best known distinguishing attack on the round reduced variants of ACORN v3 so far. At CRYPTO 2017, Todo et al. [15] proposed possible key recovery attacks on 647, 649 and 704 rounds of ACORN v3, where no more than one bit of the secret key can be recovered with unknown probability in around 2^{78}, 2^{109} and 2^{122}, respectively. The attack was improved by Wang et al. [17] at CRYPTO 2018, and possible key recovery attacks on 704 and 750 rounds of ACORN v3 are presented, where no more than one bit of the secret key can be recovered with unknown probability in around $2^{77.88}$ and $2^{120.92}$, respectively. Recently, two works [30,31] on constructing distinguishers on ACORN v3 had been published, which were done independently of our results.

Our Contribution. In this paper, a new general method of searching for cubes in cube attacks, called *iterative walk*, is proposed. Iterative walk takes the technique numeric mapping as a tool, which is used to test cubes and find out the best cubes among them. It consists of two concrete techniques, called *incremental iterative walk* and *decremental iterative walk*, respectively. Both of these two techniques split the process of searching for cubes with large size into several iterative processes, each of which aims at searching for a 'best' set of input variables with small size. After each iterative process, the input variables in the obtained 'best' set are added to (or dropped from) the cube in incremental (or decremental) iterative walk. As illustrations, we apply it to ACORN v3. Some new distinguishing attacks on round reduced variants of ACORN v3 we have obtained are listed in Table 1, and comparisons with previous works are made. Note that three key recovery attacks on the cipher in [16–18] are also listed in Table 1. In these attacks, the recovered secret variables are generally no more

[1] Only 670 initialization rounds of ACORN v3 was attacked when it was formally published in [28].

than 1 bit, while the time complexities are significantly high. Because of the high time complexities, these attacks are impractical and can not be verified by experiments, and the success probabilities of key recovery are difficult to estimate as they are based on some assumptions. Compared with them, our attacks have significantly better time complexities. Meanwhile, our attacks are deterministic rather than statistical, that is, our attacks hold with probability 1.

To verify these cryptanalytic results, we make an amount of experiments on round reduced variants of ACORN v3. The experimental results show that our distinguishing attacks are always consistent with our evaluated results. They are strong evidences of high accuracy of our method.

Table 1. Attacks on round reduced variants of ACORN v3

# Rounds	Attack	Time compleixity	Reference
647	Key recovery attack	2^{78}	[15]
	Distinguishing attack	2^{21}	[29]
	Distinguishing attack	$\mathbf{2^{18}}$	Sect. 4.2
649	Key recovery attack	2^{109}	[15]
	Distinguishing attack	2^{24}	[29]
	Distinguishing attack	$\mathbf{2^{18}}$	Sect. 4.2
676	Distinguishing attack	$200 \times 2^{33} \approx 2^{40.64}$	[27]
	Distinguishing attack	2^{36}	[29]
	Distinguishing attack	$\mathbf{2^{30}}$	Sect. 4.2
704	Key recovery attack	2^{122}	[15]
	Key recovery attack	$2^{77.88}$	[17]
	Distinguishing attack	2^{61}	[29]
	Distinguishing attack	$\mathbf{2^{50}}$	Sect. 4.2
721	Distinguishing attack	2^{95}	[29]
736	**Distinguishing attack**	$\mathbf{2^{95}}$	Sect. 4.2
750	Key recovery attack	$2^{125.71}$	[18]
750	Key recovery attack	$2^{120.92}$	[17]

This paper is organized as follows. Some preliminaries are introduced in Sect. 2. A new general method of searching for cubes in cube attacks is presented in Sect. 3. In Sect. 4, the method is applied to ACORN v3 to prove the effectiveness of our new method. The paper is concluded in Sect. 5.

2 Preliminaries

2.1 Cube Attacks and Cube Testers

Cube attack, which can be seen as a generalization of higher order differential attacks, was introduced by Dinur and Shamir [1] at EUROCRYPT 2009. It treats the output bit of a cipher as an unknown Boolean polynomial

$f(k_0, \cdots, k_{n-1}, v_0, \cdots, v_{m-1})$ where $k_0, \cdots, k_{n-1}$ are secret input variables and $v_0, \cdots, v_{m-1}$ are public input variables. Given any monomial t_I which is the product of variables in $I = \{i_1, \cdots, i_d\}$, f can be represented as the sum of terms which are supersets of I and terms which are not supersets of I:

$$f(k_0, \cdots, k_{n-1}, v_0, \cdots, v_{m-1}) = t_I \cdot p_{S(I)} + q(k_0, \cdots, k_{n-1}, v_0, \cdots, v_{m-1})$$

