IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 1, pp. 363–395. DOI:10.13154/tosc.v2020.i1.363-395

Links between Division Property and Other Cube Attack Variants

Yonglin Hao[1], Lin Jiao[1], Chaoyun Li[2], Willi Meier[3], Yosuke Todo[4] and Qingju Wang[5]

[1] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China,
haoyonglin@yeah.net, jiaolin_jl@126.com
[2] imec - Computer Security and Industrial Cryptography (COSIC) research group, Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium, chaoyun.li@esat.kuleuven.be
[3] University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Windisch, Switzerland, willi.meier@fhnw.ch
[4] NTT Secure Platform Laboratories, Tokyo 180-8585, Japan, yosuke.todo.xt@hco.ntt.co.jp
[5] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Esch-sur-Alzette, Luxembourg, qingju.wang@uni.lu

Abstract. A theoretically reliable key-recovery attack should evaluate not only the *non-randomness* for the correct key guess but also the *randomness* for the wrong ones as well. The former has always been the main focus but the absence of the latter can also cause self-contradicted results. In fact, the theoretic discussion of wrong key guesses is overlooked in quite some existing key-recovery attacks, especially the previous cube attack variants based on pure experiments. In this paper, we draw links between the division property and several variants of the cube attack. In addition to the zero-sum property, we further prove that the bias phenomenon, the non-randomness widely utilized in dynamic cube attacks and cube testers, can also be reflected by the division property. Based on such links, we are able to provide several results: Firstly, we give a dynamic cube key-recovery attack on full Grain-128. Compared with Dinur et al.'s original one, this attack is supported by a theoretical analysis of the bias based on a more elaborate assumption. Our attack can recover 3 key bits with a complexity $2^{97.86}$ and evaluated success probability 99.83%. Thus, the overall complexity for recovering full 128 key bits is 2^{125}. Secondly, now that the bias phenomenon can be efficiently and elaborately evaluated, we further derive new secure bounds for Grain-like primitives (namely Grain-128, Grain-128a, Grain-V1, Plantlet) against both the zero-sum and bias cube testers. Our secure bounds indicate that 256 initialization rounds are not able to guarantee Grain-128 to resist bias-based cube testers. This is an efficient tool for newly designed stream ciphers for determining the number of initialization rounds. Thirdly, we improve Wang et al.'s relaxed term enumeration technique proposed in CRYPTO 2018 and extend their results on Kreyvium and ACORN by 1 and 13 rounds (reaching 892 and 763 rounds) with complexities $2^{121.19}$ and $2^{125.54}$ respectively. To our knowledge, our results are the current best key-recovery attacks on these two primitives.

Keywords: Stream ciphers · Division property · Dynamic cube attack · Cube tester · MILP · Grain-128 · Kreyvium · ACORN

1 Introduction

At CRYPTO 2018, Fu et al. proposed a new cube attack variant, referred to as the "IV-representation based cube attack" hereafter, and applied it to a key-recovery attack on 855-round Trivium [FWDM18]. According to [FWDM18], with the knowledge of

Received: 2019-09-01, Revised: 2019-11-23, Accepted: 2020-01-23, Published: 2020-05-07

particular key bits, the adversary can make a transformation to the first keystream bit satisfying: for the correct key guess, the transformed output bit has an algebraic degree much lower than the original one so that zero-sum properties can be detected using cube testers; for the wrong guess, the cube summation is a random 0/1. The degree of the (transformed) output bit is evaluated with an innovative method called "IV-representation". The reliability of such a method is based purely on Fu et al.'s implementation on a huge cluster along with what they referred to as "*man-made work*" [FWDM18]. But their implementation has no theoretically reliable proof or complexity analysis. In order to verify the correctness of their method, the authors of [FWDM18] propose a practical attack on 721-round Trivium claiming *"This process can be executed in an hour in a PC... for wrong guesses, the result is 1 with probability $\frac{1}{2}$"* as supportive material for their method.

Recently, another group of researchers gives a detailed analysis of the IV-representation based cube attack method in [FWDM18] and has some astonishing findings [HJL+18]. The first finding is that [FWDM18] is self-contradictive: according to actual experiments, the 721-round practical result itself is disproving their theories because, using their parameters, the cube summations are constantly zero for both correct and incorrect key guesses[1]. The second finding is that the precise evaluation of the superpoly degree for 855-round Trivium requires a memory complexity 2^{70}, far beyond the practical reach, and further deliberate reduction may result in wrong evaluations. Based on such findings, [HJL+18] makes a comment on the results in [FWDM18] as *"questionable"*[2].

The lesson we should learn is that: a theoretically reliable key-recovery result should include 2 proofs:

Proof 1 There is detectable *non-randomness* when the key guess is correct.

