Cube Cryptanalysis of Round-Reduced ACORN *

Jingchun Yang, Meicheng Liu, and Dongdai Lin**

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{yangjingchun, liumeicheng, ddlin}@iie.ac.cn

Abstract. The cube attack is one of the most powerful techniques in cryptanalysis of symmetric cryptographic primitives. The basic idea of cube attack is to determine the value of a polynomial in key bits by summing over a cube (a subset of public variables, *e.g.*, plaintext bits or IV bits). If the degree of the polynomial is relatively low, then we can obtain a low-degree equation in key bits, thus may contribute to reducing the complexity of key recovery.

In this paper, we use cube cryptanalysis to analyze the authenticated stream cipher ACORN (one of the 6 algorithms in the final portfolio of the CAESAR competition), and give some new results in both distinguishing attacks and key recovery attacks. Firstly, we give a new method of finding cube testers, which is based on the greedy algorithm of finding cubes, and the numeric mapping method for estimating the algebraic degree of NFSR-based cryptosystems. We apply it to ACORN, and obtain the best practical distinguishing attacks for its 690-round variant using a cube of size 38, and its 706-round variant using a cube of size 46. Then we theoretically analyze the security bound of ACORN via the division property based cube attack. By exploiting the embedded property, we find some new distinguishers for ACORN, so the zero-sum property of the output of its 775-round variant can be observed with a complexity of 2^{127}. Finally, we propose a key recovery attack on ACORN reduced to 772 rounds. The time complexity to recover the linear superpoly of the 123-dimensional cube is $2^{127.46}$. As far as we know, this is the best key recovery attack on round-reduced ACORN. It is also worth noting that this work does not threaten the security of ACORN.

Keywords: cube cryptanalysis · ACORN · distinguishing attack · key recovery · numeric mapping · division property based cube attack.

* This work was supported by the National Natural Science Foundation of China (Grant No. 61872359 and No. 61672516) and Youth Innovation Promotion Association CAS.

** Corresponding author.

1 Introduction

In modern cryptographic primitives, nonlinear feedback shift register (NFSR) plays an important role in the applications with constrained environments, like the radio-frequency identification devices (RFID) and sensor networks. Based on the structure of NFSR, many symmetric ciphers have been proposed, such as the stream ciphers Trivium [6] and `Grain` [12], the authenticated stream cipher ACORN [25], the lightweight block cipher family KATAN/KTANTAN [7], and the hash function family QUARK [3,4,5]. All these algorithms possess an efficient hardware implementation and a high level security at the same time.

Cube attack was first proposed by Dinur and Shamir [8] at EUROCRYPT 2009. This attack views a cryptosystem as a black-box polynomial f. The basic idea of this attack is that the symbolic sum of all the derived polynomials obtained from the function f by assigning all the possible values to the cube variables (a subset of public variables) is exactly the superpoly (the coefficient) of the term with maximum degree over the cube. Cube attacks work by finding a number of linear superpolys in secret variables and then recovering the secret information by solving the system of linear equations. At FSE 2009, Aumasson *et al.* [2] proposed the notion of cube testers. The main idea of cube testers is similar to cube attacks, however, it aims to find some superpolys which have the distinguishable properties (*e.g.*, the superpoly is equal to a zero constant, which is most commonly used).

For cube attacks and cube testers, the key to success is to search good cubes and their superpolys. Traditional approaches [1,9,17,13,15] of finding cube testers are experimental and very limited to the current computing power. For a given cube c, to verify the balance property of this cube, we need to compute the cubesum on this cube for a number of random keys, and the complexity of this process is $r \cdot 2^{|c|}$, where r is the amount of keys. When $|c|$ is large (*e.g.*, $|c| > 50$), it is difficult to verify the cube since the complexity is exponential to $|c|$.

However, this headache has been solved gradually in recent years. At CRYPTO 2017, Liu [14] proposed a new technique, called *numeric mapping*, to iteratively estimate the upper bound on the algebraic degree of the internal states of an NFSR-based cryptosystem. For Trivium-like ciphers, some cube testers of sizes greater than 50 were presented for the first time. Simultaneously, at CRYPTO 2017, Todo *et al.* [20] proposed the *division property based cube attack* to analyze the ANF of the superpoly. Based on the propagation of the bit-based division property [19,22] of stream ciphers, they gave a proposition to decide which key bits do not exist in the superpoly. This technique led to the possible key recovery attack of 704-round ACORN. Later at CRYPTO 2018, Wang *et al.* [24,23] proposed some new techniques (flag technique, degree evaluation, and term enumeration) to improve the division property based cube attack. In the end, they presented a new possible key recovery attack on 750-round ACORN with a cube of size 101, which is also the best known key recovery attack on round-reduced ACORN.

1.1 Our Contributions

In this paper, we evaluate the security of authenticated stream cipher ACORN [25] (one of the 6 algorithms in the final portfolio of the CAESAR competition) from the aspect of cube cryptanalysis, and propose some new distinguishing attacks and a new possible key recovery attack. All of our attacks are currently the best attacks in terms of the number of attacked rounds or the feasibility of attack.

Firstly, we give a new method of finding cube testers with a much shorter time. Our method is based on the greedy algorithm of finding cubes [17,13], and the numeric mapping method for estimating the algebraic degree of NFSR-based cryptosystems [14]. With this new method, we can efficiently search cube testers in a much larger space. We apply it to ACORN, and obtain the best practical distinguishing attacks for its 690-round variant using a cube of size 38, and its 706-round variant using a cube of size 46.

Then we theoretically analyze the security bound of ACORN via the division property based cube attack. We demonstrate that the embedded property [18] proposed at ASIACRYPT 2017 can also be applied to the search of zero-sum cube testers. Based on the embedded property for cube tester, we propose several algorithms to find the zero-sum cube testers efficiently. This leads us to find some distinguishers for ACORN, so the zero-sum property of the output of its 775-round variant can be observed with a complexity of 2^{127}.

We also propose an algorithm to find suitable cubes for key recovery attacks. The basic idea is that a cube which leads to an r-round cube tester might lead to an $(r+1)$-round cube attack as well. In the process of cube searching, we try to maximize the number of attacked arounds. Meanwhile, we keep the cube dimension and the complexity to recover the superpoly be reasonable. Finally, we find a key recovery attack on ACORN reduced to 772 rounds. The time complexity to recover the linear superpoly of the 123-dimensional cube is $2^{127.46}$. As far as we know, this is the best key recovery attack on round-reduced ACORN.

Our results for ACORN are summarized in Table 1, and the comparisions with previous attacks on ACORN are also included.

Organization. The rest of the paper is organized as follows. In Section 2 we introduce some basic definitions and theories. In Section 3 we briefly describe the ACORN cipher. Section 4 shows our new method to search good cube testers, and its applications to ACORN. In Section 5, we propose several algorithms to search zero-sum cube testers and suitable cubes for key recovery attacks in the division property based cube attack. Then we use these algorithms to analyze the security of ACORN. Section 6 concludes the paper.