Where $p_{S(I)}$ is called the *superpoly* of I in f, and the set $\{v_{i_1}, \cdots, v_{i_d}\}$ is called a *cube*. The idea behind cube attacks is that the sum of the Boolean polynomial $f(k_0, \cdots, k_{n-1}, v_0, \cdots, v_{m-1})$ over the cube which contains all possible values for the cube variables is exactly $p_{S(I)}$, while this is a random function for a random polynomial. In cube attacks, low-degree superpolys in secret variables are exploited to recover the key, while cube testers work by distinguishing $p_{S(I)}$ from a random function. Especially, the superpoly $p_{S(I)}$ is equal to a zero constant, if the algebraic degree of f in the variables from I is smaller than the size of I.

2.2 Random Walk

As for cube attacks, the basic questions are how to estimate the algebraic degree of the output polynomial f which is only given as a black box, and how to choose appropriate cubes if they exist. In [1], a simple technique was proposed, which is a variant of the random walk proposed in [5]. The basic idea of this technique is briefly described as follows.

The attacker randomly chooses a size k between 1 and m and a subset I of k public variables, and computes the value of the superpoly of I by numerically summing over the cube C_I (setting each one of the other public variables to a static value, usually to zero). If his subset I is too large, the sum will be a constant value (regardless of the choice of secret variables), and in this case he has to drop one of the public variables from I and repeat the process. If his subset I is too small, the corresponding $p_{S(I)}$ is likely to be a nonlinear function in the secret variables, and in this case he has to add a public variable to I and repeat the process. The correct choice of I is the borderline between these cases, and if it does not exist the attacker can restart with a different initial I.

2.3 Numeric Mapping

In [16], Liu presented a general framework of iterative estimation of algebraic degree for NFSR-based cryptosystems, by exploiting a technique, called *numeric mapping*. Denote $\mathbb{F}_2^n$ the n-dimension vector space over $\mathbb{F}_2$. Let $\mathbb{B}_n$ be the set of all functions mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$, and let $f \in \mathbb{B}_n$. The Algebraic Normal Form (ANF) of given Boolean function f over variables $x_1, x_2, \cdots, x_n$ can be uniquely expressed as $f(x_1, x_2, \cdots, x_n) = \bigoplus_{c=(c_1,c_2,\cdots,c_n)\in\mathbb{F}_2^n} a_c \prod_{i=1}^{n} {x_i}^{c_i}$, where a_c's are coefficients of algebraic normal form of f. The numeric mapping, denoted by **DEG**, is defined as

$$\mathbf{DEG}: \quad \mathbb{B}_n \times \mathbb{Z}_n \to \mathbb{Z}_n, \\ (f, D) \mapsto \max_{a_c \neq 0} \left\{ \sum_{i=1}^{n} c_i d_i \right\}$$

where $D = (d_1, d_2, \cdots, d_n)$. For the composite function $h = f \circ G$, it defined the numeric degree of h as $\mathbf{DEG}(h, \deg(G))$, denoted $\mathbf{DEG}(h)$ for short. The algebraic degree of h is always less than or equal to the numeric degree of h. The algebraic degrees of the output bits with respect to the internal states can be estimated iteratively by using numeric mapping. Based on this technique, Liu [16] proposed a concrete and efficient algorithm (described as Algorithm 1 in Appendix for more details) to find an upper bound on the algebraic degree of the output, and then gave a general framework of iterative estimation of algebraic degree of NFSR-Based Cryptosystems.

3 Iterative Walk: A New General Method of Searching for Cubes

In Algorithm 1, an upper bound on algebraic degree of the output of a given NFSR-based cryptosystem after N initialization rounds is obtained as output. Here, we denote N_C the maximum number of rounds of not achieving maximum degree (i.e., $|C|$) when taking the variables in the set C as input variables. In this paper, we are more concerned with the value of N_C, which indicates the maximum number of rounds that efficient distinguishers can be constructed. Inspired by Algorithm 1, a new algorithm is proposed to estimate the maximum attacked number of rounds is depicted as Algorithm 2.

Algorithm 2. Estimation of the Maximum Attacked Number of Rounds

Require: Given the ANFs of the internal state $s^{(0)}$, the ANFs of the update function G and output function f, and the set of input variables C with size $|C|$.

