



Improved Division Property Based Cube Attacks Exploiting Algebraic Properties of Superpoly

Qingju Wang^{1,2,3}, Yonglin Hao^{4(✉)}, Yosuke Todo^{5(✉)}, Chaoyun Li^{6(✉)},
Takanori Isobe⁷, and Willi Meier⁸

¹ Shanghai Jiao Tong University, Shanghai, China

² Technical University of Denmark, Kongens Lyngby, Denmark

³ SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
qingju.wang@uni.lu

⁴ State Key Laboratory of Cryptology, Beijing, China
haoyonglin@yeah.net

⁵ NTT Secure Platform Laboratories, Tokyo, Japan
todo.yosuke@lab.ntt.co.jp

⁶ imec-COSIC, Department of Electrical Engineering (ESAT),
KU Leuven, Leuven, Belgium
chaoyun.li@esat.kuleuven.be

⁷ University of Hyogo, Kobe, Japan
takanori.isobe@ai.u-hyogo.ac.jp

⁸ FHNW, Windisch, Switzerland
willi.meier@fhnw.ch

Abstract. The cube attack is an important technique for the cryptanalysis of symmetric key primitives, especially for stream ciphers. Aiming at recovering some secret key bits, the adversary reconstructs a superpoly with the secret key bits involved, by summing over a set of the plaintexts/IV which is called a cube. Traditional cube attack only exploits linear/quadratic superpolies. Moreover, for a long time after its proposal, the size of the cubes has been largely confined to an experimental range, e.g., typically 40. These limits were first overcome by the division property based cube attacks proposed by Todo et al. at CRYPTO 2017. Based on MILP modelled division property, for a cube (index set) I , they identify the small (index) subset J of the secret key bits involved in the resultant superpoly. During the precomputation phase which dominates the complexity of the cube attacks, $2^{|I|+|J|}$ encryptions are required to recover the superpoly. Therefore, their attacks can only be available when the restriction $|I| + |J| < n$ is met.

In this paper, we introduced several techniques to improve the division property based cube attacks by exploiting various algebraic properties of the superpoly.

1. We propose the “flag” technique to enhance the preciseness of MILP models so that the proper non-cube IV assignments can be identified to obtain a non-constant superpoly.

2. A degree evaluation algorithm is presented to upper bound the degree of the superpoly. With the knowledge of its degree, the superpoly can be recovered without constructing its whole truth table. This enables us to explore larger cubes I 's even if $|I| + |J| \geq n$.
3. We provide a term enumeration algorithm for finding the monomials of the superpoly, so that the complexity of many attacks can be further reduced.

As an illustration, we apply our techniques to attack the initialization of several ciphers. To be specific, our key recovery attacks have mounted to 839-round TRIVIUM, 891-round Kreyvium, 184-round Grain-128a and 750-round ACORN respectively.

Keywords: Cube attack · Division property · MILP · TRIVIUM
Kreyvium · Grain-128a · ACORN · Clique

1 Introduction

Cube attack, proposed by Dinur and Shamir [1] in 2009, is one of the general cryptanalytic techniques of analyzing symmetric-key cryptosystems. After its proposal, cube attack has been successfully applied to various ciphers, including stream ciphers [2–6], hash functions [7–9], and authenticated encryptions [10, 11]. For a cipher with n secret variables $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and m public variables $\mathbf{v} = (v_1, v_2, \dots, v_m)$, we can regard the algebraic normal form (ANF) of output bits as a polynomial of \mathbf{x} and \mathbf{v} , denoted as $f(\mathbf{x}, \mathbf{v})$. For a randomly chosen set $I = \{i_1, i_2, \dots, i_{|I|}\} \subset \{1, \dots, m\}$, $f(\mathbf{x}, \mathbf{v})$ can be represented uniquely as

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

where $t_I = v_{i_1} \cdots v_{i_{|I|}}$, $p(\mathbf{x}, \mathbf{v})$ only relates to v_s 's ($s \notin I$) and the secret key bits \mathbf{x} , and $q(\mathbf{x}, \mathbf{v})$ misses at least one variable in t_I . When v_s 's ($s \notin I$) and \mathbf{x} are assigned statically, the value of $p(\mathbf{x}, \mathbf{v})$ can be computed by summing the output bit $f(\mathbf{x}, \mathbf{v})$ over a structure called *cube*, denoted as C_I , consisting of $2^{|I|}$ different \mathbf{v} vectors with $v_i, i \in I$ being active (traversing all 0-1 combinations) and non-cube indices $v_s, s \notin I$ being static constants. Traditional cube attacks are mainly concerned about linear or quadratic superpolies. By collecting linear or quadratic equations from the superpoly, the attacker can recover some secret key bits information during the online phase. Aiming to mount distinguishing attack by property testing, cube testers are obtained by evaluating superpolies of carefully selected cubes. In [2], probabilistic tests are applied to detect some algebraic properties such as constantness, low degree and sparse monomial distribution. Moreover, cube attacks and cube testers are acquired experimentally by summing over randomly chosen cubes. So the sizes of the cubes are largely confined. Breakthroughs have been made by Todo et al. in [12] where they introduce the bit-based division property, a tool for conducting integral attacks¹,

¹ Integral attacks also require to traverse some active plaintext bits and check whether the summation of the corresponding ciphertext bits have zero-sum property, which equals to check whether the superpoly has $p(\mathbf{x}, \mathbf{v}) \equiv 0$.

to the realm of cube attack. With the help of mixed integer linear programming (MILP) aided division property, they can identify the variables excluded from the superpoly and explore cubes with larger size, e.g., 72 for 832-round TRIVIUM. This enables them to improve the traditional cube attack.

Division property, as a generalization of the integral property, was first proposed at EUROCRYPT 2015 [13]. With division property, the propagation of the integral characteristics can be deduced in a more accurate manner, and one prominent application is the first theoretic key recovery attack on full MISTY1 [14].

The original division property can only be applied to word-oriented primitives. At FSE 2016, bit-based division property [15] was proposed to investigate integral characteristics for bit-based block ciphers. With the help of division property, the propagation of the integral characteristics can be represented by the operations on a set of 0-1 vectors identifying the bit positions with the zero-sum property. Therefore, for the first time, integral characteristics for bit-based block ciphers SIMON32 and Simeck32 have been proved. However, the sizes of the 0-1 vector sets are exponential to the block size of the ciphers. Therefore, as has been pointed out by the authors themselves, the deduction of bit-based division property under their framework requires high memory for block ciphers with larger block sizes, which largely limits its applications. Such a problem has been solved by Xiang et al. [16] at ASIACRYPT 2016 by utilizing the MILP model. The operations on 0-1 vector sets are transformed to imposing division property values (0 or 1) to MILP variables, and the corresponding integral characteristics are acquired by solving the models with MILP solvers like Gurobi [17]. With this method, they are able to give integral characteristics for block ciphers with block sizes much larger than 32 bits. Xiang et al.'s method has now been applied to many other ciphers for improved integral attacks [18–21].

In [12], Todo et al. adapt Xiang et al.'s method by taking key bits into the MILP model. With this technique, a set of key indices $J = \{j_1, j_2, \dots, j_{|J|}\} \subset \{1, \dots, n\}$ is deduced for the cube C_I s.t. $p(\mathbf{x}, \mathbf{v})$ can only be related to the key bits x_j 's ($j \in J$). With the knowledge of I and J , Todo et al. can recover 1-bit of secret-key-related information by executing two phases. In the *offline phase*, a proper assignment to the non-cube IVs, denoted by $\mathbf{IV} \in \mathbb{F}_2^m$, is determined ensuring $p(\mathbf{x}, \mathbf{IV})$ non-constant. Also in this phase, the whole truth table of $p(\mathbf{x}, \mathbf{IV})$ is constructed through cube summations. In the *online phase*, the exact value of $p(\mathbf{x}, \mathbf{IV})$ is acquired through a cube summation and the candidate values of x_j 's ($j \in J$) are identified by checking the precomputed truth table. A proportion of wrong keys are filtered as long as $p(\mathbf{x}, \mathbf{IV})$ is non-constant.

Due to division property and the power of MILP solver, cubes of larger dimension can now be used for key recoveries. By using a 72-dimensional cube, Todo et al. propose a theoretic cube attack on 832-round TRIVIUM. They also largely improve the previous best attacks on other primitives namely ACORN, Grain-128a and Kreyvium [12, 22]. It is not until recently that the result on TRIVIUM has been improved by Liu et al. [6] mounting to 835 rounds with a new method called the *correlation* cube attack. The correlation attack is based on the *numeric mapping* technique first appeared in [23] originally used for constructing zero-sum distinguishers.

1.1 Motivations

Due to [12,22], the power of cube attacks has been enhanced significantly, however, there are still problems remaining unhandled that we will reveal explicitly.

Finding Proper IV 's May Require Multiple Trials. As is mentioned above, the superpoly can filter wrong keys only if a *proper IV assignment* $IV \in \mathbb{F}_2^m$ in the constant part of IV s is found such that the corresponding superpoly $p(\mathbf{x}, IV)$ is non-constant. The MILP model in [12,22] only proves the existence of the proper IV 's but finding them may not be easy. According to practical experiments, there are quite some IV 's making $p(\mathbf{x}, IV) \equiv 0$. Therefore, $t \geq 1$ different IV 's might be trailed in the precomputation phase before finding a proper one. Since each IV requires to construct a truth table with complexity $2^{|I|+|J|}$, the overall complexity of the offline phase can be $t \times 2^{|I|+|J|}$. When large cubes are used ($|I|$ is big) or many key bits are involved ($|J|$ is large), such a complexity might be at the risk of exceeding the brute-force bound 2^n . Therefore, two assumptions are made to validate their cube attacks as follows.

Assumption 1 (Strong Assumption). *For a cube C_I , there are many values in the constant part of IV whose corresponding superpoly is balanced.*

Assumption 2 (Weak Assumption). *For a cube C_I , there are many values in the constant part of IV whose corresponding superpoly is not a constant function.*

These assumptions are proposed to guarantee the validity of the attacks as long as $|I| + |J| < n$, but the rationality of such assumptions is hard to be proved, especially when $|I| + |J|$ are so close to n in many cases. The best solution is to evaluate different IV s in the MILP model so that the proper IV of the constant part of IV s and the set J are determined simultaneously before implementing the attack.

Restriction of $|I| + |J| < n$. The superpoly recovery has always been dominating the complexity of the cube attack, especially in [12], the attacker knows no more information except which secret key bits are involved in the superpoly. Then she/he has to first construct the whole truth table for the superpoly in the offline phase. In general, the truth-table construction requires repeating the cube summation $2^{|J|}$ times, and makes the complexity of the offline phase about $2^{|I|+|J|}$. Apparently, such an attack can only be meaningful if $|I| + |J| < n$, where n is the number of secret variables. The restriction of $|I| + |J| < n$ barricades the adversary from exploiting cubes of larger dimension or mounting more rounds (where $|J|$ may expand). This restriction can be removed if we can avoid the truth table construction in the offline phase.

1.2 Our Contributions

This paper improves the existing cube attacks by exploiting the algebraic properties of the superpoly, which include the (non-)constantness, low degree and

sparse monomial distribution properties. Inspired by the division property based cube attack work of Todo et al. in [12], we formulate all these properties in one framework by developing more precise MILP models, thus we can reduce the complexity of superpoly recovery.

This also enables us to attack more rounds, or employ even larger cubes. Similar to [12], our methods regard the cryptosystem as a non-blackbox polynomial and can be used to evaluate cubes with large dimension compared with traditional cube attack and cube tester. In the following, our contributions are summarized into five aspects.