Table 1. Summary of the distinguishing attacks and key recovery attacks on ACORN.

Attack types	#Rounds	Cube size	Complexity †	Ref.
distinguishing attacks	676	-	$\approx 2^{40.6}$	[10]
	690	**38**	$\mathbf{2^{38}}$	**Section 4.3**
	706	**46**	$\mathbf{2^{46}}$	**Section 4.3**
	715	**54**	$\mathbf{2^{54}}$	**Section 4.3**
	775	**127**	$\mathbf{2^{127}}$	**Section 5.3**
key recovery attacks	503	5	practical	[16]
	704	64	2^{122} ‡	[20,21]
	750	101	$2^{120.92}$ ‡	[24,23]
	772	**123**	$\mathbf{2^{127.46}}$ ‡	**Section 5.3**

† For distinguishing attacks, it refers to the data complexity. For key recovery attacks, it refers to the complexity to recover one superpoly.
‡ All these three attacks are derived by the division property based cube attack, and the real superpolys can not be obtained due to the impractical complexity, so we can recover at most 1 bit key information respectively.

2 Preliminaries

2.1 Cube Attacks and Cube Testers

Almost any cryptosystem can be described as a Boolean function $f(x_1, \cdots, x_n)$. Given a term t_I containing variables from an index subset $I \subsetneq \{1, \cdots, n\}$ that are multiplied together, the function can be written as the sum of terms which are supersets of I and terms that miss at least one variable from I,

$$f(x_1, \cdots, x_n) = p_S(I) \cdot t_I + q(x_1, \cdots, x_n),$$

where $p_S(I)$ is called the superpoly of I in f. The basic idea of cube attacks [8] is that the symbolic sum of all the derived polynomials obtained from the function f by assigning all the possible values to the subset of variables in the term t_I is exactly $p_S(I)$. Cube attacks work by finding a number of linear superpolys in secret variables and then recovering the secret information by solving the system of linear equations. While cube testers [2] work by evaluating superpolys of carefully selected terms t_I's, and trying to distinguish them 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 Numeric Mapping

Let $f(x) = \bigoplus_{u=(u_1,\cdots,u_m)\in\mathbb{F}_2^m} a_u^f \prod_{i=1}^m x_i^{u_i}$ be a Boolean function on m variables. Denote by $\mathbb{B}_m$ the set of all m-variable Boolean functions. The *numeric mapping* [14], denoted by `DEG`, is defined as

$$\texttt{DEG}: \ \mathbb{B}_m \times \mathbb{Z}^m \to \mathbb{Z},$$

$$(f, D) \mapsto \max_{a_u^f \neq 0} \{\sum_{i=1}^{m} u_i d_i\},$$

where $D = (d_1, d_2, \cdots, d_m)$ and a_u^f's are coefficients of the ANF of f. Let g_i $(1 \leq i \leq m)$ be Boolean functions on n variables, and denote $\deg(G) = (\deg(g_1), \deg(g_2), \cdots, \deg(g_m))$ for $G = (g_1, g_2, \cdots, g_m)$. The numeric degree of the composite function $h = f \circ G$ is defined as $\texttt{DEG}(f, \deg(G))$, denoted by $\texttt{DEG}(h)$ for short. We call $\texttt{DEG}(f, D)$ a super numeric degree of h if $d_i \geq \deg(g_i)$ for all $1 \leq i \leq m$, where $D = (d_1, d_2, \cdots, d_m)$. We can check that the algebraic degree of h is always less than or equal to the numeric degree of h, *i.e.*,

$$\deg(h) = \deg(f(g_1, g_2, \cdots, g_m)) \leq \texttt{DEG}(h) = \max_{a_u^f \neq 0} \{\sum_{i=1}^{m} u_i \deg(g_i)\}.$$

Based on numeric mapping, Liu [14] developed an algorithm for estimating the algebraic degree of NFSR-based cryptosystems. We refer to [14] for more details.

2.3 Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP) is an optimization or feasibility program. A MILP model $\mathcal{M}$ takes an objective function $\mathcal{M}.obj$ and a system of linear constraints $\mathcal{M}.con$ where variables $\mathcal{M}.var$ are restricted to integers. A MILP solver aims to search for an optimal solution which not only satisfies all the constraints but also minimizes/maximizes the objective function. Moreover, if there is no objective function, the MILP solver only returns whether the model is feasible or not. In this paper, we use Gurobi Optimizer [11] for our experiments.

2.4 Bit-Based Division Property and MILP Representation

The division property, proposed at EUROCRYPT 2015 [19], is a generalization of the integral property for the detection of better integral characteristics for word-oriented cryptographic primitives. Later, the bit-based division property was proposed in [22] to describe the propagation of integral characteristics more precisely. The bit-based division property is defined as follows.

Definition 1 ((Bit-Based) Division Property [22]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ be a set whose elements take an n-dimensional bit vector. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, it fulfills the following conditions:*

$$\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} = \begin{cases} \text{unknown} & \text{if there exist } \boldsymbol{k} \in \mathbb{K} \text{ s.t. } \boldsymbol{u} \succeq \boldsymbol{k}, \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{u} \succeq \boldsymbol{k}$ if $u_i \geq k_i$ for all i, and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=1}^{n} {x_i}^{u_i}$.

In [19,22], the propagation rules `copy`, `xor`, `and` are provided when the bitwise operations COPY, XOR, AND are applied to the elements in $\mathbb{X}$.

Represent the Propagation of Division Property using MILP. At ASIACRYPT 2016, Xiang *et al.* [26] first introduced a new concept *division trail* to describe the propagation of the division property, and showed that the basic propagation rules `copy`, `xor`, `and` of the division property can be translated as some variables and constraints of an MILP model. With this method, all possible division trails can be covered with an MILP model $\mathcal{M}$ and the division property of some output bit can be known according to the solutions of $\mathcal{M}$.

2.5 Bit-Based Division Property and Cube Attack

In cube attack, we want to recover the superpoly $p_S(I)$ for a cube I. Let $x_0, x_1, \ldots, x_{n-1}$ be all key bits. If the initialization is not enough for thorough diffusion, the superpoly may only be related to a part of key bits $J \subsetneq \{0, 1, \cdots, n-1\}$. At CRYPTO 2017, Todo *et al.* [20] proposed an algorithm for determining such a set J by using the bit-based division property. This algorithm is based on the following proposition.

Proposition 1 (Determining the involved key bits in the superpoly [20]). *Let $f(\boldsymbol{x}, \boldsymbol{v})$ be a polynomial, where $\boldsymbol{x}$ and $\boldsymbol{v}$ denote the secret and public variables, respectively. For a set of indices $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{0, 1, \ldots, m-1\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$ are taking all possible combinations of values. Let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $\boldsymbol{v}^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \ldots v_{i_{|I|}}$, i.e. $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Assuming there is no division trail such that $(\boldsymbol{e}_\lambda, \boldsymbol{k}_I) \xrightarrow{f} 1$, x_λ is not involved in the superpoly of the cube C_I.*

Later at CRYPTO 2018, Wang *et al.* [24,23] proposed some techniques to improve the division property based cube attack. The main contribution of [23] can be summarized as follows.