1: Set $D^{(0)}$ and $E^{(0)}$ to $\deg\left(s^{(0)}, C\right)$;
2: Set N_C to 0;
3: For t from 1 to N do:
4: Compute $\mathbf{DegEst}\left(f, E^{(t)}\right)$;
5: If $\mathbf{DegEst}\left(f, E^{(t)}\right) < |C|$, then $N_C \leftarrow t$;
6: Compute $D^{(t)} = \mathbf{DegEst}\left(G, E^{(t-1)}\right)$;
7: Set $E^{(t)}$ to $\left(D^{(0)}, D^{(1)}, \cdots, D^{(t)}\right)$;
8: Return N_C.

In the algorithm above, $\left(s_1^{(0)}, s_2^{(0)}, \cdots, s_L^{(0)}\right)$ denotes the internal state at clock $t = 0$ with size L, and $\deg\left(s^{(0)}, C\right) = \left(\deg\left(s_1^{(0)}, C\right), \deg\left(s_2^{(0)}, C\right), \cdots, \deg\left(s_L^{(0)}, C\right)\right)$, where the notation $\deg\left(s_i^{(0)}, C\right)$ denotes the algebraic degree of $s_i^{(0)}$ with C as input variables. Especially, $\deg(0, C) = -\infty$ and $\deg(1, C) = 0$. Note that when Algorithm 2 is utilized to search for cubes, the key is taken as parameter, that is, $\deg(k_i, C) = 0$ for any bit k_i of the key. This is consistent with a distinguisher in the setting of fixed and unknown key. **DegEst** is a procedure for estimating algebraic degree. For a given NFSR-based cryptosystem, Algorithm 2 outputs the maximum number of rounds of not achieving maximum degree when taking a given cube as input variables. Similar to Algorithm 1, Algorithm 2 has linear time complexity of $\mathcal{O}(N)$ and needs a negligible amount of memory. Thanks to the high efficiency of Algorithm 2, checking a large amount of cubes with limited computational resources becomes feasible.

Based on Algorithm 2, a new general method of searching for cubes, called *iterative walk*, is proposed. Iterative walk splits the process of searching for cubes with large size into several iterative processes, each of which aims at searching for a 'best' cube of input variables with small size. After each iterative process, the cube varies according to the corresponding result. In this technique, Algorithm 2 is utilized as a tool to test given cubes and find out the best cubes among them. Iterative walk consists of two concrete techniques, called *incremental iterative walk* and *decremental iterative walk*, respectively. Different strategies are employed in these two techniques to search for cubes, as described in the following two subsections.

3.1 Incremental Iterative Walk

Incremental iterative walk splits the process of searching for cubes with large size into several iterative processes, each of which aims at searching for a 'best' cube of input variables with small size. After each iterative process, the input variables in the obtained 'best' set are added to the cube until the cube contains all input variables.

The detailed process of incremental iterative walk is summarized as follows. The attacker first sets the cube C to the empty set and N_C to 0. After that, he repeats the followings to search for a good cube with large size. He selects an iterative size r and generates q sets $\left\{\Omega_1^r, \Omega_2^r, \cdots, \Omega_q^r\right\}$ which consists of all possible sets by choosing r variables from $V - C$, where $q = \binom{|V-C|}{r}$. For each set Ω_i^r, the attacker takes the key K as parameter and the variables in $C \cup \Omega_i^r$ as input variables, sets the remaining variables in $V - (C \cup \Omega_i^r)$ to be zeros, and then computes $N_{C\cup\Omega_i^r}$ by implementing Algorithm 2. After implementing Algorithm 2 for q times, the attacker finds out the value of β which satisfies $N_{C\cup\Omega_\beta^r} = \max\left\{N_{C\cup\Omega_i^r}, i = 1, 2, \cdots, q\right\}$, sets N_C to $N_{C\cup\Omega_\beta^r}$ and C to $C \cup \Omega_\beta^r$, and then gives N_C and C as outputs in this iterative process.

Algorithm 3. Incremental Iterative Walk

Require: Given the ANFs of the internal state $s^{(0)}$, the ANFs of the update function G and output function f, and the sets of variables $K = (k_0, \cdots, k_{n-1})$ and $V = (v_0, \cdots, v_{m-1})$.