Flag Technique for Finding Proper IV Assignments. The previous MILP model in [12] has not taken the effect of constant 0/1 bits of the constant part of IVs into account. In their model, the active bits are initialized with division property value 1 and other non-active bits are all initialized to division property value 0. The non-active bits include constant part of IVs, together with some secret key bits and state bits that are assigned statically to 0/1 according to the specification of ciphers. It has been noticed in [22] that constant 0 bits can affect the propagation of division property. But we should pay more attention to constant 1 bits since constant 0 bits can be generated in the updating functions due to the XOR of even number of constant 1's. Therefore, we propose a formal technique which we refer as the “flag” technique where the constant 0 and constant 1 as well as other non-constant MILP variables are treated properly. With this technique, we are able to find proper assignments to constant IVs (\mathbf{IV}) that makes the corresponding superpoly ($p(\mathbf{x}, \mathbf{IV})$) non-constant. With this technique, proper IVs can now be found with MILP model rather than time-consuming trial & summations in the offline phase as in [12, 22]. According to our experiments, the flag technique has a perfect 100% accuracy for finding proper non-cube IV assignments in most cases. Note that our flag technique has partially proved the availability of the two assumptions since we are able to find proper \mathbf{IV} 's in all our attacks.

Degree Evaluation for Going Beyond the $|I| + |J| < n$ Restriction. To avoid constructing the whole truth table using cube summations, we introduce a new technique that can upper bound the algebraic degree, denoted as d , of the superpoly using the MILP-aided bit-based division property. With the knowledge of its degree d (and key indices J), the superpoly can be represented with its $\binom{|J|}{\leq d}$ coefficients rather than the whole truth table, where $\binom{|J|}{\leq d}$ is defined as

$$\binom{|J|}{\leq d} := \sum_{i=0}^d \binom{|J|}{i}. \quad (1)$$

When $d = |J|$, the complexity by our new method and that by [12] are equal. For $d < |J|$, we know that the coefficients of the monomials with degree higher than d are constantly 0. The complexity of superpoly recovery can be reduced from $2^{|I|+|J|}$ to $2^{|I|} \times \binom{|J|}{\leq d}$. In fact, for some lightweight ciphers, the algebraic degrees of their round functions are quite low. Therefore, the degrees d are

often much smaller than the number of involved key bits $|J|$, especially when high-dimensional cubes are used. Since $d \ll |J|$ for all previous attacks, we can improve the complexities of previous results and use larger cubes mounting to more rounds even if $|I| + |J| \geq n$.

Precise Term Enumeration for Further Lowering Complexities. Since the superpolies are generated through iterations, the number of higher-degree monomials in the superpoly is usually much smaller than its low-degree counterpart. For example, when the degree of the superpoly is $d < |J|$, the number of d -degree monomials are usually much smaller than the upper bound $\binom{|J|}{d}$. We propose a MILP model technique for enumerating all t -degree ($t = 1, \dots, d$) monomials that may appear in the superpoly, so that the complexities of several attacks are further reduced.

Relaxed Term Enumeration. For some primitives (such as 750-round ACORN), our MILP model can only enumerate the d -degree monomials since the number of lower-degree monomials are too large to be exhausted. Alternately, for $t = 1, \dots, d - 1$, we can find a set of key indices $JR_t \subseteq J$ s.t. all t -degree monomials in the superpoly are composed of $x_j, j \in JR_t$. As long as $|JR_t| < |J|$ for some $t = 1, \dots, d - 1$, we can still reduce the complexities of superpoly recovery.

Combining the flag technique and the degree evaluation above, we are able to lower the complexities of the previous best cube attacks in [6, 12, 22]. Particularly, we can further provide key recovery results on 839-round TRIVIUM², 891-round Kreyvium, 184-round Grain-128a, and 750-round ACORN. Furthermore, the precise & relaxed term enumeration techniques allow us to lower the complexities of 833-round TRIVIUM, 849-round Kreyvium, 184-round Grain-128a and 750-round ACORN. Our concrete results are summarized in Table 1.³ In [26], Todo et al. revisit the fast correlation attack and analyze the key-stream generator (rather than the initialization) of the Grain family (Grain-128a, Grain-128, and Grain-v1). As a result, the key-stream generators of the Grain family are insecure. In other words, they can recover the internal state after initialization more efficiently than by exhaustive search. And the secret key is recovered from the internal state because the initialization is a public permutation. To the best of our knowledge, all our results of Kreyvium, Grain-128a, and ACORN are the current best key recovery attacks on the initialization of the targeted ciphers. However, none of our results seems to threaten the security of the ciphers.

Clique View of the Superpoly Recovery. In order to lower the complexity of the superpoly recovery, the term enumeration technique has to execute many MILP instances, which is difficult for some applications. We represent the resultant superpoly as a graph, so that we can utilize the clique concept from the

² While this paper was under submission, Fu et al. released a paper on ePrint [24] and claimed that 855 rounds initialization of TRIVIUM can be attacked.

³ Because of the page limitation, we put part of detailed applications about Kreyvium, Grain-128a and ACORN in the full version [25].

graph theory to upper bound the complexity of the superpoly recovery phase, without requiring MILP solver as highly as the term enumeration technique.

Organizations. Section 2 provides the background of cube attacks, division property, MILP model etc. Section 3 introduces our flag technique for identifying proper assignments to non-cube IVs. Section 4 details the degree evaluation technique upper bounding the algebraic degree of the superpoly. Combining the flag technique and degree evaluation, we give improved key recovery cube attacks on 4 targeted ciphers in Sect. 5. The precise & relaxed term enumeration as well as their applications are given in Sect. 6. We revisit the term enumeration technique from the clique overview in Sect. 7. Finally, we conclude in Sect. 8.

Table 1. Summary of our cube attack results

Applications	#Full rounds	#Rounds	Cube size	J	Complexity	Reference
TRIVIUM	1152	799	32†	–	Practical	[4]
		832	72	5	2^{77}	[12, 22]
		833	73	7	$2^{76.91}$	Sect. 6.1
		835	37/36*	–	2^{75}	[6]
		836	78	1	2^{79}	Sect. 5.1
		839	78	1	2^{79}	Sect. 5.1
Kreyvium	1152	849	61	23	2^{84}	[22]
		849	61	23	$2^{81.7}$	Full version [25]
		849	61	23	$2^{73.41}$	Sect. 6.2
		872	85	39	2^{124}	[22]
		872	85	39	$2^{94.61}$	Full version [25]
		891	113	20	$2^{120.73}$	Full version [25]
Grain-128a	256	177	33	–	Practical	[27]
		182	88	18	2^{106}	[12, 22]
		182	88	14	2^{102}	Full version [25]
		183	92	16	2^{108}	[12, 22]
		183	92	16	$2^{108} - 2^{96.08}$	Full version [25]
		184	95	21	$2^{109.61}$	Sect. 6.3
ACORN	1792	503	5‡	-	Practical‡	[5]
		704	64	58	2^{122}	[12, 22]
		704	64	63	$2^{77.88}$	Sect. 6.4
		750	101	81	$2^{125.71}$	Full version [25]
		750	101	81	$2^{120.92}$	Sect. 6.4

†18 cubes whose size is from 32 to 37 are used, where the most efficient cube is shown to recover one bit of the secret key.

*28 cubes of sizes 36 and 37 are used, following the correlation cube attack scenario. It requires an additional 2^{51} complexity for preprocessing.

‡The attack against 477 rounds is mainly described for the practical attack in [5]. However, when the goal is the superpoly recovery and to recover one bit of the secret key, 503 rounds are attacked.

2 Preliminaries

2.1 Mixed Integer Linear Programming

MILP is an optimization or feasibility program whose variables are restricted to integers. A MILP model \mathcal{M} consists of variables $\mathcal{M}.var$, constraints $\mathcal{M}.con$, and an objective function $\mathcal{M}.obj$. MILP models can be solved by solvers like Gurobi [17]. If there is no feasible solution at all, the solver simply returns *infeasible*. If no objective function is assigned, the MILP solver only evaluates the feasibility of the model. The application of MILP model to cryptanalysis dates back to the year 2011 [28], and has been widely used for searching characteristics corresponding to various methods such as differential [29,30], linear [30], impossible differential [31,32], zero-correlation linear [31], and integral characteristics with division property [16]. We will detail the MILP model of [16] later in this section.

2.2 Cube Attack

Considering a stream cipher with n secret key bits $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and m public initialization vector (IV) bits $\mathbf{v} = (v_1, v_2, \dots, v_m)$. Then, the first output keystream bit can be regarded as a polynomial of \mathbf{x} and \mathbf{v} referred as $f(\mathbf{x}, \mathbf{v})$. For a set of indices $I = \{i_1, i_2, \dots, i_{|I|}\} \subset \{1, 2, \dots, n\}$, which is referred as cube indices and denote by t_I the monomial as $t_I = v_{i_1} \cdots v_{i_{|I|}}$, the algebraic normal form (ANF) of $f(\mathbf{x}, \mathbf{v})$ can be uniquely decomposed as

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

where the monomials of $q(\mathbf{x}, \mathbf{v})$ miss at least one variable from $\{v_{i_1}, v_{i_2}, \dots, v_{i_{|I|}}\}$. Furthermore, $p(\mathbf{x}, \mathbf{v})$, referred as the superpoly in [1], is irrelevant to $\{v_{i_1}, v_{i_2}, \dots, v_{i_{|I|}}\}$. The value of $p(\mathbf{x}, \mathbf{v})$ can only be affected by the secret key bits \mathbf{x} and the assignment to the non-cube IV bits v_s ($s \notin I$). For a secret key \mathbf{x} and an assignment to the non-cube IVs $\mathbf{IV} \in \mathbb{F}_2^m$, we can define a structure called cube, denoted as $C_I(\mathbf{IV})$, consisting of $2^{|I|}$ 0-1 vectors as follows:

$$C_I(\mathbf{IV}) := \{\mathbf{v} \in \mathbb{F}_2^m : \mathbf{v}[i] = 0/1, i \in I \bigwedge \mathbf{v}[s] = \mathbf{IV}[s], s \notin I\}. \tag{2}$$

It has been proved by Dinur and Shamir [1] that the value of superpoly p corresponding to the key \mathbf{x} and the non-cube IV assignment \mathbf{IV} can be computed by summing over the cube $C_I(\mathbf{IV})$ as follows:

$$p(\mathbf{x}, \mathbf{IV}) = \bigoplus_{\mathbf{v} \in C_I(\mathbf{IV})} f(\mathbf{x}, \mathbf{v}). \tag{3}$$

In the remainder of this paper, we refer to the value of the superpoly corresponding to the assignment \mathbf{IV} in Eq. (3) as $p_{\mathbf{IV}}(\mathbf{x})$ for short. We use C_I as the cube corresponding to arbitrary \mathbf{IV} setting in Eq. (2). Since C_I is defined according to I , we may also refer I as the “cube” without causing ambiguities. The size of I , denoted as $|I|$, is also referred as the dimension of the cube.

Note: since the superpoly p is irrelevant to cube IVs $v_i, i \in I$, the value of $\mathbf{IV}[i], i \in I$ cannot affect the result of the summation in Eq. (3) at all. Therefore in Sect. 5, our $\mathbf{IV}[i]$'s ($i \in I$) are just assigned randomly to 0-1 values.