Flag Technique. In previous MILP modeling of the basic bitwise operations (COPY, XOR, AND), each intermediate state bit b is assigned a binary value $b.val$ to represent its bit-based division property value. In [23], Wang *et al.* added a 'flag' value for each state bit. The flag value $b.F$ can be $0_c, 1_c$ or δ to indicate whether the state bit is constant 0, constant 1 or variable. This change mainly affects the MILP model for AND. If the flag value $b.F$ of state bit b is 0_c, then we add a constraint $b.val = 0$, thus may improve the accuracy of MILP model description of the division property propagation.

Degree Evaluation and Term Enumeration. To recover the superpoly more efficiently, Wang *et al.* [23] proposed another two algorithms to compute the algebraic degree and enumerate all possible terms of the superpoly, respectively. The two algorithms are based on the following proposition, which is actually a generalization of Proposition 1.

Proposition 2 (Degree evaluation and term enumeration of the superpoly [23]). *Let $f(\boldsymbol{x}, \boldsymbol{v})$ be a polynomial, where $\boldsymbol{x}$ and $\boldsymbol{v}$ denote the secret and public variables, respectively. For a set of indices $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{0, 1, \ldots, m-1\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$ are taking all possible combinations of values. Let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $\boldsymbol{v}^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \ldots v_{i_{|I|}}$. Let $\boldsymbol{k}_\Lambda$ be an n-dimensional bit vector. Assuming there is no division trail such that $(\boldsymbol{k}_\Lambda || \boldsymbol{k}_I) \xrightarrow{f} 1$, the term $x^{\boldsymbol{k}_\Lambda}$ is not involved in the superpoly of the cube C_I.*

For convenience, in the rest of this paper, we denote the algorithm [23] of the degree evaluation of the superpoly, and the term enumeration of the superpoly by Algorithm `A`, and Algorithm `B` respectively.

3 A Brief Description of ACORN

ACORN [25] is an authenticated encryption stream cipher, and it has been selected as one of the 6 algorithms in the final portfolio of the CAESAR competition. ACORN has a 128-bit key and a 128-bit initialization vector. As an authenticated encryption scheme, ACORN has 4 procedures: initialization, processing the associated data, encryption, and finalization. In this paper, we only focus on the process of initialization, because the number of rounds we can attack is smaller than the 1792 initialization rounds. For more details about ACORN, we refer to [25].

Denote the internal state (at step t) of ACORN by $S_t = (s_t, s_{t+1}, \ldots, s_{t+292})$, where $t \in \{0, \ldots, 1791\}$. The initial state $S_0 = (s_0, s_1, \ldots, s_{292})$ is set to $(0, \ldots, 0)$. Denote the key and initialization vector by K and IV respectively. Let

$$m_t = \begin{cases} K_t & \text{for } t = 0 \text{ to } 127, \\ IV_{t-128} & \text{for } t = 128 \text{ to } 255, \\ K_0 \oplus 1 & \text{for } t = 256, \\ K_{t \bmod 128} & \text{for } t = 257 \text{ to } 1791. \end{cases}$$

At each step t, where $t \in \{0, \ldots, 1791\}$, the state is updated as follows.

1. update using six LFSRs.
   $s_{t+289} = s_{t+289} \oplus s_{t+235} \oplus s_{t+230}$;
   $s_{t+230} = s_{t+230} \oplus s_{t+196} \oplus s_{t+193}$;
   $s_{t+193} = s_{t+193} \oplus s_{t+160} \oplus s_{t+154}$;
   $s_{t+154} = s_{t+154} \oplus s_{t+111} \oplus s_{t+107}$;
   $s_{t+107} = s_{t+107} \oplus s_{t+66} \oplus s_{t+61}$;
   $s_{t+61} = s_{t+61} \oplus s_{t+23} \oplus s_t$;
2. generate the keystream bit.
   $ks_t = s_{t+12} \oplus s_{t+154} \oplus s_{t+235} s_{t+61} \oplus s_{t+235} s_{t+193} \oplus s_{t+61} s_{t+193} \oplus s_{t+230} s_{t+111} \oplus s_{t+230} s_{t+66} \oplus s_{t+66}$;

3. generate the nonlinear feedback bit.
   $f_t = s_t \oplus s_{t+107} \oplus 1 \oplus s_{t+244}s_{t+23} \oplus s_{t+244}s_{t+160} \oplus s_{t+23}s_{t+160} \oplus s_{t+196} \oplus ks_t$;
4. update with the feedback bit f_t.
   $s_{t+293} = f_t \oplus m_t$;

4 A New Method to Search Good Cube Testers

During the process of cube attack, searching good cubes is the most important and time-consuming part. In this section, we will give a new method to find good cube testers more efficiently. The search algorithm is inspired by the work of Stankovski [17] and Karlsson *et al.* [13]. We first introduce the greedy algorithm of finding cube testers, then we give our new method to accelerate the process of searching, finally we apply this method to ACORN.

4.1 Greedy Algorithm for Finding Cube Testers

In 2010, Stankovski [17] utilized the greedy algorithm for finding a practical (*e.g.*, the size of the bitset is no more than 40) bitset S as a cube which leads to a distinguisher or nonrandomness detector. The main procedure of this algorithm can be described as follows:

1. Choose an optimal starting bitset of a small size s_0 (which can be zero).
2. Add n bits into the bitset and select the best bitset of size $s_1 = s_0 + n$ which leads to a distinguisher or a nonrandomness detector with the largest number of rounds.
3. Repeat step 2, until a cube of expected size $s_i = s_0 + in$ is derived.

In 2017, Karlsson *et al.* [13] improved the greedy nonrandomness detectors with a more general solution. Their main idea is to extend the naive greedy algorithm by examining more possible paths. The main procedure is described as follows:

1. Consider a set of candidates from a previous iteration, or from an optimal starting set.
2. For each candidate in the list, add the k_i best bitsets (each bitset has n_i new bits) and store them in a new list. Now, we have one such new list for each candidate in the original list.
3. Merge all lists, sorting by the number of distinguishable rounds. This gives a list of $k_i \prod_{t=0}^{i-1} k_t\alpha_t$ items.
4. Finally, reduce the size of this list with the factor $\alpha_i (0 < \alpha_i \leq 1.0)$, limiting the size of the combined list to $\prod_{t=0}^{i} k_t\alpha_t$ items.
5. Repeat steps $1 \sim 4$, until a bitset of the expected size has been found.