Proof 2 *Randomness* is verified when the key guess is wrong.

In fact, verification of random behavior if a key guess is wrong has been neglected in the past as well. In Dinur et al.'s dynamic cube attack on full Grain-128 [DGP+11], the authors have exhausted the capacity of a FPGA cluster only to test the correct guess on 107 randomly chosen keys. Whereas, for wrong guesses, they were unable to provide strong evidence for the randomness property of cube summations. They simply believe that the success of the small-cube reduced-round simulations in [DS11] can simply transplanted to the case of full-round attack. The result is not so positive: according to [DGP+11], even for the 107 correct guesses, they only find 8 of them having significant bias[3]. It remains to be a question whether for the 8 keys, the significant bias also exists when some of their 39 key bits are wrongly guessed. The absence of the wrong-key-guess discussion in [DGP+11] is due to a lack of computational capacity and a theoretic method: there are 39 key bits to be guessed and each of the $2^{39}-1$ wrong key guesses requires to run a cube tester of time complexity $2^{50}+50\cdot 2^{49}\approx 2^{54.7}$.

In this paper, we reveal that with the successful introduction of division property into the realm of cube attacks [TIHM17a, TIHM17b, WHT+17, WHT+18], new hopes appear for giving dynamic cube attacks having both Proof 1 and 2 even if the used cubes are very large.

Related Works. Division property is a generalization of the integral property initially proposed by Todo at EUROCRYPT 2015 [Tod15b]. It has a solid algebraic foundation and theoretically provable propagation rules so as to be successfully applied to cryptanalysis of MISTY1 and SIMON32/SIMECK32 soon afterwards [Tod15a, TM16]. With division

[1] The source code of [HJL+18] is at `https://github.com/peterhao89/Analyze721Trivium`

[2] In accordance with [HJL+18], the authors of [FWDM18] have agreed that their data were incorrect [FWD+18]. They also provide parameters different from those of [FWDM18] as a refinement to their practical result, but no further explanation has been given for the important theoretic result on 855-round Trivium [FWD+18].

[3] Quoted from [DGP+11]: *"by running a 50-dimensional cube tester for 107 random keys and discovering a very strong bias in about 7.5% of these key".*

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. At ASIACRYPT 2016, Xiang et al. [XZBL16] draw a link between division property and the MILP model[4]: an entry of the 0-1 vectors, referred to as the division property value of the state bit hereafter, is equivalent to a binary variable in the MILP model; the propagation rules can be described as linear constraints imposed on the variables. Thanks to Xiang et al., the memory consuming division property propagation has become an efficient MILP model solving process using existing highly efficient MILP solvers like Gurobi [GRB], which largely stimulates the application of division property to block cipher cryptanalysis for finding better integral characteristics [SWW16, SWW17, FTIM17, WGR18, WHG+18, HW19]. It was not until CRYPTO 2017 that the division property was introduced into the realm of cube attacks on stream ciphers [TIHM17a, TIHM17b].

Cube attack (as well as its distinguisher variant cube testers [ADMS09]), introduced by Dinur and Shamir [DS09] in the year 2009, has become a general tool for evaluating cryptographic primitives. It is usually regarded as a generalization of the chosen IV statistical attack on stream ciphers [Saa06, FKM08, EJT07] or a combination of higher order differential cryptanalysis and AIDA [Vie07]. Same with the setting of integral attacks, the cube attack also considers the situation when some input bits are active (cube IVs) while others being constants (non-cube IVs), and check the non-random property in the summation of an output bit. But the non-randomness of cube summations is not restricted to the zero-sum property. As is proved in [DS09], the cube summation is equal to a boolean function called the "*superpoly*" whose algebraic normal form (ANF) is a polynomial of key bits and non-cube IV bits. In addition to constant 0, cube attacks and testers can also cultivate other non-randomness of superpolies such as constantness, neutrality, linearity, bias, etc., and have been successfully applied to all kinds of cryptographic primitives, including stream ciphers [TIHM17a, ADMS09, DS11, FV13, SBD+16, LYWL18, KMN10, SMB17], hash functions [DMP+15, HWX+17, LBDW17] and authenticated encryption [LDW17, DLWQ17].

Among all such results, Todo et al.'s cube attack [TIHM17a] stands out as it draws a link between the division property and the key bits involved in the superpolies. Therefore, theoretic key recoveries can be achieved when only very few key bits are involved in the ANF of the superpolies, resulting in improved cryptanalysis results on several primitives namely Trivium, Acorn, Grain-128a and Kreyvium [TIHM17a, TIHM17b]. Recently at CRYPTO 2018, Wang et al. [WHT+17, WHT+18] have further improved the division property based cube attack by introducing the flag technique and the degree evaluation: the former takes the effect of constant 0-1 bits into MILP models for identifying proper non-cube IV assignments; the latter draws the highly qualified upper bounds to the algebraic degrees of the superpolies so as to further lower the complexities.