1: Set C to $\emptyset$;
2: Set N_C to 0;
3: If $C \subset V$, repeat the followings :
4: Select the iterative size r;
5: Set $\left\{\Omega_1^r, \Omega_2^r, \cdots, \Omega_q^r\right\}$ to the set of all possible sets by choosing r variables from $V - C$, where $q = \binom{|V-C|}{r}$;
6: Set two intermediate variables α and β to -1;
7: For i from 1 to q do :
8: Take the key K as parameter and the variables in $C \cup \Omega_i^r$ as input variables, set the remaining variables in $V - (C \cup \Omega_i^r)$ to be zeros, and then compute $N_{C \cup \Omega_i^r}$ by implementing Algorithm 2;
9: If $N_{C \cup \Omega_i^r} > \alpha$, then $\alpha \leftarrow N_{C \cup \Omega_i^r}$ and $\beta \leftarrow i$;
10: Set $N_C \leftarrow \alpha$ and $C \leftarrow C \cup \Omega_\beta^r$;
11: Return N_C and C.

In Algorithm 3, N_C denotes the maximum number of rounds of not achieving maximum degree $|C|$ when taking the set C as input variables. α and β are two intermediate variables and utilized to store necessary calculation results. For a given NFSR-based cryptosystem, Algorithm 3 gives the maximum number of rounds that efficient distinguishers can be constructed and the corresponding cube as outputs for each iterative process.

Complexity. Let T_0 denotes the time complexity of implementing Algorithm 2 once. Assume that the iterative processes (i.e., Step 4–11 in Algorithm 3) are executed λ times, with the corresponding iterative sizes $r_1, \cdots, r_\lambda$, respectively. In the first iterative process, Algorithm 2 is executed $\binom{|V-C|}{r_1}$ times with $C = \emptyset$, which leads to a time complexity of $T_0 \cdot \binom{m}{r_1}$. In the second iterative process, Algorithm 2 is executed $\binom{|V-C|}{r_2}$ times with $|C| = r_1$, which leads to a time complexity of $T_0 \cdot \binom{m-r_1}{r_2}$. Similarly, the time complexity of all iterative processes can be calculated easily. Thus, the total time complexity of Algorithm 3 can be obtained as

$$T = T_0 \cdot \left[\binom{m}{r_1} + \binom{m - r_1}{r_2} + \cdots + \binom{m - \sum_{i=1}^{\lambda-1} r_i}{r_\lambda} \right]$$

The time complexity of Algorithm 3 mainly depends on the time complexity of Algorithm 2 (i.e., T_0), the IV size m and the selected iterative sizes $r_1, \cdots, r_\lambda$. This algorithm needs a negligible amount of memory.

3.2 Decremental Iterative Walk

Incremental iterative walk searches for a cube with large size, by adding input variables to the cube gradually. The basic idea of decremental iterative walk is similar to incremental iterative walk, while a different strategy is employed to search for cubes in decremental iterative walk. Decremental iterative walk splits the process of searching for cubes into several iterative processes, each of which aims at searching for a 'best' cube of input variables with small size. After each iterative process, the input variables in the obtained 'best' set are dropped from the cube until the cube contains no input variables, which is different from incremental iterative walk, as depicted in Algorithm 4.

Algorithm 4. Decremental Iterative Walk

Require: Given the ANFs of the internal state $s^{(0)}$, the ANFs of the update function G and output function f, and the sets of variables $K = (k_0, \cdots, k_{n-1})$ and $V = (v_0, \cdots, v_{m-1})$.

1: Set C to V;
2: Take the key K as parameter and the variables in C as input variables, implement Algorithm 1 to compute N_C;
3: If $C \neq \emptyset$, repeat the followings :
4: Select the iterative size r;
5: Set $\left\{\Omega_1^r, \Omega_2^r, \cdots, \Omega_q^r\right\}$ to the set of all possible cube sets by choosing r variables from C, where $q = \binom{|C|}{r}$;
6: Set two intermediate variables $\alpha \leftarrow N_C$ and $\beta \leftarrow -1$;
7: For i from 1 to q do :
8: Take the key K as parameter and the variables in $C - \Omega_i^r$ as input variables, set the remaining variables in $(V - C) \cup \Omega_i^r$ to be zeros, and then compute $N_{C-\Omega_i^r}$ by implementing Algorithm 2;
9: If $N_{C-\Omega_i^r} > \alpha$, then $\alpha \leftarrow N_{C-\Omega_i^r}$ and $\beta \leftarrow i$;
10: Set $N_C \leftarrow \alpha$ and $C \leftarrow C - \Omega_\beta^r$;
11: Return N_C and C.

Similar to Algorithm 3, for a given NFSR-based cryptosystem, Algorithm 4 also gives the maximum number of rounds that efficient distinguishers can be constructed and the corresponding cube as outputs for each iterative process.