4.2 Accelerating the Greedy Algorithm via Numeric Mapping

From the previous subsection, the greedy algorithm can surely find a good cube of a practical size, however, when the size increases, this searching

process becomes extremely time-consuming. While using the numeric mapping method [14], one can estimate the algebraic degree of the output bit of NFSR-based cryptosystems. Regardless of the size of the cube, this estimation can give an upper bound of real algebraic degree of the output bit in a linear time complexity. If the estimated degree of the output bit is smaller than the size of cube, then we obtain a zero-sum distinguisher.

We give the accelerated greedy algorithm in Algorithm 1 and Algorithm 2. When we compute the number of distinguishable rounds of a cube, we do not need to compute the cubesum on this cube and repeat this calculation on enough random keys to observe the bias of the cubesum. Instead, we use the degree estimation method to compute the lower bound of the maximum number of distinguishable rounds of this cube. The degree estimation algorithm usually costs a complexity of about $\mathcal{O}(N)$, where N is the number of initialization rounds. While the traditional approach needs a complexity of $\mathcal{O}(r \cdot 2^{|c|})$, where r is the number of random keys, and $|c|$ is the size of the cube.

Algorithm 1: Accelerated Greedy Algorithm

Input : Key K, IV V, bit space B, the number of iterations: m, vector $\boldsymbol{k}$, vector $\boldsymbol{n}$, vector $\boldsymbol{\alpha}$.
Output: Bitset S_m.

$S_0 = \{\emptyset\}$;
`/* The set` S_0 `contains a single empty bitset */`
for each $i \in \{0, \ldots, m-1\}$ **do**
  for each $c \in S_i$ **do**
    $L_c =$ **FastFindBest** (K, V, B, c, k_i, n_i);
  end for
  S_{i+1} = concatenate (all L_c from above);
  sort S_{i+1};
  reduce the number of elements in S_{i+1} by a factor α_i;
end for
return S_m;

4.3 Applications to ACORN

We apply our new method to ACORN cipher. Based on our observations on the updated functions of ACORN, we first construct a linear-time algorithm for determining the upper bound on the algebraic degree of the cipher. Then, we apply our accelerated greedy algorithm to finding good cube testers of ACORN.

The Algorithm of Degree Estimation of ACORN. We present an algorithm for ACORN to compute the upper bound of the algebraic degree of the output for a given cube. The algorithm is depicted in Algorithm 3.

Algorithm 2: FastFindBest

Input : Key K, IV V, bit space B, current bitset c, the number of best bitsets to retain: k, the number of bits to add: n.

Output: k bitsets each of size $|c| + n$.

`/* let comb(`S, k`) denote the set of all` k`-combinations of a set` S `*/`

1: $S = \emptyset$;
2: **for** each n-tuple $\{b_1, \ldots, b_n\} \in \texttt{comb}(B \setminus c, n)$ **do**
3: **using numeric mapping method, compute the lower bound z of the maximum number of distinguishable rounds for the bitset** $c \cup \{b_1, \ldots, b_n\}$;
4: **if** z is among the k highest values **then**
5: add $c \cup \{b_1, \ldots, b_n\}$ to S;
6: reduce S to k elements by removing element with lowest z;
7: **end if**
8: **end for**
9: **return** S;

In this algorithm, $d^{(t)}$ (where $t \in \{0, \ldots, N+292\}$) are global variables. Each $d^{(t)}$ gives the estimated (real when $t \leq 292$) algebraic degree of the internal state bit s_t. We first initialize the degree of s_t (where $t \in \{0, \ldots, 292\}$) by $-\infty$, since the initial state is set to $\{0, \ldots, 0\}$. At each step, we need to compute the estimated degrees of six updated bits and the keystream bit. Two subfunctions $\texttt{KSG128}(t)$ and $\texttt{FBK128}(t)$ (see Appendix A) are used to compute the estimated degree of the keystream bit and the feedback bit separately. In lines $7 \sim 14$, $d^{(t+293)}$ is updated by the feedback function $s_{t+293} = f_t \oplus m_t$. The algebraic degrees of cube variables are equal to 1. We set other non-cube IV variables to 0, so their degrees are $-\infty$. The degrees of key bits are equal to 0, since they are constants with respect to cube variables in X. Lines $16 \sim 21$ are justified by the rule of update using six LFSRs. Finally, the algorithm returns $ks^{(N)}$ as the estimated degree of the first keystream bit after N initialization rounds.

Theorem 1. *Algorithm 3 gives an upper bound on the algebraic degree of the first keystream bit of N-round ACORN cipher with X as cube variables.*

We give our proof of Theorem 1 in Appendix B. Both $\texttt{KSG128}(t)$ and $\texttt{FBK128}(t)$ can be executed in constant time, thus Algorithm 3 has a time complexity of $\mathcal{O}(N)$. It requires a memory of $\mathcal{O}(N)$.

Experimental Results. In Algorithm 1, we need to specify the number of iterations m, vector $\boldsymbol{k}$, vector $\boldsymbol{n}$, and vector $\boldsymbol{\alpha}$. We set the maximum size of cube to 66. The number of iterations m is 16. At each iteration, we retain 100 best cubes of the current size. At the first iteration, we exhaust all possible cubes of size 6. After that, we add 4 new bits at each iteration.

We list part of output of our program in Table 2 in Appendix C. Our experiments show that, with the new method of finding cubes, we can surely

Algorithm 3: Degree Estimation of ACORN Cipher

Require: Given the initialization rounds N, and the set X of cube variables.

for t **from** 0 **to** 292 **do**
  $d^{(t)} \leftarrow -\infty$;
end for
for t **from** 0 **to** $N-1$ **do**
  $ks^{(t)} \leftarrow \texttt{KSG128}(t)$;
  $f^{(t)} \leftarrow \texttt{FBK128}(t)$;
  if $128 \leq t \leq 255$ **then**
    if $IV_{t-128} \in X$ **then**
      $d^{(t+293)} \leftarrow \max\{f^{(t)},\ 1\}$;
    else
      $d^{(t+293)} \leftarrow f^{(t)}$;
    end if
  else
    $d^{(t+293)} \leftarrow \max\{f^{(t)},\ 0\}$;
  end if
  $d^{(t+289)} \leftarrow \max\{d^{(t+289)},\ d^{(t+235)},\ d^{(t+230)}\}$;
  $d^{(t+230)} \leftarrow \max\{d^{(t+230)},\ d^{(t+196)},\ d^{(t+193)}\}$;
  $d^{(t+193)} \leftarrow \max\{d^{(t+193)},\ d^{(t+160)},\ d^{(t+154)}\}$;
  $d^{(t+154)} \leftarrow \max\{d^{(t+154)},\ d^{(t+111)},\ d^{(t+107)}\}$;
  $d^{(t+107)} \leftarrow \max\{d^{(t+107)},\ d^{(t+66)},\ d^{(t+61)}\}$;
  $d^{(t+61)} \leftarrow \max\{d^{(t+61)},\ d^{(t+23)},\ d^{(t)}\}$;
end for
$ks^{(N)} \leftarrow \texttt{KSG128}(N)$;
return $ks^{(N)}$;

obtain better cryptanalytic results with a much shorter time. For example, we have found a cube tester when the cube size increases to 54, and the number of rounds we can attack is 715.