2.3 Bit-Based Division Property and Its MILP Representation

At 2015, the division property, a generalization of the integral property, was proposed in [13] with which better integral characteristics for word-oriented cryptographic primitives have been detected. Later, the bit-based division property was introduced in [15] so that the propagation of integral characteristics can be described in a more precise manner. The definition of the bit-based division property is as follows:

Definition 1 ((Bit-Based) Division Property). *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:*

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

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

When the basic bitwise operations COPY, XOR, AND are applied to the elements in \mathbb{X} , transformations of the division property should also be made following the propagation corresponding rules `copy`, `xor`, `and` and proved in [13,15]. Since round functions of cryptographic primitives are combinations of bitwise operations, we only need to determine the division property of the chosen plaintexts, denoted by $\mathcal{D}_{\mathbb{K}_0}^{1^n}$. Then, after r -round encryption, the division property of the output ciphertexts, denoted by $\mathcal{D}_{\mathbb{K}_r}^{1^n}$, can be deduced according to the round function and the propagation rules. More specifically, when the plaintext bits at index positions $I = \{i_1, i_2, \dots, i_{|I|}\} \subset \{1, 2, \dots, n\}$ are active (the active bits traverse all $2^{|I|}$ possible combinations while other bits are assigned to static 0/1 values), the division property of such chosen plaintexts is $\mathcal{D}_{\mathbf{k}}^{1^n}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Then, the propagation of the division property from $\mathcal{D}_{\mathbf{k}}^{1^n}$ is evaluated as

$$\{\mathbf{k}\} := \mathbb{K}_0 \rightarrow \mathbb{K}_1 \rightarrow \mathbb{K}_2 \rightarrow \dots \rightarrow \mathbb{K}_r,$$

where $\mathcal{D}_{\mathbb{K}_i}$ is the division property after i -round propagation. If the division property \mathbb{K}_r does not have a unit vector \mathbf{e}_i whose only i th element is 1, the i th bit of r -round ciphertexts is balanced.

However, when round r gets bigger, the size of \mathbb{K}_r expands exponentially towards $O(2^n)$ requiring huge memory resources. So the bit-based division property has only been applied to block ciphers with tiny block sizes, such as SIMON32 and Simeck32 [15]. This memory-crisis has been solved by Xiang et al. using the MILP modeling method.

Propagation of Division Property with MILP. At ASIACRYPT 2016, Xiang et al. first introduced a new concept *division trail* defined as follows:

Definition 2 (Division Trail [16]). *Let us consider the propagation of the division property $\{\mathbf{k}\} \stackrel{\text{def}}{=} \mathbb{K}_0 \rightarrow \mathbb{K}_1 \rightarrow \mathbb{K}_2 \rightarrow \dots \rightarrow \mathbb{K}_r$. Moreover, for any vector $\mathbf{k}_{i+1}^* \in \mathbb{K}_{i+1}$, there must exist a vector $\mathbf{k}_i^* \in \mathbb{K}_i$ such that \mathbf{k}_i^* can propagate to \mathbf{k}_{i+1}^* by the propagation rule of the division property. Furthermore, for $(\mathbf{k}_0, \mathbf{k}_1, \dots, \mathbf{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \dots \times \mathbb{K}_r)$ if \mathbf{k}_i can propagate to \mathbf{k}_{i+1} for all $i \in \{0, 1, \dots, r-1\}$, we call $(\mathbf{k}_0 \rightarrow \mathbf{k}_1 \rightarrow \dots \rightarrow \mathbf{k}_r)$ an r -round division trail.*

Let E_k be the target r -round iterated cipher. Then, if there is a division trail $\mathbf{k}_0 \xrightarrow{E_k} \mathbf{k}_r = \mathbf{e}_j$ ($j = 1, \dots, n$), the summation of j th bit of the ciphertexts is unknown; otherwise, if there is no division trail s.t. $\mathbf{k}_0 \xrightarrow{E_k} \mathbf{k}_r = \mathbf{e}_j$, we know the i th bit of the ciphertext is balanced (the summation of the i th bit is constant 0). Therefore, we have to evaluate all possible division trails to verify whether each bit of ciphertexts is balanced or not. Xiang et al. proved 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 particular output bits can be acquired by analyzing the solutions of the \mathcal{M} . After Xiang et al.'s work, some simplifications have been made to the MILP descriptions of `copy`, `xor`, `and` in [12, 18]. We present the current simplest MILP-based `copy`, `xor`, `and` as follows:

Proposition 1 (MILP Model for COPY [18]). *Let a $\xrightarrow{\text{COPY}}$ (b_1, b_2, \dots, b_m) be a division trail of COPY. The following inequalities are sufficient to describe the propagation of the division property for `copy`.*

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_1, b_2, \dots, b_m \text{ as binary.} \\ \mathcal{M}.con \leftarrow a = b_1 + b_2 + \dots + b_m \end{cases}$$

Proposition 2 (MILP Model for XOR [18]). *Let $(a_1, a_2, \dots, a_m) \xrightarrow{\text{XOR}} b$ be a division trail of XOR. The following inequalities are sufficient to describe the propagation of the division property for `xor`.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \dots, a_m, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow a_1 + a_2 + \dots + a_m = b \end{cases}$$

Proposition 3 (MILP Model for AND [12]). *Let $(a_1, a_2, \dots, a_m) \xrightarrow{\text{AND}} b$ be a division trail of AND. The following inequalities are sufficient to describe the propagation of the division property for `and`.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \dots, a_m, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow b \geq a_i \text{ for all } i \in \{1, 2, \dots, m\} \end{cases}$$

Note: Proposition 3 includes redundant propagations of the division property, but they do not affect preciseness of the obtained characteristics [12].

2.4 The Bit-Based Division Property for Cube Attack

When the number of initialization rounds is not large enough for a thorough diffusion, the superpoly $p(\mathbf{x}, \mathbf{v})$ defined in Eq. (2) may not be related to all key bits x_1, \dots, x_n corresponding to some high-dimensional cube I . Instead, there is a set of key indices $J \subseteq \{1, \dots, n\}$ s.t. for arbitrary $\mathbf{v} \in \mathbb{F}_2^m$, $p(\mathbf{x}, \mathbf{v})$ can only be related to x_j 's ($j \in J$). In CRYPTO 2017, Todo et al. proposed a method for determining such a set J using the bit-based division property [12]. They further showed that, with the knowledge of such J , cube attacks can be launched to recover some information related to the secret key bits. More specifically, they proved the following Lemma 1 and Proposition 4.

Lemma 1. *Let $f(\mathbf{x})$ be a polynomial from \mathbb{F}_2^n to \mathbb{F}_2 and $\mathbf{a}_u^f \in \mathbb{F}_2$ ($\mathbf{u} \in \mathbb{F}_2^n$) be the ANF coefficients of $f(\mathbf{x})$. Let \mathbf{k} be an n -dimensional bit vector. Assuming there is no division trail such that $\mathbf{k} \xrightarrow{f} 1$, then \mathbf{a}_u^f is always 0 for $\mathbf{u} \succeq \mathbf{k}$.*

Proposition 4. *Let $f(\mathbf{x}, \mathbf{v})$ be a polynomial, where \mathbf{x} and \mathbf{v} denote the secret and public variables, respectively. For a set of indices $I = \{i_1, i_2, \dots, i_{|I|}\} \subset \{1, 2, \dots, m\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1}, v_{i_2}, \dots, v_{i_{|I|}}\}$ are taking all possible combinations of values. Let \mathbf{k}_I be an m -dimensional bit vector such that $\mathbf{v}^{\mathbf{k}_I} = t_I = v_{i_1}v_{i_2} \cdots v_{i_{|I|}}$, i.e. $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Assuming there is no division trail such that $(\mathbf{e}_\lambda, \mathbf{k}_I) \xrightarrow{f} 1$, x_λ is not involved in the superpoly of the cube C_I .*

When f represents the first output bit after the initialization iterations, we can identify J by checking whether there is a division trial $(\mathbf{e}_\lambda, \mathbf{k}_I) \xrightarrow{f} 1$ for $\lambda = 1, \dots, n$ using the MILP modeling method introduced in Sect. 2.3. If the division trial $(\mathbf{e}_\lambda, \mathbf{k}_I) \xrightarrow{f} 1$ exists, we have $\lambda \in J$; otherwise, $\lambda \notin J$.

When J is determined, we know that for some assignment to the non-cube $\mathbf{IV} \in \mathbb{F}_2^m$, the corresponding superpoly $p_{\mathbf{IV}}(\mathbf{x})$ is not constant 0, and it is a polynomial of $x_j, j \in J$. With the knowledge of J , we recover offline the superpoly $p_{\mathbf{IV}}(\mathbf{x})$ by constructing its truth table using cube summations defined as Eq. (3). As long as $p_{\mathbf{IV}}(\mathbf{x})$ is not constant, we can go to the online phase where we sum over the cube $C_I(\mathbf{IV})$ to get the exact value of $p_{\mathbf{IV}}(\mathbf{x})$ and refer to the precomputed truth table to identify the $x_j, j \in J$ assignment candidates. We summarize the whole process as follows:

1. **Offline Phase: Superpoly Recovery.** Randomly pick an $\mathbf{IV} \in \mathbb{F}_2^m$ and prepare the cube $C_I(\mathbf{IV})$ defined as Eq. (2). For $\mathbf{x} \in \mathbb{F}_2^n$ whose $x_j, j \in J$ traverse all $2^{|J|}$ 0-1 combinations, we compute and store the value of the superpoly $p_{\mathbf{IV}}(\mathbf{x})$ as Eq. (3). The $2^{|J|}$ values compose the truth table of $p_{\mathbf{IV}}(\mathbf{x})$ and the ANF of the superpoly is determined accordingly. If $p_{\mathbf{IV}}(\mathbf{x})$ is constant, we pick another \mathbf{IV} and repeat the steps above until we find an appropriate one s.t. $p_{\mathbf{v}}(\mathbf{x})$ is not constant.
2. **Online Phase: Partial Key Recovery.** Query the cube $C_I(\mathbf{IV})$ to encryption oracle and get the summation of the $2^{|I|}$ output bits. We denoted the

summation by $\lambda \in \mathbb{F}_2$ and we know $p_{IV}(x) = \lambda$ according to Eq. (3). So we look up the truth table of the superpoly and only reserve the $x_j, j \in J$ s.t. $p_{IV}(x) = \lambda$.

3. **Brute-Force Search.** Guess the remaining secret variables to recover the entire value in secret variables.

Phase 1 dominates the time complexity since it takes $2^{|I|+|J|}$ encryptions to construct the truth table of size $2^{|J|}$. It is also possible that $p_{IV}(x)$ is constant so we have to run several different IV 's to find the one we need. The attack can only be meaningful when (1) $|I| + |J| < n$; (2) appropriate IV 's are easy to be found. The former requires the adversary to use "good" cube I 's with small J while the latter is the exact reason why Assumptions 1 and 2 are proposed [12, 22].

3 Modeling the Constant Bits to Improve the Preciseness of the MILP Model

In the initial state of stream ciphers, there are secret key bits, public modifiable IV bits and constant 0/1 bits. In the previous MILP model, the initial bit-based division properties of the cube IVs are set to 1, while those of the non-cube IVs, constant state bits or even secret key bits are all set to 0.

Obviously, when constant 0 bits are involved in multiplication operations, it always results in an constant 0 output. But, as is pointed out in [22], such a phenomenon cannot be reflected in previous MILP model method. In the previous MILP model, the widely used COPY+AND operation:

$$\text{COPY+AND} : (s_1, s_2) \rightarrow (s_1, s_2, s_1 \wedge s_2). \tag{4}$$

can result in division trials $(x_1, x_2) \xrightarrow{\text{COPY+AND}} (y_1, y_2, a)$ as follows:

$$\begin{aligned} (1, 0) &\xrightarrow{\text{COPY+AND}} (0, 0, 1), \\ (0, 1) &\xrightarrow{\text{COPY+AND}} (0, 0, 1). \end{aligned}$$

Assuming that either s_1 or s_2 of Eq. (4) is a constant 0 bit, $(s_1 \wedge s_2)$ is always 0. In this occasion, the division property of $(s_1 \wedge s_2)$ must be 0 which is overlooked by the previous MILP model. To prohibit the propagation above, an additional constraint $\mathcal{M}.con \leftarrow a = 0$ should be added when either s_1 or s_2 is constant 0.