However, in Algorithm 4, the cube first contains all input variables, and then the input variables are dropped from the cube gradually. Let T_0 denotes the time complexity of implementing Algorithm 2 once. Assume that the iterative processes (i.e., Step 4–11 in Algorithm 4) are executed λ times, with the corresponding iterative sizes $r_1, \cdots, r_\lambda$, respectively. Similar to the complexity calculation of Algorithm 3, the total time complexity of Algorithm 4 can be easily given as

$$T = T_0 \cdot \left[\binom{m}{r_1} + \binom{m - r_1}{r_2} + \cdots + \binom{m - \sum_{i=1}^{\lambda-1} r_i}{r_\lambda} \right]$$

4 Application to ACORN v3

In this section, we first give a brief description of ACORN v3, and then apply our new method to ACORN v3 to exploit new distinguishing attacks on it.

4.1 A Brief Description of ACORN v3

ACORN v3 is an authenticated encryption stream cipher, and it has been selected as one of the seven algorithms in the final portfolio of the CAESAR competition. The structure of ACORN v3 is shown in Fig. 1. The state size of ACORN v3 is 293 bits, denoted by $S^{(t)} = (s_0^{(t)}, s_1^{(t)}, \cdots, s_{292}^{(t)})$ at t-th clock. It is constructed by using 6 LFSRs of different lengths 61, 46, 47, 39, 37, 59 and one additional register of length 4, and uses a 128-bit key and a 128-bit IV. ACORN v3 passes through the key-IV initialization phase, associated data processing phase, encryption/decryption phase and tag generation/verification phase. Since our work is fully based on the key-IV initialization phase, we present a brief description of the cipher during this phase. We refer to the original description of ACORN v3 in [4] for more details.

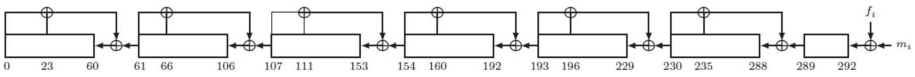

Fig. 1. The structure of authenticated encryption cipher ACORN v3

At t-th clock, the cipher executes the state update function $S^{(t+1)} = State\text{-}Update128(S^{(t)}, m_t, ca_t, cb_t)$, which is given as follows.

Step 1. Linear feedback update:

$s_{t,289} \leftarrow s_{t,289} \oplus s_{t,235} \oplus s_{t,230}$
$s_{t,230} \leftarrow s_{t,230} \oplus s_{t,196} \oplus s_{t,193}$
$s_{t,193} \leftarrow s_{t,193} \oplus s_{t,160} \oplus s_{t,154}$
$s_{t,154} \leftarrow s_{t,154} \oplus s_{t,111} \oplus s_{t,107}$
$s_{t,107} \leftarrow s_{t,107} \oplus s_{t,66} \oplus s_{t,61}$
$s_{t,61} \leftarrow s_{t,61} \oplus s_{t,23} \oplus s_{t,0}$

Step 2. Generate keystream bit:

$z_t \leftarrow s_{t,12} \oplus s_{t,154} \oplus s_{t,235} \cdot s_{t,61} \oplus s_{t,235} \cdot s_{t,193} \oplus s_{t,61} \cdot s_{t,193}$
$\oplus s_{t,230} \cdot s_{t,111} \oplus (s_{t,230} \oplus 1) \cdot s_{t,66}$

Step 3. Generate the nonlinear feedback bit:

$f_t \leftarrow s_{t,0} \oplus s_{t,107} \oplus 1 \oplus s_{t,244} \cdot s_{t,23} \oplus s_{t,244} \cdot s_{t,160} \oplus s_{t,23} \cdot s_{t,160}$
$\oplus ca_t \cdot s_{t,230} \oplus cb_t \cdot z_t$

Step 4. Shift the 293-bit register with the feedback bit f_t:

$s_{t+1,i} \leftarrow s_{t,i+1}$ for $i = 0, 1, \cdots, 291$
$s_{t+1,292} \leftarrow f_t \oplus m_t$

The initialization of ACORN v3 consists of loading the key and IV into the state, and running the cipher for 1792 steps.

1. Initialize the state S_{-1792} to 0.
2. Let $m_{-1792+t} = k_t$ for $t = 0$ to 127;
   Let $m_{-1792+128+t} = iv_t$ for $t = 0$ to 127;
   Let $m_{-1792+256} = k_{t \bmod 128} \oplus 1$ for $t = 0$;
   Let $m_{-1792+256+t} = k_{t \bmod 128}$ for $t = 1$ to 1535;
3. Let $ca_{-1792+t} = 1$ for $t = 0$ to 1791;
   Let $cb_{-1792+t} = 1$ for $t = 0$ to 1791;
4. For $t = -1792$ to $t = -1$, $S^{(t+1)} = StateUpdate128(S^{(t)}, m_t, ca_t, cb_t)$.