Moreover, we have also obtained a cube of a practical size 38, which can lead to a 690-round zero-sum distinguisher. We randomly select 64 different keys, and then compute the 64 different cubesums of the first 64 keystream bits produced by 690-round ACORN. The results show that the cubesums of the first keystream bit are always equal to zero. The cubesums of other keystream bits are all non-zero. This result further justifies the 690-round zero-sum cube tester.

5 Searching Cubes in Division Property Based Cube Attack

In this section, we propose a method to find zero-sum cube testers for an iterated cipher via the division property based cube attack. We apply this method to ACORN, and obtain the best distinguishing attack for 775-round ACORN. Based on the embedded property for cube tester, we also propose an algorithm

to find suitable cubes for key recovery attacks. In the end, we find the best key recovery attack for 772-round ACORN.

5.1 Finding Cube Testers via Division Property Based Cube Attack

In [18], Sun *et al.* proposed the algorithms to find optimal integral distinguishers for ARX ciphers by exploiting the embedded property (see Appendix D). Next, we will show that, the embedded property can apply to the search of cube testers as well. We first introduce the following lemma which was proposed in [23].

Lemma 1 ([23]). *If $\boldsymbol{k} \succeq \boldsymbol{k}'$ and there is division trail $\boldsymbol{k} \xrightarrow{f} \boldsymbol{l}$, then there is also division trail $\boldsymbol{k}' \xrightarrow{f} \boldsymbol{l}'$ s.t. $\boldsymbol{l} \succeq \boldsymbol{l}'$.*

Based on this lemma, we propose the following proposition.

Proposition 3. *Let $f(\boldsymbol{x}, \boldsymbol{v})$ be a polynomial, where $\boldsymbol{x}$ and $\boldsymbol{v}$ denote the secret and public variables, respectively. For a set of indices $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{0, 1, \ldots, m-1\}$, let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $\boldsymbol{v}^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \ldots v_{i_{|I|}}$. Let $\boldsymbol{k}_\Lambda$ be an n-dimensional bit vector. For a given set of indices $I_S \subsetneq I$, if there is no division trail such that $(\boldsymbol{k}_\Lambda || \boldsymbol{k}_{I_S}) \xrightarrow{f} 1$ for any $\boldsymbol{k}_\Lambda \in \mathbb{F}_2^n$, then there is also no division trail such that $(\boldsymbol{k}_\Lambda || \boldsymbol{k}_I) \xrightarrow{f} 1$ for any $\boldsymbol{k}_\Lambda \in \mathbb{F}_2^n$.*

Proof. From Lemma 1, if $\boldsymbol{k} \succeq \boldsymbol{k}'$ and there is division trail $\boldsymbol{k} \xrightarrow{f} 1$, then there is also division trail $\boldsymbol{k}' \xrightarrow{f} 1$. Suppose there is a division trail such that $(\boldsymbol{k}^*_\Lambda || \boldsymbol{k}_I) \xrightarrow{f} 1$ for a fixed $\boldsymbol{k}^*_\Lambda \in \mathbb{F}_2^n$, then there is also a division trail such that $(\boldsymbol{k}^*_\Lambda || \boldsymbol{k}_{I_S}) \xrightarrow{f} 1$ (since $(\boldsymbol{k}^*_\Lambda || \boldsymbol{k}_I) \succeq (\boldsymbol{k}^*_\Lambda || \boldsymbol{k}_{I_S})$), which leads to a contradiction. □

Denote the superpoly of cube I by $p_S(I)$. From the above Proposition and Proposition 2 in Section 2, we know that,

Proposition 4 (Embedded Property for Cube Tester). *For an r-round iterated cipher, if I_S is a subset of cube I, and there is no monomials in $p_S(I_S)$ (*i.e.*, $\deg(p_S(I_S)) = 0$), then there is also no monomials in $p_S(I)$ (*i.e.*, $\deg(p_S(I)) = 0$). Likewise, if $\deg(p_S(I)) \neq 0$, then $\deg(p_S(I_S)) \neq 0$.*

This simple property helps to search cube testers efficiently in the division property based cube attack scenario. In the following, we propose Algorithm 4, 5, and 6 to efficiently reduce the complexity of searching. Algorithm 4 is used to determine the maximum number of distinguishable rounds r_m for a given cube indices I. Algorithm 4 can be seen as a subfunction of Algorithm 5. In Algorithm 5, we determine the maximum number of distinguishable rounds for a specific cipher, and restrict the search scope. In Algorithm 6, we use the output of Algorithm 5 as input, and returns a set of zero-sum cube testers.

The basic idea of Algorithm 4 is binary search, which can reduce the complexity of searching. In Algorithm 4, we set two variables r_h and r_l to indicate the upper bound and lower bound of the maximum number of distinguishable

Algorithm 4: Determining the Maximum Number of Distinguishable Rounds for a Given Cube

Input : Iterated cipher f with R initialzation rounds, cube indices I.
Output: The maximum number of distinguishable rounds r_m for cube I.

1: $r_h = R,\ r_l = 0,\ r_m = 0,\ r = 0,\ flag = 0$;
2: **while** $r_h - r_l > 1$ **do**
3: $r = \lfloor (r_h + r_l)/2 \rfloor$
4: use Algorithm `A` to evaluate the degree d of the superpoly of cube I for f reduced to r rounds;
5: **if** $d == 0$ **then**
6: $r_l = r,\ flag = 0$;
7: **else**
8: $r_h = r,\ flag = 1$;
9: **end if**
10: **end while**
11: **if** $flag == 0$ **then**
12: $r_m = r$;
13: **else**
14: $r_m = r - 1$;
15: **end if**
16: **return** r_m;

rounds r_m for a specific cipher. For a cipher f with R initialzation rounds, we first use Algorithm `A` to evaluate the degree d of the superpoly of cube I for f reduced to $\lfloor (r_h + r_l)/2 \rfloor = \lfloor R/2 \rfloor$ rounds. If $d == 0$, then r_m is at least $\lfloor R/2 \rfloor$, so we set $r_l = r$. Otherwise, we set $r_h = r$. We iteratively repeat this process, so the distance between r_h and r_l can be reduced quickly. In the end, we can determine the value of r_m with at most $\lceil \log_2 R \rceil$ iterations.