In [22], the authors only consider the constant 0 bits. They thought the model can be precise enough when all the state bits initialized to constant 0 bits are handled. But in fact, although constant 1 bits do not affect the division property propagation, we should still be aware because 0 bits might be generated when even number of constant 1 bits are XORed during the updating process. This is shown in Example 2 for Kreyvium in Appendix A [25].

Therefore, for all variables in the MILP $v \in \mathcal{M}.var$, we give them an additional flag $v.F \in \{1_c, 0_c, \delta\}$ where 1_c means the bit is constant 1, 0_c means

constant 0 and δ means variable. Apparently, when $v.F = 0_c/1_c$, there is always a constraint $v = 0 \in \mathcal{M}.con$. We define $=$, \oplus and \times operations for the elements of set $\{1_c, 0_c, \delta\}$. The $=$ operation tests whether two elements are equal (naturally $1_c = 1_c$, $0_c = 0_c$ and $\delta = \delta$). The \oplus operation follows the rules:

$$\begin{cases} 1_c \oplus 1_c = 0_c \\ 0_c \oplus x = x \oplus 0_c = x \text{ for arbitrary } x \in \{1_c, 0_c, \delta\} \\ \delta \oplus x = x \oplus \delta = \delta \end{cases} \quad (5)$$

The \times operation follows the rules:

$$\begin{cases} 1_c \times x = x \times 1_c = x \\ 0_c \times x = x \times 0_c = 0_c \text{ for arbitrary } x \in \{1_c, 0_c, \delta\} \\ \delta \times \delta = \delta \end{cases} \quad (6)$$

Therefore, in the remainder of this paper, the MILP models for COPY, XOR and AND should also consider the effects of flags. So the previous `copy`, `xor`, and `and` should now add the assignment to flags. We denote the modified versions as `copyf`, `xorf`, and `andf` and define them as Propositions 5, 6 and 7 as follows.

Proposition 5 (MILP Model for COPY with Flag). *Let a \xrightarrow{COPY} (b_1, b_2, \dots, b_m) be a division trail of COPY. The following inequalities are sufficient to describe the propagation of the division property for `copyf`.*

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_1, b_2, \dots, b_m \text{ as binary.} \\ \mathcal{M}.con \leftarrow a = b_1 + b_2 + \dots + b_m \\ a.F = b_1.F = \dots = b_m.F \end{cases}$$

We denote this process as $(\mathcal{M}, b_1, \dots, b_m) \leftarrow \text{copyf}(\mathcal{M}, a, m)$.

Proposition 6 (MILP Model for XOR with Flag). *Let $(a_1, a_2, \dots, a_m) \xrightarrow{XOR} b$ be a division trail of XOR. The following inequalities are sufficient to describe the propagation of the division property for `xorf`.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \dots, a_m, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow a_1 + a_2 + \dots + a_m = b \\ b.F = a_1.F \oplus a_2.F \oplus \dots \oplus a_m.F \end{cases}$$

We denote this process as $(\mathcal{M}, b) \leftarrow \text{xorf}(\mathcal{M}, a_1, \dots, a_m)$.

Proposition 7 (MILP Model for AND with Flag). *Let $(a_1, a_2, \dots, a_m) \xrightarrow{AND} b$ be a division trail of AND. The following inequalities are sufficient to describe the propagation of the division property for `andf`.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \dots, a_m, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow b \geq a_i \text{ for all } i \in \{1, 2, \dots, m\} \\ b.F = a_1.F \times a_2.F \times \dots \times a_m.F \\ \mathcal{M}.con \leftarrow b = 0 \text{ if } b.F = 0_c \end{cases}$$

We denote this process as $(\mathcal{M}, b) \leftarrow \text{andf}(\mathcal{M}, a_1, \dots, a_m)$.

Algorithm 1. Evaluate secret variables by MILP with Flags

```

1: procedure attackFramework(Cube indices  $I$ , specific assignment to non-cube IVs
    $\mathbf{IV}$  or  $\mathbf{IV} = \text{NULL}$ )
2:   Declare an empty MILP model  $\mathcal{M}$ 
3:   Declare  $\mathbf{x}$  as  $n$  MILP variables of  $\mathcal{M}$  corresponding to secret variables.
4:   Declare  $\mathbf{v}$  as  $m$  MILP variables of  $\mathcal{M}$  corresponding to public variables.
5:    $\mathcal{M}.con \leftarrow v_i = 1$  and assign  $v_i.F = \delta$  for all  $i \in I$ 
6:    $\mathcal{M}.con \leftarrow v_i = 0$  for all  $i \in (\{1, 2, \dots, n\} - I)$ 
7:    $\mathcal{M}.con \leftarrow \sum_{i=1}^n x_i = 1$  and assign  $x_i.F = \delta$  for all  $i \in \{1, \dots, n\}$ 
8:   if  $\mathbf{IV} = \text{NULL}$  then
9:      $v_i.F = \delta$  for all  $i \in (\{1, 2, \dots, m\} - I)$ 
10:  else
11:    Assign the flags of  $v_i, i \in (\{1, 2, \dots, m\} - I)$  as:

$$v_i.F = \begin{cases} 1_c & \text{if } \mathbf{IV}[i] = 1 \\ 0_c & \text{if } \mathbf{IV}[i] = 0 \end{cases}$$

12:  end if
13:  Update  $\mathcal{M}$  according to round functions and output functions
14:  do
15:    solve MILP model  $\mathcal{M}$ 
16:    if  $\mathcal{M}$  is feasible then
17:      pick index  $j \in \{1, 2, \dots, n\}$  s.t.  $x_j = 1$ 
18:       $J = J \cup \{j\}$ 
19:       $\mathcal{M}.con \leftarrow x_j = 0$ 
20:    end if
21:  while  $\mathcal{M}$  is feasible
22:  return  $J$ 
23: end procedure

```

With these modifications, we are able to improve the preciseness of the MILP model. The improved attack framework can be written as Algorithm 1. It enables us to identify the involved keys when the non-cube IVs are set to specific constant 0/1 values by imposing corresponding flags to the non-cube MILP binary variables. With this method, we can determine an $\mathbf{IV} \in \mathbb{F}_2^m$ s.t. the corresponding superpoly $p_{\mathbf{IV}}(\mathbf{x}) \neq 0$.

4 Upper Bounding the Degree of the Superpoly

For an $\mathbf{IV} \in \mathbb{F}_2^m$ s.t. $p_{\mathbf{IV}}(\mathbf{x}) \neq 0$, the ANF of $p_{\mathbf{IV}}(\mathbf{x})$ can be represented as

$$p_{\mathbf{IV}}(\mathbf{x}) = \sum_{\mathbf{u} \in \mathbb{F}_2^m} a_{\mathbf{u}} \mathbf{x}^{\mathbf{u}} \tag{7}$$

where $a_{\mathbf{u}}$ is determined by the values of the non-cube IVs. If the degree of the superpoly is upper bounded by d , then for all \mathbf{u} 's with Hamming weight satisfying $hw(\mathbf{u}) > d$, we constantly have $a_{\mathbf{u}} = 0$. In this case, we no longer have

to build the whole truth table to recover the superpoly. Instead, we only need to determine the coefficients a_u for $hw(\mathbf{u}) \leq d$. Therefore, we select $\sum_{i=0}^d \binom{|J|}{i}$ different \mathbf{x} 's and construct a linear system with $\left(\sum_{i=0}^d \binom{|J|}{i}\right)$ variables and the coefficients as well as the whole ANF of $p_{IV}(\mathbf{x})$ can be recovered by solving such a linear system. So the complexity of Phase 1 can be reduced from $2^{|I|+|J|}$ to $2^{|I|} \times \sum_{i=0}^d \binom{|J|}{i}$. For the simplicity of notations, we denote the summation $\sum_{i=0}^d \binom{|J|}{i}$ as $\binom{|J|}{\leq d}$ in the remainder of this paper. With the knowledge of the involved key indices $J = \{j_1, j_2, \dots, j_{|J|}\}$ and the degree of the superpoly $d = \deg p_{IV}(\mathbf{x})$, the attack procedure can be adapted as follows:

1. **Offline Phase: Superpoly Recovery.** For all $\binom{|J|}{\leq d}$ \mathbf{x} 's satisfying $hw(\mathbf{x}) \leq d$ and $\bigoplus_{j \in J} e_j \succeq \mathbf{x}$, compute the values of the superpolys as $p_{IV}(\mathbf{x})$ by summing over the cube $C_I(\mathbf{IV})$ as Eq. (3) and generate a linear system of the $\binom{|J|}{\leq d}$ coefficients a_u ($hw(\mathbf{u}) \leq d$). Solve the linear system, determine the coefficient a_u of the $\binom{|J|}{\leq d}$ terms and store them in a lookup table T . The ANF of the $p_{IV}(\mathbf{x})$ can be determined with the lookup table.
2. **Online Phase: Partial Key Recovery.** Query the encryption oracle and sum over the cube $C_I(\mathbf{IV})$ as Eq. (3) and acquire the exact value of $p_{IV}(\mathbf{x})$. For each of the $2^{|J|}$ possible values of $\{x_{j_1}, \dots, x_{j_{|J|}}\}$, compute the values of the superpoly as Eq. (7) (the coefficient a_u are acquired by looking up the precomputed table T) and identify the correct key candidates.
3. **Brute-force search phase.** Attackers guess the remaining secret variables to recover the entire value in secret variables.

The complexity of Phase 1 becomes $2^{|I|} \times \binom{|J|}{\leq d}$. Phase 2 now requires $2^{|I|}$ encryptions and $2^{|J|} \times \binom{|J|}{\leq d}$ table lookups, so the complexity can be regarded as $2^{|I|} + 2^{|J|} \times \binom{|J|}{\leq d}$. The complexity of Phase 3 remains 2^{n-1} . Therefore, the number of encryptions a feasible attack requires is

$$\max \left\{ 2^{|I|} \times \binom{|J|}{\leq d}, 2^{|I|} + 2^{|J|} \times \binom{|J|}{\leq d} \right\} < 2^n. \tag{8}$$

The previous limitation of $|I| + |J| < n$ is removed.

The knowledge of the algebraic degree of superpolys can largely benefit the efficiency of the cube attack. Therefore, we show how to estimate the algebraic degree of superpolys using the division property. Before the introduction of the method, we generalize Proposition 4 as follows.