Motivations. Among various non-randomness utilized by cube attacks and testers, the bias phenomenon is the most popular one. There are cube testers based on bias and such bias testers are widely used to various stream ciphers [KMN10, SMB17, LLW15]. As to key recoveries, in addition to Dinur et al.'s dynamic cube attack [DS11], the correlation cube attack [LYWL18] also relies on biases in cube summations. However, once the non-randomness in **Proof 1** refers to the bias phenomenon, **Proof 2** corresponds to the absence or insignificance of biases for the wrong key guesses. Again, the division property method might be a promising tool for achieving this goal.

Our Contributions. We first draw links between the bias phenomenon and the division property. We find that the bias of a superpoly is closely related to a particular algebraic structure referred to as "split set". We prove that the minimal split set can draw lower bounds on the bias in superpolies. And we provide a heuristic algorithm with the advanced

[4] Short for Mixed Integer Linear Programming [MWGP11]. A mathematic model widely used in various cryptanalysis methods such as differential [SHW+14b, SHW+14a], linear [SHW+14a], impossible differential [CJF+16, ST17], zero-correlation linear [CJF+16] characteristics etc.

Table 1: Summarize all available attacks on full Grain-128

Method	Time / Data Complexity	Success Prob.	Reference
Dynamic Cube	$2^{93.71}/2^{54.71}$	7.5%	[DGP+11]
Dynamic Cube	$\mathbf{2^{125}/2^{94.86}}$	**99.83%**	**Section 6**
Fast Correlation	$2^{114.4}/2^{112.8}$	99%	[TIM+18]

division property using the flag technique and the degree evaluation method in [WHT+18], which can find split sets approximating the minimal one.

Secondly, we propose a new MILP modeling method for describing the division property propagations when the nullification strategies [DGP+11] are used. Our MILP models not only describe the zero-sum (or bias) phenomenon for the correct key guess but analyze the randomness in the wrong guesses as well. The theoretic cryptanalysis results using this new modeling method are equipped with both Proof 1 and Proof 2.

Based on such new techniques, we are able to give some new results:

1. We give a new dynamic cube attack on Grain-128 with theoretical analysis of the success probability basing on a more elaborate assumption than the randomness assumption used in Dinur et al..'s attack [DGP+11]. Our method only nullifies 1 intermediate state bit and only requires to guess 3 key bits, fewer than Dinur et al.'s 13 and 39 respectively. Furthermore, our cube testers have provable zero-sum properties for correct key guesses, much more reliable than the bias tester. Most importantly, the division property derived split set equips our attack with evidence to Proof 2 and a theoretical analysis of its success probability basing on a corollary derived from an elaborate assumption and the concept of the minimal split set. Our attack recovers 3 key bits with a complexity $2^{97.86}$ and a theoretically estimated success probability 99.83%. The remaining 125 key bits are to be exhaustively searched, so the overall complexity of our attack is 2^{125}. The success probability of our attack is overwhelmingly higher than that of Dinur et al.'s in [DGP+11], promising a better complexity on the average. In comparison with Todo et al.'s fast correlation attack in [TIM+18], the only theoretically reliable result on full Grain-128, our $2^{94.86}$ data complexity is lower than theirs' $2^{112.8}$. The comparison among the 3 attacks is in Table 1[5].

2. Similar to Liu's drawing secure bounds for Trivium-like ciphers against the zero-sum cube tester [Liu17], we draw improved secure bounds for Grain-like stream ciphers (namely Grain-128 [HJMM06], Grain-128a [ÅHJM11], Grain-V1 [HJM07] and Plantlet [MAM16]) against both zero-sum and bias cube testers. We find that for Grain-128, 254 rounds are enough for resisting zero-sum cube tester but, to resist bias testers, the initialization round number should increase to 265. On the contrary, the bounds for the other 3 primitives are 190, 82 and 138 respectively, far below their initialization round numbers. This indicates that our method can also be used as a useful tool in the design process of stream ciphers for identifying proper initialization round numbers.

3. We improve the current best key-recovery attack on Kreyvium [CCF+16] and Acorn [Wu16] by 1 and 13 rounds with complexities $2^{121.19}$ and $2^{125.54}$ respectively. The method of this attack is the same as that of [WHT+18] but our new MILP modeling technique enables us to cultivate more information from the superpoly in order to lower the complexities even further. As can be seen in Table 2, our results are the current best cube attacks on these two primitives.

[5] Note that there is another result given by Fu et al. in [FWC+17] which is using the same degree evaluation method with [FWDM18]. According to [HJL+18] and [FWD+18], such a method is "questionable" so we exclude it from Table 1.