In Algorithm 5, we first check all cubes of dimension $m - 1$, where m is the number of public variables. For each $(m-1)$-dimensional cube I, we use Algorithm 4 to compute its maximum number of distinguishable rounds as cube testers. Among all m cubes, we select those cubes which can lead to the longest (r_{max}-round) cube tester, and store their missing index i of public variables in $\mathbb{S}$. We claim that the elements in the complementary set $\bar{\mathbb{S}} = \{0, 1, \ldots, m-1\} \backslash \mathbb{S}$ of $\mathbb{S}$ are 'necessary' bit indices to obtain an r_{max}-round cube tester. By Proposition 4, if any index which belongs to $\bar{\mathbb{S}}$ is not in cube indices I, then this cube will not lead to an r_{max}-round cube tester. In the following, we call $\bar{\mathbb{S}}$ the *necessary set*, whose elements must be in the cube indices, while $\mathbb{S}$ is called the *sufficient set*, and the elements in $\mathbb{S}$ are called *sufficient indices*.

In Algorithm 6, we first test whether the cube $I = \{0, 1, \ldots, m-1\} \setminus \mathbb{S}$ will lead to the r_{max}-round cube tester. If not, we gradually increase the dimension of cubes by reducing the value of t where we pick t indices from $\mathbb{S}$, and check whether the cube tester exists or not. After t is fixed (Line 14 in Algorithm 6), there exists at least one cube which will lead to the r_{max}-round cube tester, so Algorithm 6 returns a set of zero-sum cube testers.

Algorithm 5: Determining the Maximum Number of Distinguishable Rounds & Restricting the Search Scope

Input : Iterated cipher f with m public variables $(v_0, \ldots, v_{m-1})$.
Output: The maximum number of distinguishable rounds r_{max} of cube testers, and the index set $\mathbb{S}$.

$r_{max} = 0$, $\mathbb{S} = \emptyset$;
for $i = 0$; $i < m$ **do**
  let cube indices $I_i = \{0, 1, \ldots, m-1\} \setminus \{i\}$;
  use Algorithm 4 to compute the maximum number of distinguishable rounds r_i for cube I_i, and store (I_i, r_i);
  if $r_{max} < r_i$ **then**
    $r_{max} = r_i$;
  end if
end for
for $i = 0$; $i < m$ **do**
  if $r_i == r_{max}$ **then**
    $\mathbb{S} = \mathbb{S} \cup \{i\}$;
  end if
end for
return $r_{max}, \mathbb{S}$;

5.2 Finding Cubes for Key Recovery Attacks

In this subsection, we give an algorithm (see Algorithm 7) to search a suitable cube which might lead to a key recovery attack for an iterated cipher f reduced to r-rounds.

The basic idea of Algorithm 7 is that a cube which leads to an r-round cube tester might lead to an $(r+1)$-round cube attack as well, as long as the cube dimension and the complexity to recover the superpoly is reasonable. The complexity to recover the superpoly is $2^{|I|} \times (1 + \sum_{t=1}^{d} |J_t|)$ [23], where I is the cube indices, d is the algebraic degree of the superpoly, and J_t is all possible terms of degree t. Suppose the length of key is n bits. To recover secret information, the superpoly should contain at least 1 key bit. To make the attack meaningful, we also have $2^{|I|} \times (1 + \sum_{t=1}^{d} |J_t|) < 2^n$. Thus we need to restrict $|I| + 1 < n$. In other words, the dimension of cube can not exceed $n - 2$.

Suppose there are m public variables. In Algorithm 7, we first find the sufficient set $\mathbb{S}$ for $(r-1)$-round cube tester. Then we gradually test cubes of dimensions range from $m-2$ to $m-|\mathbb{S}|$. For $(m-k)$-dimensional cube, there are $\binom{|\mathbb{S}|}{k}$ different choices. If a cube I can lead to the $(r-1)$-round cube tester, then we test whether this cube would lead to the key recovery attack of the r-round cipher. For all tested cubes of dimension $(m-k+1)$, if none of them can lead to the $(r-1)$-round cube tester, then by the embedded property for cube tester, we do not need to test cubes of dimension $(m-k)$, we just quit the while loop and return an emptyset.

Algorithm 6: Finding the Zero-Sum Cube Testers

Input : Iterated cipher f with m public variables $(v_0, \ldots, v_{m-1})$, the maximum number of distinguishable rounds r_{max} of cube testers, and the sufficient set $\mathbb{S}$.

Output: A set Res containing the zero-sum cube testers.

$Res = \emptyset$, $flag = 0$;
$t = |\mathbb{S}|$;
while $flag == 0$ **do**
  for every t-tuple $(i_0, i_1, \ldots, i_{t-1})$ of $\mathbb{S}$ **do**
    let cube indices $I = \{0, 1, \ldots, m-1\} \setminus \{i_0, i_1, \ldots, i_{t-1}\}$;
    use Algorithm `A` to evaluate the degree d of the superpoly of cube I for f reduced to r_{max} rounds;
    if $d == 0$ **then**
      $flag = 1$;
      break;
    end if
  end for
  $t = t - 1$;
end while
$t = t + 1$;
for every t-tuple $(i_0, i_1, \ldots, i_{t-1})$ of $\mathbb{S}$ **do**
  let cube indices $I = \{0, 1, \ldots, m-1\} \setminus \{i_0, i_1, \ldots, i_{t-1}\}$;
  use Algorithm `A` to evaluate the degree d of the superpoly of cube I for f reduced to r_{max} rounds;
  if $d == 0$ **then**
    $Res = Res \cup \{I\}$;
  end if
end for
return Res;

5.3 Applications to ACORN

In this part, we apply our methods of finding cubes for cube testers and key recovery attacks to ACORN. Our experiments are based on the MILP model of division property for ACORN, Algorithm `A`, and Algorithm `B` given in [23]. We use Gurobi Optimizer [11] with Python interface to solve the MILP problems.

Cube Testers of 775-round ACORN. Using Algorithm 6, we find 6 cubes of dimension 127, all of which can lead to the cube tester for 775-round ACORN. The cube indices are as follows.

$I_1 = \{0, 1, \ldots, 127\} \setminus \{1\}$, $I_2 = \{0, 1, \ldots, 127\} \setminus \{2\}$, $I_3 = \{0, 1, \ldots, 127\} \setminus \{11\}$,
$I_4 = \{0, 1, \ldots, 127\} \setminus \{18\}$, $I_5 = \{0, 1, \ldots, 127\} \setminus \{26\}$, $I_6 = \{0, 1, \ldots, 127\} \setminus \{27\}$.

A Key Recovery Attack of 772-round ACORN. Using Algorithm 7, we find a cube of dimension 123 which can lead to a key recovery attack for 772-

Algorithm 7: Finding Cube for Key Recovery Attack

Input : Iterated cipher f reduced to r-rounds, with m public variables $(v_0, \ldots, v_{m-1})$ and n secret variables $(x_0, \ldots, x_{n-1})$.

Output: Cube indices I for key recovery attack of f reduced to r-rounds.