4.2 Results on ACORN v3

In this subsection, we will apply our Algorithm 3 and 4 respectively to ACORN v3 to search for cubes. A key step to apply them is choosing the iterative sizes.

The Results of Applying Algorithm. 3 to ACORN v3. When applying Algorithm 3 to ACORN v3, the chosen iterative sizes in the whole iterative process and the corresponding experimental results are listed in Table 2. In the i-th iterative process, the iterative size r_i is choosed, and then Algorithm 3 gives N_C and C as outputs, where C is obtained by adding the r_i input variables listed in the third column of Table 2 to the outputted cube in the $(i-1)$-th iterative process. N_C denotes the maximum number of rounds of not achieving maximum degree $|C|$ when taking the variables in the set C as input variables. As shown in Table 2, the best result is found in the 19-th iterative process, which results into

Table 2. The results of applying Algorithm 3 to ACORN v3

The i-th iterative process	Iterative size r_i	Added input variables	Cube size $\|C\|$	N_C
1	5	117, 121, 122, 125, 127	5	550
2	5	112, 118, 123, 124, 126	10	604
3	5	86, 91, 96, 113, 119	15	625
4	5	95, 104, 107, 116, 120	20	641
5	5	108, 109, 110, 114, 115	25	653
6	5	94, 98, 99, 100, 105	30	669
7	5	82, 87, 89, 90, 103	35	686
8	5	81, 83, 84, 101, 106	40	695
9	5	85, 88, 92, 97, 111	45	695
10	5	76, 77, 79, 93, 102	50	696
11	5	69, 70, 72, 78, 80	55	699
12	5	65, 67, 71, 74, 75	60	708
13	6	60, 61, 62, 63, 64, 73	66	710
14	6	49, 50, 56, 57, 58, 66	72	719
15	6	48, 51, 52, 53, 54, 55	78	719
16	6	42, 43, 44, 45, 46, 47	84	719
17	5	34, 35, 36, 38, 59, 68	90	723
18	6	25, 26, 29, 31, 33, 40	96	730
19	7	16, 20, 21, 22, 24, 28, 37	103	732
20	7	9, 11, 12, 18, 19, 27, 41	110	732
21	7	7, 13, 14, 15, 17, 30, 39	117	725
22	11	0, 1, 2, 3, 4, 5, 6, 8, 10, 23, 32	128	708

a distinguishing attack on 732 rounds of ACORN v3 with a time complexity of 2^{103}. All these results are obtained on a common PC with 2.5 GHz Intel Pentium 4 processor within about two days.

The Results of Applying Algorithm 4 to ACORN v3. When applying Algorithm 4 to ACORN v3, the chosen iterative sizes in the whole iterative process and the corresponding experimental results are listed in Table 3. In the i-th iterative process, the iterative size r_i is choosed, and then Algorithm 4 gives N_C and C as outputs, where C is obtained by dropping the r_i input variables listed in the third column of Table 2 from the outputted cube in the $(i-1)$-th iterative process. N_C denotes the maximum number of rounds of not achieving maximum degree $|C|$ when taking the variables in the set C as input variables. In our experiments, it should be noted that $N_V = 708$ when taking all IV bits as input variables. As shown in Table 3, the best result is found in the 1-th iterative process, which results into a distinguishing attack on 731 rounds of ACORN v3 with a time complexity of 2^{123}. All these results are obtained on a common PC with 2.5 GHz Intel Pentium 4 processor within about two days.

The Improved Results. Since the IV bits of ACORN v3 are sequentially loaded into the internal state in the second 128 initialization rounds, it is a nature and reasonable idea that we select the latter IV variables into the cube.