Proposition 8. *Let $f(\mathbf{x}, \mathbf{v})$ be a polynomial, where \mathbf{x} and \mathbf{v} denote the secret and public variables, respectively. For a set of indices $I = \{i_1, i_2, \dots, i_{|I|}\} \subset \{1, 2, \dots, m\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1}, v_{i_2}, \dots, v_{i_{|I|}}\}$ are taking all possible combinations of values. Let \mathbf{k}_I be an m -dimensional bit vector such that $\mathbf{v}^{\mathbf{k}_I} = t_I = v_{i_1}v_{i_2} \dots v_{i_{|I|}}$. Let \mathbf{k}_A be an n -dimensional bit vector. Assuming there is no division trail such that $(\mathbf{k}_A || \mathbf{k}_I) \xrightarrow{f} 1$, the monomial $\mathbf{x}^{\mathbf{k}_A}$ is not involved in the superpoly of the cube C_I .*

Proof. The ANF of $f(\mathbf{x}, \mathbf{v})$ is represented as follows

$$f(\mathbf{x}, \mathbf{v}) = \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m}} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{\mathbf{u}},$$

where $a_{\mathbf{u}}^f \in \mathbb{F}_2$ denotes the ANF coefficients. The polynomial $f(\mathbf{x}, \mathbf{v})$ is decomposed into

$$\begin{aligned} f(\mathbf{x}, \mathbf{v}) &= \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m} | \mathbf{u} \succeq (\mathbf{0} \parallel \mathbf{k}_I)} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{\mathbf{u}} \oplus \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m} | \mathbf{u} \not\succeq (\mathbf{0} \parallel \mathbf{k}_I)} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{\mathbf{u}} \\ &= t_I \cdot \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m} | \mathbf{u} \succeq (\mathbf{0} \parallel \mathbf{k}_I)} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{\mathbf{u} \oplus (\mathbf{0} \parallel \mathbf{k}_I)} \oplus \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m} | \mathbf{u} \not\succeq (\mathbf{0} \parallel \mathbf{k}_I)} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{(\mathbf{0} \parallel \mathbf{u})} \\ &= t_I \cdot p(\mathbf{x}, \mathbf{v}) \oplus q(\mathbf{x}, \mathbf{v}). \end{aligned}$$

Therefore, the superpoly $p(\mathbf{x}, \mathbf{v})$ is represented as

$$p(\mathbf{x}, \mathbf{v}) = \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m} | \mathbf{u} \succeq (\mathbf{0} \parallel \mathbf{k}_I)} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{\mathbf{u} \oplus (\mathbf{0} \parallel \mathbf{k}_I)}.$$

Since there is no division trail $(\mathbf{k}_A \parallel \mathbf{k}_I) \xrightarrow{f} 1$, $a_{\mathbf{u}}^f = 0$ for $\mathbf{u} \succeq (\mathbf{k}_A \parallel \mathbf{k}_I)$ because of Lemma 1. Therefore,

$$p(\mathbf{x}, \mathbf{v}) = \bigoplus_{\mathbf{u} \in \mathbb{F}_2^{n+m} | \mathbf{u} \succeq (\mathbf{0} \parallel \mathbf{k}_I), \mathbf{u}^{k_A} \parallel \mathbf{0} = 0} a_{\mathbf{u}}^f \cdot (\mathbf{x} \parallel \mathbf{v})^{\mathbf{u} \oplus (\mathbf{0} \parallel \mathbf{k}_I)}.$$

This superpoly is independent of the monomial \mathbf{x}^{k_A} since $\mathbf{u}^{k_A} \parallel \mathbf{0}$ is always 0. \square

According to Proposition 8, the existence of the division trail $(\mathbf{k}_A \parallel \mathbf{k}_I) \xrightarrow{f} 1$ is in accordance with the existence of the monomial \mathbf{x}^{k_A} in the superpoly of the cube C_I .

If there is $d \geq 0$ s.t. for all \mathbf{k}_A of hamming weight $hw(\mathbf{k}_A) > d$, the division trail \mathbf{x}^{k_A} does not exist, then we know that the algebraic degree of the superpoly is bounded by d . Using MILP, this d can be naturally modeled as the maximum of the objective function $\sum_{j=1}^n x_j$. With the MILP model \mathcal{M} and the cube indices I , we can bound the degree of the superpoly using Algorithm 2. Same with Algorithm 1, we can also consider the degree of the superpoly for specific assignment to the non-cube IVs. So we also add the input \mathbf{IV} that can either be a specific assignment or a NULL referring to arbitrary assignment. The solution $\mathcal{M}.obj = d$ is the upper bound of the superpoly’s algebraic degree. Furthermore, corresponding to $\mathcal{M}.obj = d$ and according to the definition of $\mathcal{M}.obj$, there should also be a set of indices $\{l_1, \dots, l_d\}$ s.t. the variables representing the initially declared \mathbf{x} (representing the division property of the key bits) satisfy the constraints $x_{l_1} = \dots = x_{l_d} = 1$. We can also enumerate all t -degree ($1 \leq t \leq d$) monomials involved in the superpoly using a similar technique which we will detail later in Sect. 6.

Algorithm 2. Evaluate upper bound of algebraic degree on the superpoly

```

1: procedure DegEval(Cube indices  $I$ , specific assignment to non-cube IVs  $IV$  or
    $IV = \text{NULL}$ )
2:   Declare an empty MILP model  $\mathcal{M}$ .
3:   Declare  $\mathbf{x}$  be  $n$  MILP variables of  $\mathcal{M}$  corresponding to secret variables.
4:   Declare  $\mathbf{v}$  be  $m$  MILP variables of  $\mathcal{M}$  corresponding to public variables.
5:    $\mathcal{M}.con \leftarrow v_i = 1$  and assign the flags  $v_i.F = \delta$  for all  $i \in I$ 
6:    $\mathcal{M}.con \leftarrow v_i = 0$  for  $i \in (\{1, \dots, n\} - I)$ 
7:   if  $IV = \text{NULL}$  then
8:     Assign the flags  $v_i.F = \delta$  for  $i \in (\{1, \dots, n\} - I)$ 
9:   else
10:    Assign the flags of  $v_i$ ,  $i \in (\{1, 2, \dots, n\} - I)$  as:

$$v_i.F = \begin{cases} 1_c & \text{if } IV[i] = 1 \\ 0_c & \text{if } IV[i] = 0 \end{cases}$$

11:   end if
12:   Set the objective function  $\mathcal{M}.obj \leftarrow \sum_{i=1}^n x_i$ 
13:   Update  $\mathcal{M}$  according to round functions and output functions
14:   Solve MILP model  $\mathcal{M}$ 
15:   return The solution of  $\mathcal{M}$ .
16: end procedure

```

5 Applications of Flag Technique and Degree Evaluation

We apply our method to 4 NLFSR-based ciphers namely TRIVIUM, Kreyvium, Grain-128a and ACORN. Among them, TRIVIUM, Grain-128a and ACORN are also targets of [12]. Using our new techniques, we can both lower the complexities of previous attacks and give new cubes that mount to more rounds. We give details of the application to TRIVIUM in this section, and the applications to Kreyvium, Grain-128a and ACORN in our full version [25].

5.1 Specification of Trivium

TRIVIUM is an NLFSR-based stream cipher, and the internal state is represented by 288-bit state $(s_1, s_2, \dots, s_{288})$. Figure 1 shows the state update function of TRIVIUM. The 80-bit key is loaded to the first register, and the 80-bit IV is loaded to the second register. The other state bits are set to 0 except the least three bits in the third register. Namely, the initial state bits are represented as

$$\begin{aligned}
 (s_1, s_2, \dots, s_{93}) &= (K_1, K_2, \dots, K_{80}, 0, \dots, 0), \\
 (s_{94}, s_{95}, \dots, s_{177}) &= (IV_1, IV_2, \dots, IV_{80}, 0, \dots, 0), \\
 (s_{178}, s_{279}, \dots, s_{288}) &= (0, 0, \dots, 0, 1, 1, 1).
 \end{aligned}$$

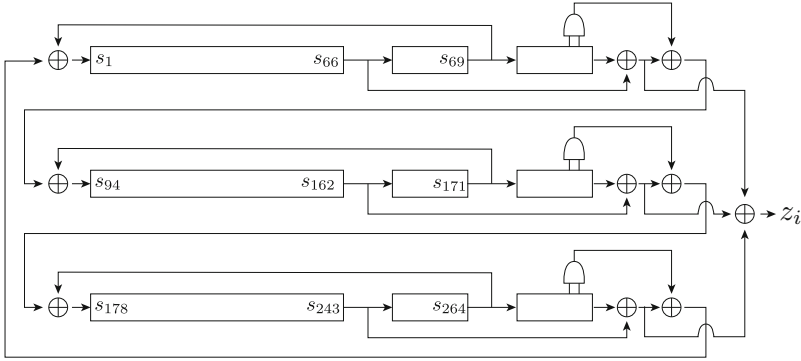


Figure 1. Structure of TRIVIUM

The pseudo code of the update function is given as follows.

$$\begin{aligned}
 t_1 &\leftarrow s_{66} \oplus s_{93} \\
 t_2 &\leftarrow s_{162} \oplus s_{177} \\
 t_3 &\leftarrow s_{243} \oplus s_{288} \\
 z &\leftarrow t_1 \oplus t_2 \oplus t_3 \\
 t_1 &\leftarrow t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171} \\
 t_2 &\leftarrow t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264} \\
 t_3 &\leftarrow t_3 \oplus s_{286} \cdot s_{287} \oplus s_{69} \\
 (s_1, s_2, \dots, s_{93}) &\leftarrow (t_3, s_1, \dots, s_{92}) \\
 (s_{94}, s_{95}, \dots, s_{177}) &\leftarrow (t_1, s_{94}, \dots, s_{176}) \\
 (s_{178}, s_{279}, \dots, s_{288}) &\leftarrow (t_2, s_{178}, \dots, s_{287})
 \end{aligned}$$

Here z denotes the 1-bit key stream. First, in the key initialization, the state is updated $4 \times 288 = 1152$ times without producing an output. After the key initialization, one bit key stream is produced by every update function.

5.2 MILP Model of Trivium

The only non-linear component of TRIVIUM is a 2-degree core function denoted as f_{core} that takes as input a 288-bit state \mathbf{s} and 5 indices i_1, \dots, i_5 , and outputs a new 288-bit state $\mathbf{s}' \leftarrow f_{core}(\mathbf{s}, i_1, \dots, i_5)$ where

$$s'_i = \begin{cases} s_{i_1} s_{i_2} + s_{i_3} + s_{i_4} + s_{i_5}, & i = i_5 \\ s_i, & \text{otherwise} \end{cases} \tag{9}$$

The division property propagation for the core function can be represented as Algorithm 3. The input of Algorithm 3 consists of \mathcal{M} as the current MILP model, a vector of 288 binary variables \mathbf{x} describing the current division property of the

288-bit NFSR state, and 5 indices i_1, i_2, i_3, i_4, i_5 corresponding to the input bits. Then Algorithm 3 outputs the updated model \mathcal{M} , and a 288-entry vector \mathbf{y} describing the division property after f_{core} .

Algorithm 3. MILP model of division property for the core function (Eq. (9))

```

1: procedure Core( $\mathcal{M}, \mathbf{x}, i_1, i_2, i_3, i_4, i_5$ )
2:   ( $\mathcal{M}, y_{i_1}, z_1$ )  $\leftarrow$  copyf( $\mathcal{M}, x_{i_1}$ )
3:   ( $\mathcal{M}, y_{i_2}, z_2$ )  $\leftarrow$  copyf( $\mathcal{M}, x_{i_2}$ )
4:   ( $\mathcal{M}, y_{i_3}, z_3$ )  $\leftarrow$  copyf( $\mathcal{M}, x_{i_3}$ )
5:   ( $\mathcal{M}, y_{i_4}, z_4$ )  $\leftarrow$  copyf( $\mathcal{M}, x_{i_4}$ )
6:   ( $\mathcal{M}, a$ )  $\leftarrow$  andf( $\mathcal{M}, z_1, z_2$ )
7:   ( $\mathcal{M}, y_{i_5}$ )  $\leftarrow$  xorf( $\mathcal{M}, a, z_2, z_3, z_4, x_{i_5}$ )
8:   for all  $i \in \{1, 2, \dots, 288\}$  w/o  $i_1, i_2, i_3, i_4, i_5$  do
9:      $y_i = x_i$ 
10:  end for
11:  return ( $\mathcal{M}, \mathbf{y}$ )
12: end procedure

```

With the definition of Core, the MILP model of R -round TRIVIUM can be described as Algorithm 4. This algorithm is a subroutine of Algorithm 1 for generating the MILP model \mathcal{M} , and the model \mathcal{M} can evaluate all division trails for TRIVIUM whose initialization rounds are reduced to R . Note that constraints to the input division property are imposed by Algorithm 1.

5.3 Experimental Verification

Identical to [12], we use the cube $I = \{1, 11, 21, 31, 41, 51, 61, 71\}$ to verify our attack and implementation. The experimental verification includes: the degree evaluation using Algorithm 2, specifying involved key bits using Algorithm 1 with $\mathbf{IV} = \text{NULL}$ or specific non-cube IV settings.