$r_D = r - 1,\ flag = 1$;
similar to Algorithm 5, find the sufficient set $\mathbb{S}$ for f reduced to r_D rounds;
if $|\mathbb{S}| \leq 1$ **then**
  return $\emptyset$;
end if
$t = 2$;
while $flag == 1$ **do**
  $flag = 0$;
  for every t-tuple $(i_0, i_1, \ldots, i_{t-1})$ of $\mathbb{S}$ **do**
    let cube indices $I = \{0, 1, \ldots, m-1\} \setminus \{i_0, i_1, \ldots, i_{t-1}\}$;
    use Algorithm `A` to compute the degree d_D of the superpoly of cube I for f reduced to r_D rounds;
    if $d_D == 0$ **then**
      $flag = 1$;
      use Algorithm `A` and Algorithm `B` to compute the degree d and the involved monomials $J_1, \ldots, J_d$ of the superpoly of cube I for f reduced to r rounds, where J_k is all possible terms of degree k;
      if $2^{|I|} \times (1 + \sum_{k=1}^{d} |J_k|) < 2^n$ **then**
        return I;
      end if
    end if
  end for
  if $t == |\mathbb{S}|$ **then**
    break;
  end if
  $t = t + 1$;
end while
return $\emptyset$;

round ACORN. The cube indices is as follows,

$$I = \{0, 1, \ldots, 127\} \setminus \{1, 2, 11, 26, 27\}.$$

By running Algorithm `A`, we know the degree of the superpoly of this cube is 1. Using Algorithm `B`, we know only the following 21 key bits might exist in the superpoly,

$$\texttt{key indices} := \{0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 19, 24, 31, 33, 35, 39, 41, 44, 45, 78\}.$$

Thus, the complexity to recover the linear superpoly is $2^{123} \times (1 + 21) \approx 2^{127.46}$. In online phase, we can sum over this cube with a complexity of 2^{123}, then we obtain one bit secret information, and the remaining 127 bits can be recovered by brute force.

6 Conclusions

In this paper, we analyzed the security of the authenticated stream cipher ACORN with cube cryptanalysis. Firstly, we gave a new method of finding cube testers. We applied it to ACORN, and obtained the best practical distinguishing attacks. Then we proposed several algorithms to search zero-sum cube testers and suitable cubes for key recovery in the division property based cube attack. We found some new distinguishers for 775-round ACORN. We also found a key recovery attack on ACORN reduced to 772 rounds.

Acknowledgements. We are grateful to the anonymous reviewers of ISC 2019.

A KSG128(t) and FBK128(t)

Algorithm 8: KSG128(t)

1: $d_1 \leftarrow \max\{d^{(t+12)},\ d^{(t+154)},\ d^{(t+111)}\}$;
2: $d_2 \leftarrow \max\{d^{(t+61)},\ d^{(t+23)},\ d^{(t)}\}$;
3: $d_3 \leftarrow \max\{d^{(t+193)},\ d^{(t+160)},\ d^{(t+154)}\}$;
4: $d_4 \leftarrow \max\{d^{(t+230)},\ d^{(t+196)},\ d^{(t+193)}\}$;
5: $d \leftarrow \max\{d_1,\ d^{(t+107)},\ d^{(t+235)} + \max\{d_2,\ d_3\},$
$d_2 + d_3,\ \max\{d^{(t+111)},\ d^{(t+66)}\} + d_4,\ d^{(t+66)}\}$;
6: **return** d;

Algorithm 9: FBK128(t)

1: $d_1 \leftarrow \max\{d^{(t+12)},\ d^{(t+154)},\ d^{(t+111)}\}$;
2: $d_2 \leftarrow \max\{d^{(t+61)},\ d^{(t+23)},\ d^{(t)}\}$;
3: $d_3 \leftarrow \max\{d^{(t+193)},\ d^{(t+160)},\ d^{(t+154)}\}$;
4: $d_4 \leftarrow \max\{d^{(t+230)},\ d^{(t+196)},\ d^{(t+193)}\}$;
5: $d \leftarrow \max\{d^{(t)},\ d^{(t+61)},\ 0,\ d^{(t+244)} + \max\{d^{(t+23)},\ d^{(t+160)}\},\ d^{(t+196)},\ d_1,$
$d^{(t+235)} + \max\{d_2,\ d_3\},\ d^{(t+61)} + d_3,\ \max\{d^{(t+193)},\ d^{(t+154)}\} + d^{(t+23)},$
$d^{(t)} + d_3,\ \max\{d^{(t+111)},\ d^{(t+66)}\} + d_4\}$;
6: **return** d;

B Proof of Theorem 1

Proof. It is sufficient to justify Algorithm 8 and Algorithm 9. By the rule of state update of ACORN, we have

$$\begin{aligned} ks_t &= s_{t+12} \oplus (s_{t+154} \oplus s_{t+111} \oplus s_{t+107}) \oplus s_{t+235}(s_{t+61} \oplus s_{t+23} \oplus s_t) \oplus s_{t+235} \\ &\quad (s_{t+193} \oplus s_{t+160} \oplus s_{t+154}) \oplus (s_{t+61} \oplus s_{t+23} \oplus s_t)(s_{t+193} \oplus s_{t+160} \oplus s_{t+154}) \\ &\quad \oplus (s_{t+230} \oplus s_{t+196} \oplus s_{t+193})(s_{t+111} \oplus s_{t+66}) \oplus s_{t+66} \\ &= \underline{(s_{t+12} \oplus s_{t+154} \oplus s_{t+111})} \oplus \underline{s_{t+107}} \oplus \underline{s_{t+235}((s_{t+61} \oplus s_{t+23} \oplus s_t) \oplus (s_{t+193}} \\ &\quad \underline{\oplus s_{t+160} \oplus s_{t+154}))} \oplus \underline{(s_{t+61} \oplus s_{t+23} \oplus s_t)(s_{t+193} \oplus s_{t+160} \oplus s_{t+154})} \\ &\quad \oplus \underline{(s_{t+230} \oplus s_{t+196} \oplus s_{t+193})(s_{t+111} \oplus s_{t+66})} \oplus \underline{s_{t+66}}, \end{aligned}$$

and

$$\begin{aligned} f_t &= s_t \oplus (s_{t+107} \oplus s_{t+66} \oplus s_{t+61}) \oplus 1 \oplus s_{t+244}s_{t+23} \oplus s_{t+244}s_{t+160} \oplus s_{t+23}s_{t+160} \\ &\quad \oplus s_{t+196} \oplus s_{t+12} \oplus (s_{t+154} \oplus s_{t+111} \oplus s_{t+107}) \oplus s_{t+235}(s_{t+61} \oplus s_{t+23} \oplus s_t) \oplus \\ &\quad s_{t+235}(s_{t+193} \oplus s_{t+160} \oplus s_{t+154}) \oplus (s_{t+61} \oplus s_{t+23} \oplus s_t)(s_{t+193} \oplus s_{t+160} \oplus \\ &\quad s_{t+154}) \oplus (s_{t+230} \oplus s_{t+196} \oplus s_{t+193})(s_{t+111} \oplus s_{t+66}) \oplus s_{t+66} \\ &= \underline{s_t} \oplus \underline{s_{t+61}} \oplus \underline{1} \oplus \underline{s_{t+244}(s_{t+23} \oplus s_{t+160})} \oplus \underline{s_{t+196}} \oplus \underline{(s_{t+12} \oplus s_{t+154} \oplus s_{t+111})} \\ &\quad \oplus \underline{s_{t+235}((s_{t+61} \oplus s_{t+23} \oplus s_t) \oplus (s_{t+193} \oplus s_{t+160} \oplus s_{t+154}))} \oplus \underline{s_{t+61}(s_{t+193} \oplus} \\ &\quad \underline{s_{t+160} \oplus s_{t+154})} \oplus \underline{s_{t+23}(s_{t+193} \oplus s_{t+154})} \oplus \underline{s_t(s_{t+193} \oplus s_{t+160} \oplus s_{t+154})} \oplus \\ &\quad \underline{(s_{t+230} \oplus s_{t+196} \oplus s_{t+193})(s_{t+111} \oplus s_{t+66})}. \end{aligned}$$