Table 3. The results of applying Algorithm 4 to ACORN v3

The i-th iterative process	Iterative size r_i	Dropped input variables	Cube size $\|C\|$	N_C
1	5	5, 7, 13, 14, 22	123	731
2	5	6, 15, 16, 24, 31	118	724
3	5	0, 3, 8, 17, 57	113	717
4	5	1, 12, 21, 23, 47	108	714
5	5	4, 20, 29, 42, 48	103	706
6	5	2, 10, 19, 26, 43	98	703
7	5	9, 11, 18, 27, 44	93	695
8	5	25, 28, 30, 32, 45	88	679
9	5	37, 38, 39, 40, 41	78	657
10	5	49, 50, 51, 52, 53	73	650
11	5	54, 55, 56, 58, 59	68	648
12	5	60, 61, 62, 63, 64	63	639
13	5	65, 66, 67, 68, 69	58	630
14	6	70, 71, 72, 73, 74, 75	52	608
15	6	76, 77, 78, 79, 80, 81	46	599
16	7	82, 83, 84, 85, 86, 87, 88	39	588
17	7	89, 90, 91, 92, 93, 94, 95	32	550
18	8	96, 97, 98, 99, 100, 101, 102, 103	24	532
19	9	104, 105, 106, 107, 108, 109, 110, 111, 112	15	481
20	9	113, 114, 115, 116, 117, 118, 119, 120, 121	6	376
21	6	122, 123, 124, 125, 126, 127	0	0

To reduce the search space, we fix the first p IV variables to be zeros, i.e., $iv_i = 0, i = 0, \cdots, p-1$, and put the last $q(\geq 0)$ IV variables into the cube. We consider applying Algorithm 4 when the V is dropped from $(v_0, \cdots, v_{127})$ to $(v_p, \cdots, v_{127-q})$. Some better results we have found are listed in Table 4, and the corresponding cubes are given in Appendix. As for 676 rounds of ACORN v3, the best result we have found implies $\mathbf{DEG}(f, X) = 29$, which leads to a practical distinguishing attack on it with a time complexity of 2^{30} and improves the previous distinguishing attack [29] by a factor of 2^6. As for 736 rounds of ACORN v3, the best result we have found implies $\mathbf{DEG}(f, X) = 94$, which leads to a distinguishing attack on it with a time complexity of 2^{95}. This is the best result we have found.

Experiments. Since 2^{18} and 2^{30} in Table 4 are practical, we verify these results by carrying out a test for random 100 keys within half a day on a common PC with 2.5 GHz Intel Pentium 4 processor. All outputs of 647, 649 and 676 rounds of ACORN v3 always sum to 0. This clearly confirms the effectiveness and accuracy of our method.

Table 4. The improved results on ACORN v3

# Rounds	The values of p and q	The iterative size r	Time compleixity
647	$p = 106, q = 0$	4	2^{18}
649	$p = 104, q = 0$	6	2^{18}
676	$p = 94, q = 0$	4	2^{30}
704	$p = 72, q = 0$	6	2^{50}
736	$p = 21, q = 73$	12	2^{95}

5 Conclusions

In this paper, we focus on proposing a new general method of searching for cubes in cube attacks. The new method is called *iterative walk*, which takes the technique numeric mapping as a tool. It consists of two concrete techniques, called *incremental iterative walk* and *decremental iterative walk*, respectively. Both of them split the process of searching for cubes with large size into several iterative processes, each of which aims at searching for a 'best' set of input variables with small size. After each iterative process, the input variables in the obtained 'best' set are added to (or dropped from) the cube in incremental (or decremental) iterative walk. The effectiveness and accuracy of our new method is confirmed by applying it to the authenticated encryption cipher ACORN v3. Hopefully, our new method can provide a new perspective to search for cubes in cube attacks.

Acknowledgements. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grant 61602514, 61802437, 61272488, 61202491, 61572516, 61272041, 61772547, National Cryptography Development Fund under Grant MMJJ20170125 and National Postdoctoral Program for Innovative Talents under Grant BX201700153.

Appendix A

Algorithm 1. [16] Estimation of Degree of NFSR-Based Cryptosystems

Require: Given the ANFs of the internal state $s^{(0)}$, the ANFs of the update function G and output function f, and the set of input variables X.

1: Set $D^{(0)}$ and $E^{(0)}$ to $\deg\left(s^{(0)}, X\right)$;
2: For t from 1 to N do:
3: Compute $D^{(t)} = \mathbf{DegEst}\left(G, E^{(t-1)}\right)$;
4: Set $E^{(t)}$ to $\left(D^{(0)}, D^{(1)}, \cdots, D^{(t)}\right)$;
5: Return $\mathbf{DegEst}\left(f, E^{(N)}\right)$

(See Table 5)

Table 5. The cubes used in Table 4

# Rounds	The cube size	The cube
647	18	107, ⋯, 120, 122, 123, 125, 127
649	18	104, 109, ⋯, 122, 124, 125, 127
676	30	94, ⋯, 116, 119, ⋯, 124, 127
704	50	72, 74, 75, 77, 79, ⋯, 85, 87, ⋯, 94, 97, ⋯, 127
736	95	21, 23, 24, 25, 28, 29, 30, 32, 33, 38, 39, 40, 41, 42, 46, ⋯, 51, 53, ⋯, 127