Example 1 (Verification of Our Attack against 591-round TRIVIUM). With $\mathbf{IV} = \text{NULL}$ using Algorithm 1, we are able to identify $J = \{23, 24, 25, 66, 67\}$. We know that with some assignment to the non-cube IV bits, the superpoly can be a polynomial of secret key bits $x_{23}, x_{24}, x_{25}, x_{66}, x_{67}$. These are the same with [12]. Then, we set \mathbf{IV} to random values and acquire the degree through Algorithm 2, and verify the correctness of the degree by practically recovering the corresponding superpoly.

- When we set $\mathbf{IV} = \text{0xcc2e487b, 0x78f99a93, 0xbeae}$, and run Algorithm 2, we get the degree 3. The practically recovered superpoly is also of degree 3:

$$p_v(\mathbf{x}) = x_{66}x_{23}x_{24} + x_{66}x_{25} + x_{66}x_{67} + x_{66},$$

which is in accordance with the deduction by Algorithm 2 through MILP model.

Algorithm 4. MILP model of division property for TRIVIUM

```

1: procedure TriviumEval(round  $R$ )
2:   Prepare empty MILP Model  $\mathcal{M}$ 
3:    $\mathcal{M}.var \leftarrow v_i$  for  $i \in \{1, 2, \dots, 80\}$ . ▷ Declare Public Modifiable IVs
4:    $\mathcal{M}.var \leftarrow x_i$  for  $i \in \{1, 2, \dots, 80\}$ . ▷ Declare Secret Keys
5:    $\mathcal{M}.var \leftarrow s_i^0$  for  $i \in \{1, 2, \dots, 288\}$ 
6:    $s_i^0 = x_i, s_{i+93}^0 = v_i$  for  $i = 1, \dots, 80$ .
7:    $\mathcal{M}.con \leftarrow s_i^0 = 0$  for  $i = 81, \dots, 93, 174, \dots, 288$ .
8:    $s_i^0.F = 0_c$  for  $i = 81, \dots, 285$  and  $s_j^0.F = 1_c$  for  $j = 286, 287, 288$ . ▷ Assign the
   flags for constant state bits
9:   for  $r = 1$  to  $R$  do
10:      $(\mathcal{M}, \mathbf{x}) = \text{Core}(\mathcal{M}, \mathbf{s}^{r-1}, 66, 171, 91, 92, 93)$ 
11:      $(\mathcal{M}, \mathbf{y}) = \text{Core}(\mathcal{M}, \mathbf{x}, 162, 264, 175, 176, 177)$ 
12:      $(\mathcal{M}, \mathbf{z}) = \text{Core}(\mathcal{M}, \mathbf{y}, 243, 69, 286, 287, 288)$ 
13:      $\mathbf{s}^r = \mathbf{z} \ggg 1$ 
14:   end for
15:   for all  $i \in \{1, 2, \dots, 288\}$  w/o 66, 93, 162, 177, 243, 288 do
16:      $\mathcal{M}.con \leftarrow s_i^R = 0$ 
17:   end for
18:    $\mathcal{M}.con \leftarrow (s_{66}^R + s_{93}^R + s_{162}^R + s_{177}^R + s_{243}^R + s_{288}^R) = 1$ 
19:   return  $\mathcal{M}$ 
20: end procedure

```

- When we set $\mathbf{IV} = 0x61f5972c, 0x19f5972c, 0x65c1$, the degree evaluation of Algorithm 2 is 2. The practically recovered superpoly is also of degree 2:

$$p_v(\mathbf{x}) = x_{23}x_{24} + x_{25} + x_{67} + 1.$$

- When we set $\mathbf{IV} = 0x5b942db1, 0x83ce1016, 0x6ce$, the degree is 0 and the superpoly recovered is also constant 0.

On the Accuracy of MILP Model with Flag Technique. As a comparison, we use the cube above and conduct practical experiments on different rounds namely 576, 577, 587, 590, 591 (selected from Table 2 of [22]). We try 10000 randomly chosen \mathbf{IV} 's. For each of them, we use the MILP method to evaluate the degree d , in comparison with the practically recovered ANF of the superpoly $p_{\mathbf{IV}}(\mathbf{x})$. For 576, 577, 587 and 590 rounds, the accuracy is 100%. In fact, such 100% accuracy is testified for most of our applied ciphers, which is shown in [25]. For 591-round, the accuracies are distributed as:

1. When the MILP model gives degree evaluation $d = 0$, the accuracy is 100% that the superpoly is constant 0.
2. When the MILP model gives degree evaluation $d = 3$, there is an accuracy 49% that the superpoly is a 3-degree polynomial. For the rest, the superpoly is constant 0.
3. When the MILP model gives degree evaluation $d = 2$, there is accuracy 43% that the superpoly is a 2-degree polynomial. For the rest, the superpoly is constant 0.

The ratios of error can easily be understood: for example, in some case, one key bit may multiply with constant 1 in one step $x_i \cdot 1$ and be canceled by XORing with itself in the next round, this results in a newly generated constant 0 bit $((x_i \cdot 1) \oplus x_i = 0)$. However, by the flag technique, this newly generated bit has flag value $\delta = (\delta \times 1_c) + \delta$. In our attacks, the size of cubes tends to be large, which means most of the IV bits become active, the above situation of $(x_i \cdot 1) \oplus x_i = 0$ will now become $(x_i \cdot v_j) \oplus x_i$. Therefore, when larger cubes are used, fewer constant 0/1 flags are employed, and the MILP models are becoming closer to those of $\mathbf{IV} = \mathit{NULL}$. It is predictable that the accuracy of the flag technique tends to increase when larger cubes are used. To verify this statement, we construct a 10-dimensional cube $I = \{5, 13, 18, 22, 30, 57, 60, 65, 72, 79\}$ for 591-round TRIVIUM. When $\mathbf{IV} = \mathit{NULL}$, we acquire the same upper bound of the degree $d = 3$. Then, we tried thousands of random IVs, and get an overall accuracy 80.9%. From above, we can conclude that the flag technique has high preciseness and can definitely improve the efficiency of the division property based cube attacks.

5.4 Theoretical Results

The best result in [12] mounts to 832-round TRIVIUM with cube dimension $|I| = 72$ and the superpoly involves $|J| = 5$ key bits. The complexity is 2^{77} in [12]. Using Algorithm 2, we further acquire that the degree of such a superpoly is 3. So the complexity for superpoly recovery is $2^{72} \times \binom{5}{\leq 3} = 2^{76.7}$ and the complexity for recovering the partial key is $2^{72} + 2^3 \times \binom{5}{3}$. Therefore, according to Eq. (8), the complexity of this attack is $2^{76.7}$.

We further construct a 77-dimensional cube, $I = \{1, \dots, 80\} \setminus \{5, 51, 65\}$. Its superpoly after 835 rounds of initialization only involves 1 key bit $J = \{57\}$. So the complexity of the attack is 2^{78} . Since there are only 3 non-cube IVs, we let \mathbf{IV} be all 2^3 possible non-cube IV assignments and run Algorithm 1. We find that x_{57} is involved in all of the 2^3 superpolys. So the attack is available for any of the 2^3 non-cube IV assignments. This can also be regarded as a support to the rationality of Assumption 1.

According previous results, TRIVIUM has many cubes whose superpolys only contain 1 key bit. These cubes are of great value for our key recovery attacks. Firstly, the truth table of such superpoly is balanced and the Partial Key Recovery phase can definitely recover 1 bit of secret information. Secondly, the Superpoly Recovery phase only requires $2^{|I|+1}$ and the online Partial Key Recovery only requires $2^{|I|}$ encryptions. Such an attack can be meaningful as long as $|I| + 1 < 80$, so we can try cubes having dimension as large as 78. Therefore, we investigate 78-dimensional cubes and find the best cube attack on TRIVIUM is 839 rounds. By running Algorithm 1 with $2^2 = 4$ different assignments to non-cube IVs, we know that the key bit x_{61} is involved in the superpoly for $\mathbf{IV} = 0x0, 0x4000, 0x0$ or $\mathbf{IV} = 0x0, 0x4002, 0x0$. In other words, the 47-th IV bit must be assigned to constant 1. The summary of our new results about TRIVIUM is in Table 2.

Table 2. Summary of theoretical cube attacks on TRIVIUM. The time complexity in this table shows the time complexity of Superpoly Recovery (Phase 1) and Partial Key Recovery (Phase 2).

#Rounds	$ I $	Degree	Involved keys J	Time complexity
832	72†	3	34, 58, 59, 60, 61 ($ J = 5$)	$2^{76.7}$
833	73‡	3	49, 58, 60, 74, 75, 76 ($ J = 7$)	2^{79}
833	74*	1	60 ($ J = 1$)	2^{75}
835	77★	1	57 ($ J = 1$)	2^{78}
836	78◦	1	57 ($ J = 1$)	2^{79}
839	78●	1	61 ($ J = 1$)	2^{79}

- †: $I = \{1, 2, \dots, 65, 67, 69, \dots, 79\}$
- ‡: $I = \{1, 2, \dots, 67, 69, 71, \dots, 79\}$
- *: $I = \{1, 2, \dots, 69, 71, 73, \dots, 79\}$
- ★: $I = \{1, 2, 3, 4, 6, 7, \dots, 50, 52, 53, \dots, 64, 66, 67, \dots, 80\}$
- : $I = \{1, \dots, 11, 13, \dots, 42, 44, \dots, 80\}$
- : $I = \{1, \dots, 33, 35, \dots, 46, 48, \dots, 80\}$ and $IV[47] = 1$

6 Lower Complexity with Term Enumeration

In this section, we show how to further lower the complexity of recovering the superpoly (Phase 1) in Sect. 4.

With cube indices I , key bits J and degree d , the complexity of the current superpoly recovery is $2^I \times \binom{|J|}{\leq d}$, where $\binom{|J|}{\leq d}$ corresponds to all 0-, 1-..., d -degree monomials. When $d \leq |J|/2$ (which is true in most of our applications), we constantly have $\binom{|J|}{0} \leq \dots \leq \binom{|J|}{d}$. But in practice, high-degree terms are generated in later iterations and the high-degree monomials should be fewer than their low-degree counterparts. Therefore, for all $\binom{|J|}{i}$ monomials, only very few of them may appear in the superpoly. Similar to Algorithm 1 that decides all key bits appear in the superpoly, we propose Algorithm 5 that enumerates all t -degree monomials that may appear in the superpoly. Apparently, when we use $t = 1$, we can get $J_1 = J$, the same output as Algorithm 1 containing all involved keys. If we use $t = 2, 3, \dots, d$, we get J_2, \dots, J_d that contains all possible monomials of degrees $2, 3, \dots, d$. Therefore, we only need to determine $1 + |J_1| + |J_2| + \dots + |J_d|$ coefficients in order to recover the superpoly and apparently, $|J_t| \leq \binom{|J|}{t}$ for $t = 1, \dots, d$. With the knowledge of $J_t, t = 1, \dots, d$, the complexity for Superpoly Recovery (Phase 1) has now become

$$2^{|I|} \times \left(1 + \sum_{t=1}^d |J_t|\right) \leq 2^{|I|} \times \binom{|J|}{\leq d}. \tag{10}$$

And the size of the lookup table has also reduced to $(1 + \sum_{t=1}^d |J_t|)$. So the complexity of the attack is now

$$\max\{2^{|I|} \times (1 + \sum_{t=1}^d |J_t|), 2^{|I|} + 2^{|J|} \times (1 + \sum_{t=1}^d |J_t|)\}. \quad (11)$$

Furthermore, since high-degree monomials are harder to be generated through iterations than low-degree ones, we can often find $|J_i| < \binom{|J|}{i}$ when i approaches d . So the complexity for superpoly recovery has been reduced.

Note: J_t 's ($t = 1, \dots, d$) can be generated by `TermEnum` of Algorithm 5 and they satisfy the following Property 1. This property is equivalent to the ‘‘Embed Property’’ given in [19].