In the above expressions, each underlined term corresponds to an estimated algebraic degree. For example, we have

$$\begin{aligned} \deg(s_{t+244}(s_{t+23} \oplus s_{t+160})) &\le \deg(s_{t+244}) + \max\{\deg(s_{t+23}), \deg(s_{t+160})\} \\ &\le d^{(t+244)} + \max\{d^{(t+23)}, d^{(t+160)}\}. \end{aligned}$$

Hence, the super numeric degree of $s_{t+244}(s_{t+23} \oplus s_{t+160})$ is $d^{(t+244)} + \max\{d^{(t+23)}, d^{(t+160)}\}$, which can be found at line 5 in Algorithm 9. One can check that the degrees of all underlined terms are evaluated. Thus, our algorithms of degree estimation of ACORN are correct. □

C Cube Testers of Different Dimensions

Table 2: Some cubes found by the accelerated greedy algorithm. The sizes of these cubes range from 38 to 54.

Cube Size	#Rounds	Cube Indexes
38	690	52, 53, 58, 62, 63, 77, 82, 84, 86, 87, 88, 91, 92, 93, 96, 97, 101, 102, 106, 107, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127

Table 2: Some cubes found by the accelerated greedy algorithm. The sizes of these cubes range from 38 to 54.

Cube Size	#Rounds	Cube Indexes
42	697	52, 53, 58, 60, 63, 77, 82, 84, 85, 86, 87, 89, 90, 91, 92, 93, 96, 97, 101, 102, 103, 106, 107, 108, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127
46	706	52, 53, 57, 58, 60, 62, 63, 69, 77, 78, 82, 84, 85, 86, 87, 89, 90, 91, 92, 93, 96, 97, 101, 102, 103, 106, 107, 108, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127
50	706	44, 48, 49, 52, 53, 58, 60, 61, 63, 69, 77, 81, 82, 83, 84, 85, 86, 87, 89, 91, 92, 93, 96, 97, 99, 100, 101, 102, 103, 106, 107, 108, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127
54	715	48, 52, 53, 58, 60, 62, 63, 67, 69, 70, 77, 78, 81, 82, 83, 84, 85, 86, 87, 88, 89, 91, 92, 93, 95, 96, 97, 99, 100, 101, 102, 103, 104, 106, 107, 108, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127

D Embedded Property

The embedded property [18] says that, for different initial division properties $\boldsymbol{k}_0$ and $\boldsymbol{k}_1$ *s.t.* $\boldsymbol{k}_0 \succeq \boldsymbol{k}_1$, there is no need to test $\boldsymbol{k}_1$, if the output multi-set under $\boldsymbol{k}_0$ does not have integral property, likewise, it is not necessary to test $\boldsymbol{k}_0$, if the output multi-set under $\boldsymbol{k}_1$ has integral property.

References

1. Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128. SHARCS09 Special-purpose Hardware for Attacking Cryptographic Systems p. 147 (2009)
2. Aumasson, J.P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Fast Software Encryption. pp. 1–22. Springer (2009)
3. Aumasson, J.P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: A lightweight hash. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 1–15. Springer (2010)
4. Aumasson, J.P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: A lightweight hash. Journal of cryptology **26**(2), 313–339 (2013)
5. Aumasson, J.P., Knellwolf, S., Meier, W.: Heavy Quark for secure AEAD. DIAC-Directions in Authenticated Ciphers (2012)
6. De Canniere, C.: Trivium: A stream cipher construction inspired by block cipher design principles. In: International Conference on Information Security. pp. 171–186. Springer (2006)
7. De Canniere, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers. In: Cryptographic Hardware and Embedded Systems-CHES 2009, pp. 272–288. Springer (2009)
8. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 278–299. Springer (2009)
9. Fouque, P.A., Vannet, T.: Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In: International Workshop on Fast Software Encryption. pp. 502–517. Springer (2013)

10. 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. Journal of Ambient Intelligence and Humanized Computing pp. 1–8 (2018)
11. Gurobi: Gurobi Optimizer. http://www.gurobi.com/
12. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of stream ciphers. In: New Stream Cipher Designs, pp. 179–190. Springer (2008)
13. Karlsson, L., Hell, M., Stankovski, P.: Improved greedy nonrandomness detectors for stream ciphers. In: ICISSP. pp. 225–232 (2017)
14. Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Annual International Cryptology Conference. pp. 227–249. Springer (2017)
15. Liu, M., Lin, D., Wang, W.: Searching cubes for testing boolean functions and its application to Trivium. In: ISIT. pp. 496–500. IEEE (2015)
16. Salam, M.I., Bartlett, H., Dawson, E., Pieprzyk, J., Simpson, L., Wong, K.K.: Investigating cube attacks on the authenticated encryption stream cipher ACORN. In: Applications and Techniques in Information Security - 6th International Conference, ATIS 2016, Cairns, QLD, Australia, October 26-28, 2016, Proceedings. pp. 15–26 (2016)
17. Stankovski, P.: Greedy distinguishers and nonrandomness detectors. In: International Conference on Cryptology in India. pp. 210–226. Springer (2010)
18. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 128–157. Springer (2017)
19. Todo, Y.: Structural evaluation by generalized integral property. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 287–314. Springer (2015)
20. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Annual International Cryptology Conference. pp. 250–279. Springer (2017)
21. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. IACR Cryptology ePrint Archive **2017**, 306 (2017), http://eprint.iacr.org/2017/306
22. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: International Conference on Fast Software Encryption. pp. 357–377. Springer (2016)
23. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. IACR Cryptology ePrint Archive **2017**, 1063 (2017), http://eprint.iacr.org/2017/1063
24. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In: Annual International Cryptology Conference. pp. 275–305. Springer (2018)
25. Wu, H.: ACORN: a lightweight authenticated cipher (v3). Candidate for the CAESAR Competition (2016), https://competitions.cr.yp.to/round3/acornv3.pdf
26. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 648–678. Springer (2016)