References

1. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_16
2. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Proceeding Symposium Communication and Coding Cryptography, pp. 227–233. Kluwer Academic Publishers (1994)
3. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_16
4. Englund, H., Johansson, T., Sönmez Turan, M.: A framework for chosen iv statistical analysis of stream ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77026-8_20
5. Fischer, S., Khazaei, S., Meier, W.: Chosen IV statistical analysis for key recovery attacks on stream ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68164-9_16
6. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_1
7. Dinur, I., Shamir, A.: Breaking grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_10
8. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round keccak sponge function. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 259–288. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_9

9. Liu, M., Yang, J., Wang, W., Lin, D.: Correlation cube attacks: from weak-key distinguisher to key recovery. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 715–744. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_23
10. Mroczkowski, P., Szmidt, J.: The cube attack on stream cipher trivium and quadraticity tests. Fundam. Inf. **114**(3–4), 309–318 (2012)
11. Aumasson, J., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128. Cryptology ePrint Archive, Report 2009/218 (2009). https://eprint.iacr.org/2009/218
12. Stankovski, P.: Greedy distinguishers and nonrandomness detectors. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 210–226. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17401-8_16
13. Fouque, P.-A., Vannet, T.: Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_26
14. Liu, M., Lin, D., Wang, W.: Searching cubes for testing Boolean functions and its application to Trivium. In: IEEE International Symposium on Information Theory (ISIT 2015), Hong Kong, China, 14–19 June 2015, pp. 496–500. IEEE (2015)
15. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_9
16. Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 227–249. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_8
17. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 275–305. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_10
18. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly (full version). Cryptology ePrint Archive, Report 2017/1063 (2017). https://eprint.iacr.org/2017/1063
19. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property (full version). Cryptology ePrint Archive, Report 2017/306 (2017). https://eprint.iacr.org/2017/306.pdf
20. Zhang, X., Liu, M., Lin, D.: Conditional cube searching and applications on Trivium-variant ciphers. In: Chen, L., Manulis, M., Schneider, S. (eds.) ISC 2018. LNCS, vol. 11060, pp. 151–168. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99136-8_9
21. Wu, H.: ACORN: a lightweight authenticated cipher (v3). CAESAR Submission (2016). http://competitions.cr.yp.to/round3/acornv3.pdf
22. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/index.html
23. Siddhanti, A.A., Maitra, S., Sinha, N.: Certain observations on ACORN v3 and the implications to TMDTO attacks. In: Ali, S.S., Danger, J.-L., Eisenbarth, T. (eds.) SPACE 2017. LNCS, vol. 10662, pp. 264–280. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71501-8_15

24. Zhang, X., Lin, D.: Cryptanalysis of acorn in nonce-reuse setting. In: Chen, X., Lin, D., Yung, M. (eds.) Inscrypt 2017. LNCS, vol. 10726, pp. 342–361. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75160-3_21
25. Zhang, X., Feng, X., Lin, D.: Fault attack on ACORN v3. Comput. J. **61**(8), 1166–1179 (2018)
26. Adomnicai, A., Masson, L., Fournier, J.J.A.: Practical algebraic side-channel attacks against ACORN. In: Lee, K. (ed.) ICISC 2018. LNCS, vol. 11396, pp. 325–340. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12146-4_20
27. Ghafari, V.A., Hu, H.: A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a. Cryptology ePrint Archive, Report 2017/1103 (2017). https://eprint.iacr.org/2017/1103.pdf
28. Ghafari, V.A., Hu, H.: A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a. J. Amb. Intel. Hum. Comp. **2018**, 1–8 (2018)
29. Ding, L., Wang, L., Gu, D., Jin, C., Guan, J.: Algebraic degree estimation of ACORN v3 using numeric mapping. Secur. Commun. Netw. **2019**, 1–5 (2019). https://doi.org/10.1155/2019/7429320. Article ID 7429320
30. Yang, Jingchun., Liu, Meicheng, Lin, Dongdai: Cube cryptanalysis of round-reduced ACORN. In: Lin, Zhiqiang, Papamanthou, Charalampos, Polychronakis, Michalis (eds.) ISC 2019. LNCS, vol. 11723, pp. 44–64. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30215-3_3
31. Kesarwani, A., Roy, D., Sarkar, S., Meier, W.: New cube distinguishers on NFSR-based stream ciphers. Des. Codes Cryptogr. **88**, 173–199 (2020). https://doi.org/10.1007/s10623-019-00674-1