Property 1. For $t = 2, \dots, d$, if there is $T = (i_1, i_2, \dots, i_t) \in J_t$ and $T' = (i_{s_1}, \dots, i_{s_l})$ ($l < t$) is a subsequence of T ($1 \leq s_1 < \dots < s_l \leq t$). Then, we constantly have $T' \in J_l$.

Before proving Property 1, we first prove the following Lemma 2.

Lemma 2. *If $\mathbf{k} \succeq \mathbf{k}'$ and there is division trial $\mathbf{k} \xrightarrow{f} \mathbf{l}$, then there is also division trial $\mathbf{k}' \xrightarrow{f} \mathbf{l}'$ s.t. $\mathbf{l} \succeq \mathbf{l}'$.*

Proof. Since f is a combination of COPY, AND and XOR operations, and the proofs when f equals to each of them are similar, we only give a proof of the case when f equals to COPY. Let $f : (*, \dots, *, x) \xrightarrow{COPY} (*, \dots, *, x, x)$.

First assume the input division property be $\mathbf{k} = (\mathbf{k}_1, 0)$, since $\mathbf{k} \succeq \mathbf{k}'$, there must be $\mathbf{k}' = (\mathbf{k}'_1, 0)$ and $\mathbf{k}_1 \succeq \mathbf{k}'_1$. We have $\mathbf{l} = \mathbf{k}$, $\mathbf{l}' = \mathbf{k}'$, thus the property holds.

When the input division property is $\mathbf{k} = (\mathbf{k}_1, 1)$, we know that the output division property can be $\mathbf{l} \in \{(\mathbf{k}_1, 0, 1), (\mathbf{k}_1, 1, 0)\}$. Since $\mathbf{k} \succeq \mathbf{k}'$, we know $\mathbf{k}' = (\mathbf{k}'_1, 1)$ or $\mathbf{k}' = (\mathbf{k}'_1, 0)$, and $\mathbf{k}_1 \succeq \mathbf{k}'_1$. When $\mathbf{k}' = (\mathbf{k}'_1, 0)$, then $\mathbf{l}' = \mathbf{k}' = (\mathbf{k}'_1, 0)$, the relation holds. When $\mathbf{k}' = (\mathbf{k}'_1, 1)$, we know $\mathbf{l}' \in \{(\mathbf{k}'_1, 0, 1), (\mathbf{k}'_1, 1, 0)\}$, the relation still holds. \square

Now we are ready to prove Property 1.

Proof. Let $\mathbf{k}, \mathbf{k}' \in \mathbb{F}_2^n$ satisfy $k_i = 1$ for $i \in T$ and $k_i = 0$ otherwise; $k'_i = 1$ for $i \in T'$ and $k'_i = 0$ otherwise. Since $T \in J_t$, we know that there is division trial $(\mathbf{k}, \mathbf{k}_I) \xrightarrow{R\text{-Rounds}} (\mathbf{0}, 1)$. Since $\mathbf{k} \succeq \mathbf{k}'$, we have $(\mathbf{k}, \mathbf{k}_I) \succeq (\mathbf{k}', \mathbf{k}_I)$ and according to Lemma 2, there is division trial s.t. $(\mathbf{k}', \mathbf{k}_I) \xrightarrow{R\text{-Rounds}} (\mathbf{0}^{m+n}, s)$ where $(\mathbf{0}^{m+n}, 1) \succeq (\mathbf{0}^{m+n}, s)$. Since the hamming weight of $(\mathbf{k}', \mathbf{k}_I)$ is larger than 0 and there is no combination of COPY, AND and XOR that makes non-zero division property to all-zero division property. So we have $s = 1$ and there exist division trial $(\mathbf{k}', \mathbf{k}_I) \xrightarrow{R\text{-Rounds}} (\mathbf{0}, 1)$. \square

Property 1 reveals a limitation of Algorithm 5. Assume the superpoly is

$$p_v(x_1, x_2, x_3, x_4) = x_1x_2x_3 + x_1x_4.$$

We can acquire $J_3 = \{(1, 2, 3)\}$ by running `TermEnum` of Algorithm 5. But, if we run `TermEnum` with $t = 2$, we will not acquire just $J_2 = \{(1, 4)\}$ but $J_2 = \{(1, 4), (1, 2), (1, 3), (2, 3)\}$ due to $(1, 2, 3) \in J_3$ and $(1, 2), (1, 3), (2, 3)$ are its subsequences. Although there are still redundant terms, the reduction from $\binom{|J|}{d}$ to $|J_d|$ is usually huge enough to improve the existing cube attack results.

Applying such term enumeration technique, we are able to lower complexities of many existing attacks namely: 832-, 833-round `TRIVIUM`, 849-round `Kreyvium`, 184-round `Grain-128a` and 704-round `ACORN`. The attack on 750-round `ACORN` can also be improved using a relaxed version of `TermEnum` which is presented as `RTermEnum` on the righthand side of Algorithm 5. In the relaxed algorithm, `RTermEnum` is acquired from `TermEnum` by replacing some states which are marked in red in Algorithm 5, and we state details later in Sect. 6.4.

6.1 Application to Trivium

As can be seen in Table 2, the attack on 832-round `TRIVIUM` has $J = J_1 = 5$ and degree $d = 3$, so we have $\binom{5}{\leq 3} = 26$ using previous technique. But by running Algorithm 5, we find that $|J_2| = 5, |J_3| = 1$, so we have $1 + \sum_{t=1}^3 |J_t| = 12 < \binom{5}{\leq 3} = 26$. Therefore, the complexity has now been reduced from $2^{76.7}$ to $2^{75.8}$. Similar technique can also be applied to the 73 dimensional cube of Table 2. Details are shown in Table 3.

Table 3. Results of `TRIVIUM` with Precise Term Enumeration

#Rounds	$ I $	$ J_1 $	$ J_2 $	$ J_3 $	$ J_4 $	$ J_5 $	$ J_t , t \geq 6$	$1 + \sum_{t=1}^d J_t $	Previous	Improved
832	72	5	5	1	0	0	0	$12 \approx 2^{3.58}$	$2^{76.7}$	$2^{75.58}$
833	73	7	6	1	0	0	0	$15 \approx 2^{3.91}$	2^{79}	$2^{76.91}$

6.2 Applications to Kreyvium

We revisit the 61-dimensional cube first given in [23] and transformed to a key recovery attack on 849-round `Kreyvium` in [22]. The degree of the superpoly is 9, so the complexity is given as $2^{81.7}$ in Appendix A of [25]. Since $J = J_1$ is of size 23, we enumerate all the terms of degree 2–9 and acquire the sets J_2, \dots, J_9 . $1 + \sum_{t=1}^d |J_t| = 5452 \approx 2^{12.41}$. So the complexity is now lowered to $2^{73.41}$. The details are listed in Table 4.

Table 4. Results of `Kreyvium` with Precise Term Enumeration

#Rounds	$ I $	$ J_1 $	$ J_2 $	$ J_3 $	$ J_4 $	$ J_5 $	$ J_6 $	$ J_7 $	$ J_8 $	$ J_9 $	$1 + \sum_{t=1}^d J_t $	Previous	Improved
849	61	23	158	555	1162	1518	1235	618	156	26	$5452 \approx 2^{12.41}$	$2^{81.7}$	$2^{73.41}$

Algorithm 5. Enumerate all the terms of degree t

<pre> 1: procedure TermEnum(Cube indices I, specific assignment to non-cube IVs \mathbf{IV} or $\mathbf{IV} = \text{NULL}$, targeted degree t) 2: Declare an empty MILP model \mathcal{M} and an empty set $J_t = \phi \subseteq \{1, \dots, n\}^n$ 3: Declare \mathbf{x} as n MILP variables of \mathcal{M} corresponding to secret variables. 4: Declare \mathbf{v} as m MILP variables of \mathcal{M} corresponding to public variables. 5: $\mathcal{M}.con \leftarrow v_i = 1$ and assign $v_i.F = \delta$ for all $i \in I$ 6: $\mathcal{M}.con \leftarrow v_i = 0$ for all $i \in$ $(\{1, 2, \dots, n\} - I)$ 7: $\mathcal{M}.con \leftarrow \sum_{i=1}^n x_i = t$ and assign $x_i.F = \delta$ for all $i \in \{1, \dots, n\}$ 8: if $\mathbf{IV} = \text{NULL}$ then 9: $v_i.F = \delta$ for all $i \in$ $(\{1, 2, \dots, n\} - I)$ 10: else 11: Assign the flags of v_i, $i \in$ $(\{1, 2, \dots, n\} - I)$ as: </pre> $v_i.F = \begin{cases} 1_c & \text{if } \mathbf{IV}[i] = 1 \\ 0_c & \text{if } \mathbf{IV}[i] = 0 \end{cases}$ <pre> 12: end if 13: Update \mathcal{M} according to round functions and output functions 14: do 15: solve MILP model \mathcal{M} 16: if \mathcal{M} is feasible then 17: pick index sequence $(j_1, \dots, j_t) \subseteq \{1, \dots, n\}^t$ s.t. $x_{j_1} = \dots = x_{j_t} = 1$ 18: $J_t = J_t \cup \{(j_1, \dots, j_t)\}$ 19: $\mathcal{M}.con \leftarrow \sum_{i=1}^t x_{j_i} \leq t - 1$ 20: end if 21: while \mathcal{M} is feasible 22: return J_t 23: end procedure </pre>	<pre> 1: procedure RTermEnum(Cube indices I, specific assignment to non-cube IVs \mathbf{IV} or $\mathbf{IV} = \text{NULL}$, targeted degree t) 2: Declare an empty MILP model \mathcal{M} and an empty set $JR_t = \phi \subseteq$ $\{1, \dots, n\}$ 3: Declare \mathbf{x} as n MILP variables of \mathcal{M} corresponding to secret variables. 4: Declare \mathbf{v} as m MILP variables of \mathcal{M} corresponding to public variables. 5: $\mathcal{M}.con \leftarrow v_i = 1$ and assign $v_i.F = \delta$ for all $i \in I$ 6: $\mathcal{M}.con \leftarrow v_i = 0$ for all $i \in$ $(\{1, 2, \dots, n\} - I)$ 7: $\mathcal{M}.con \leftarrow \sum_{i=1}^n x_i \geq t$ and assign $x_i.F = \delta$ for all $i \in \{1, \dots, n\}$ 8: if $\mathbf{IV} = \text{NULL}$ then 9: $v_i.F = \delta$ for all $i \in$ $(\{1, 2, \dots, n\} - I)$ 10: else 11: Assign the flags of v_i, $i \in$ $(\{1, 2, \dots, n\} - I)$ as: </pre> $v_i.F = \begin{cases} 1_c & \text{if } \mathbf{IV}[i] = 1 \\ 0_c & \text{if } \mathbf{IV}[i] = 0 \end{cases}$ <pre> 12: end if 13: Update \mathcal{M} according to round functions and output functions 14: do 15: solve MILP model \mathcal{M} 16: if \mathcal{M} is feasible then 17: pick index set $\{j_1, \dots, j_{t'}\} \subseteq \{1, \dots, n\}$ s.t. $t' \geq t$ and $x_{j_1} = \dots = x_{j_{t'}} = 1$ 18: $JR_t = JR_t \cup \{j_1, \dots, j_{t'}\}$ 19: $\mathcal{M}.con \leftarrow \sum_{i \notin JR_t} x_i \geq 1$ 20: end if 21: while \mathcal{M} is feasible 22: return JR_t 23: end procedure </pre>
--	---

6.3 Applications to Grain-128a

For the attack on 184-round Grain-128a, the superpoly has degree $d = 14$, the number of involved key bits is $|J| = |J_1| = 21$ and we are able to enumerate all terms of degree 1–14 as Table 5.

Table 5. Results of Grain-128a with term Enumeration

#Rounds	$ I $	$ J_1 $	$ J_i $ ($2 \leq i \leq 14$)	$1 + \sum_{t=1}^d J_t $	Previous	Improved
184	95	21	157, 651, 1765, 3394, 4838, 5231, 4326, 2627, 1288, 442, 104, 15, 1	$2^{14.61}$	$2^{115.95}$	$2^{109.61}$

6.4 Applications to ACORN

For the attack on 704-round ACORN, with the cube dimension 64, the number of involved key bits in the superpoly is 72, and the degree is 7. We enumerate all the terms of degree from 2 to 7 as in Table 6, therefore we manage to improve the complexity of our cube attack in the previous section.

Table 6. Results of ACORN with Precise Term Enumeration

#Rounds	$ I $	$ J_1 $	$ J_2 $	$ J_3 $	$ J_4 $	$ J_5 $	$ J_6 $	$ J_7 $	$1 + \sum_{t=1}^d J_t $	Previous	Improved
704	64	72	1598	4911	5755	2556	179	3	$2^{13.88}$	$2^{93.23}$	$2^{77.88}$

Relaxed Algorithm 5. For the attack on 750-round ACORN (the superpoly is of degree $d = 5$), The left part of Algorithm 5 can only be carried out for the 5-degree terms $|J_5| = 46$. For $t = 2, 3, 4$, the sizes of J_t are too large to be enumerated. We settle for the index set JR_t containing the key indices that composing all the t -degree terms. For example, when $J_3 = \{(1, 2, 3), (1, 2, 4)\}$, we have $JR_3 = \{1, 2, 3, 4\}$. The relationship between J_t and JR_t is $|J_t| \leq \binom{|JR_t|}{t}$ and $J_1 = JR_1$. The searching space for J_t in Algorithm 5 is $\binom{|J_1|}{t}$ while that of the relaxed algorithm is only $\binom{|JR_t|}{t}$. So it is much easier to enumerate JR_t , therefore the complexity can still be improved (in comparison with Eq. (8)) as long as $|JR_t| < |J_1|$. The complexity of this relaxed version can be written as

$$\max\{2^{|I|} \times (1 + \sum_{t=1}^{d-1} \binom{|JR_t|}{t} + J_d), 2^{|I|} + 2^{|J|} \times (1 + \sum_{t=1}^{d-1} \binom{|JR_t|}{t} + J_d)\} \quad (12)$$

For 750-round ACORN, we enumerate J_5 and JR_1, \dots, JR_4 whose sizes are listed in Table 7. The improved complexity, according to Eq. (12), is $2^{120.92}$, lower than the original $2^{125.71}$ given in Appendix A in [25].

Table 7. Results of ACORN with Relaxed Term Enumeration

#Rounds	$ I $	$ JR_1 $	$ JR_2 $	$ JR_3 $	$ JR_4 $	$ J_5 $	$1 + \sum_{t=1}^{d-1} \binom{ JR_t }{t} + J_d $	Previous	Improved
750	101	81	81	77	70	46	$2^{19.92}$	$2^{125.71}$	$2^{120.92}$

7 A Clique View of the Superpoly Recovery

The precise & relaxed term enumeration technique introduced in Sect. 6 have to execute many MILP instances, which is difficult for some applications. In this section, we represent the resultant superpoly as a graph, which is called *superpoly graph*, so that we can utilize the clique concept from the graph theory to upper bound the complexity of the superpoly recovery phase in our attacks, without requiring MILP solver as highly as the term enumeration technique.

Definition 3 (Clique [33]). *In a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, a subset $C \subseteq V$, s.t. each pair of vertices in C is connected by an edge is called a clique.*

A *i-clique* is defined as a clique consists of i vertices, and i is called the *clique number*. A 1-clique is a vertex, a 2-clique is just an edge, and a 3-clique is called a triangle.

Given a cube C_I , by running Algorithm 5 for degree i , we determine J_i , which is the set of all the degree- i terms that might appear in the superpoly $p(\mathbf{x}, \mathbf{v})$ (see Sect. 6). Then we represent $p(\mathbf{x}, \mathbf{v})$ as a graph $G = (J_1, J_2)$, where the vertices in J_1 correspond to the involved secret key bits in $p(\mathbf{x}, \mathbf{v})$, the edges between any pairs of the vertices reveal the quadratic terms involved in $p(\mathbf{x}, \mathbf{v})$. We call the graph $G = (J_1, J_2)$ the *superpoly graph* of the cube C_I . The set of i -cliques in the superpoly graph is denoted as \mathcal{K}_i . Note that there is a natural one-to-one correspondence between the sets J_i and \mathcal{K}_i for $i = 1, 2$.

It follows from the definition of a clique that any i -clique in \mathcal{K}_i ($i \geq 2$) represents a monomial of degree i whose all divisors of degree 2 belong to J_2 . On the other hand, due to the “embed” Property 1 in Sect. 6, we have that all its quadratic divisors must be in J_2 . Then any monomial in J_i can be represented by an i -clique in \mathcal{K}_i . Hence for all $i \geq 2$, J_i corresponds to a subset of \mathcal{K}_i . Denote the number of i -cliques as $|\mathcal{K}_i|$, then $|J_i| \leq |\mathcal{K}_i|$. Apparently, $|\mathcal{K}_i| \leq \binom{|J|}{i}$ for all $1 \leq i \leq d$.

Now we show a simple algorithm for constructing \mathcal{K}_i from J_1 and J_2 for $i \geq 3$. For instance, when constructing \mathcal{K}_3 , we take the union operation of all possible combinations of three elements from J_2 , and only keep the elements of degree 3. Similarly, we construct \mathcal{K}_i for $3 < i \leq d$, where d is the degree of the superpoly. Therefore, all the i -cliques ($3 \leq i \leq d$) are found by the simple algorithm, i.e. the number of i -cliques $|\mathcal{K}_i|$ in $G(J_1, J_2)$ is determined. We therefore can upper bound the complexity of the offline phase as

$$2^{|I|} \times \left(1 + \sum_{i=1}^d |\mathcal{K}_i|\right). \tag{13}$$

Note that we have $|J_i| \leq |\mathcal{K}_i| \leq \binom{|J_1|}{i}$. It indicates that the upper bound of the superpoly recovery given by clique theory in Eq. (13) is better than the one provided by our degree evaluation in Eq. (8), while it is weaker than the one presented by our term enumeration techniques in Eq. (10). However, it is unclear if there exists a specific relation between $|\mathcal{K}_i|$ and $\binom{|JR_i|}{i}$ in the relaxed terms enumeration technique.

Advantage over the Terms Enumeration Techniques. In Sect. 6 when calculating J_i ($i \geq 3$) by Algorithm 5, we set the target degree as i and solve the newly generated MILP to obtain J_i , regardless of the knowledge of J_{i-1} we already hold. On the other hand, as is known in some cases, the MILP solver might take long time before providing J_i as desired. However, by using clique theory, we first acquire J_1 and J_2 , which are essential for the term enumeration method as well. According to the “embed” property, we then make full use of the knowledge of J_1 and J_2 , to construct \mathcal{K}_i for $i \geq 3$ by an algorithm which is actually just performing simple operations (like union operations among elements, or removal of repeated elements, etc) in sets. So hardly any cost is required to find all the \mathcal{K}_i ($3 \leq i \leq d$) we want. This significantly saves the computation costs since solving MILP is usually very time-consuming.

8 Conclusion

Algebraic properties of the resultant superpoly of the cube attacks were further studied. We developed a division property based framework of cube attacks enhanced by the flag technique for identifying proper non-cube IV assignments. The relevance of our framework is three-fold: For the first time, it can identify proper non-cube IV assignments of a cube leading to a non-constant superpoly, rather than randomizing trails & summations in the offline phase. Moreover, our model derived the upper bound of the superpoly degree, which can break the $|I| + |J| < n$ barrier and enable us to explore even larger cubes or mount to attacks on more rounds. Furthermore, our accurate term enumeration techniques further reduced the complexities of the superpoly recovery, which brought us the current best key recovery attacks on ciphers namely TRIVIUM, Kreyvium, Grain-128a and ACORN.

Besides, when term enumeration cannot be carried out, we represent the resultant superpoly as a graph. By constructing all the cliques of our superpoly graph, an upper bound of the complexity of the superpoly recovery can be obtained.

Acknowledgements. We would like to thank Christian Rechberger, Elmar Tischhauser, Lorenzo Grassi and Liang Zhong for their fruitful discussions, and the anonymous reviewers for their valuable comments. This work is supported by University of Luxembourg project - FDISC, National Key Research and Development Program of China (Grant No. 2018YFA0306404), National Natural Science Foundation of China (No. 61472250, No. 61672347), Program of Shanghai Academic/Technology

Research Leader (No. 16XD1401300), the Research Council KU Leuven: C16/15/058, OT/13/071, the Flemish Government through FWO projects and by European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-MSCA-ITN-2014-643161 ECRYPT-NET.

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)
2. 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)
3. 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)
4. 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)
5. Salam, M.I., Bartlett, H., Dawson, E., Pieprzyk, J., Simpson, L., Wong, K.K.-H.: Investigating cube attacks on the authenticated encryption stream cipher ACORN. In: Batten, L., Li, G. (eds.) ATIS 2016. CCIS, vol. 651, pp. 15–26. Springer, Singapore (2016)
6. 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, Part II. LNCS, vol. 10821, pp. 715–744. Springer, Cham (2018)
7. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 733–761. Springer, Heidelberg (2015)
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, Part II. LNCS, vol. 10211, pp. 259–288. Springer, Cham (2017)
9. Li, Z., Bi, W., Dong, X., Wang, X.: Improved conditional cube attacks on Keccak keyed modes with MILP method. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 99–127. Springer, Cham (2017)
10. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ASCON. IACR Trans. Symmetric Cryptol. **2017**(1), 175–202 (2017)
11. Dong, X., Li, Z., Wang, X., Qin, L.: Cube-like attack on round-reduced initialization of Ketje Sr. IACR Trans. Symmetric Cryptol. **2017**(1), 259–280 (2017)
12. 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, Part III. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017)

13. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015)
14. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015)
15. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016)
16. 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: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016)
17. Gu, Z., Rothberg, E., Bixby, R.: Gurobi optimizer. <http://www.gurobi.com/>
18. Sun, L., Wang, W., Wang, M.: MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. Cryptology ePrint Archive, Report 2016/811 (2016). <https://eprint.iacr.org/2016/811>
19. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017)
20. Funabiki, Y., Todo, Y., Isobe, T., Morii, M.: Improved integral attack on HIGHT. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017, Part I. LNCS, vol. 10342, pp. 363–383. Springer, Cham (2017)
21. Wang, Q., Grassi, L., Rechberger, C.: Zero-sum partitions of PHOTON permutations. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 279–299. Springer, Cham (2018)
22. 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>
23. Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 227–249. Springer, Cham (2017)
24. Fu, X., Wang, X., Dong, X., Meier, W.: A key-recovery attack on 855-round Trivium. Cryptology ePrint Archive, Report 2018/198 (2018). <https://eprint.iacr.org/2018/198>
25. 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>
26. Todo, Y., Isobe, T., Meier, W., Aoki, K., Zhang, B.: Fast correlation attack revisited—cryptanalysis on full Grain-128a, Grain-128, and Grain-v1. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 129–159. Springer, Cham (2018)
27. Lehmann, M., Meier, W.: Conditional differential cryptanalysis of Grain-128a. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 1–11. Springer, Heidelberg (2012)
28. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012)

29. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014)
30. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747 (2014). <https://eprint.iacr.org/2014/747>
31. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. Cryptology ePrint Archive, Report 2016/689 (2016). <https://eprint.iacr.org/2016/689>
32. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 185–215. Springer, Cham (2017)
33. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications, vol. 290. Macmillan, London (1976)