Table 2: Cube Attacks on Kreyvium and Acorn

Target	Full Rounds	Attacked Rounds	Cube size	Complexity	Reference
Kreyvium	1152	872	85	2^{124}	[TIHM17a, TIHM17b]
		891	113	$2^{120.73}$	[WHT+17, WHT+18]
		892	**115**	$\mathbf{2^{121.19}}$	**Section 8**
Acorn	1792	503	5	practical	[SBD+16]
		704	64	2^{122}	[TIHM17a, TIHM17b]
		750	101	$2^{120.92}$	[WHT+17, WHT+18]
		763	**116**	$\mathbf{2^{125.54}}$	**Section 8**

Since all our results are affected by the accuracy of the division property, we check the feasibility of all our results with the current most accurate division property method in [YT19], proving that the problems reported in [YT19] do not occur in our results.

All our results, no matter theoretic deductions using MILP-aided division property or practical verifications, are equipped with C++ source codes in the supplementary materials for double-check[6].

Organization. Section 2 provides a brief description of the division property and the dynamic cube attack method in [DGP+11]. Section 3 formalizes the Proof 1 and 2 used in our dynamic cube attack. Section 4 gives a theoretical explanation to the bias phenomenon and introduce the concept of a minimal split set. It also presents the general idea of our split-set-based bias evaluation technique. Section 5 illustrates the division property based MILP model construction process for dynamic cube attacks, and shows how such new techniques can be used to find highly qualified split sets and cubes. Section 6 introduces the main procedure of our dynamic cube attack on full 256-round Grain-128. Section 7 discusses rationalities as well as limitations of our new techniques and compare our dynamic cube attack with Dinur et al.'s in detail. It also points out some promising directions for improvements. In Section 8, our theories are additionally applied for drawing secure bounds of Grain-like ciphers against bias testers, and improved cube attacks on Kreyvium and Acorn. Finally, we conclude the paper in Section 9.

2 Preliminaries

We first introduce the common notations frequently used in this paper:

- The small letters n and m are positive integers. The ith bit of a positive integer n is denoted as $n[i]$. The boolean field is denoted as $\mathbb{F}_2$. Its n-dimensional vector space is $\mathbb{F}_2^n$. We commonly use small letters to represent a single element and capital letters for sets. For a set S, we denote its size as $|S|$. We denote the complexity and success probability of an attack as $Comp$ and P_S respectively. The integer set $\{i, \dots, n-1\}$ is presented as $[i, n)$ for short. For the subset $J \subseteq [0, n)$, we denote its complementary set as $\bar{J}$. Let $x_0, \dots, x_{n-1}$ be n variables, then, for a set $J \subseteq [0, n)$, the corresponding set of variables is $x_J := \{x_j : j \in J\}$.
- An n-dimensional vector is denoted as $\vec{x} = (x_0, \dots, x_{n-1})$. Its ith entry ($i = 0, \dots, n-1$) is referred to as $\vec{x}[i] = x_i$. Furthermore, for a set $J \subseteq [0, n)$, we define the corresponding vector $\vec{k}_J \in \mathbb{F}_2^n$ s.t. $\vec{k}_J[j] = 1$ for $j \in J$ and $\vec{k}_J[j] = 0$ for $j \in \bar{J}$. For two n-dimensional boolean vectors $\vec{x} = (x_0, \dots, x_{n-1})$ and $\vec{k} = (k_0, \dots, k_{n-1}) \in \mathbb{F}_2^n$, there is an operation $\vec{x}^{\vec{k}} = \prod_{i=0}^{n-1} x_i^{k_i}$. For example, let $x_{[0,n)}$ in $\vec{x}$ be n boolean variables, then any polynomial p on the ring $\mathbb{F}_2[x_{[0,n)}]$ can be represented as

$$p(\vec{x}) = \sum_{\vec{k} \in \mathbb{F}_2^n} a_{\vec{k}} \vec{x}^{\vec{k}}, \text{ where } a_{\vec{k}} \in \mathbb{F}_2. \tag{1}$$

[6] https://github.com/peterhao89/Grain128DynamicCubeAttack.git

Furthermore, we denote $\vec{e}_i \in \mathbb{F}_2^n$ for $i \in [0,n)$ as the n unit vectors ($\vec{e}_i[i] = 1$ and $\vec{e}_i[j] = 0$ for $j \neq i$) while $\vec{1}$ and $\vec{0}$ are vectors with all entries being 1 and 0 respectively. Two vectors $\vec{u}, \vec{k} \in \mathbb{F}_2^n$ have relation $\vec{u} \succeq \vec{k}$ if $\vec{u}[i] \geq \vec{k}[i]$ for all $i \in [0,n)$.

- The algebraic degree of a polynomial p is represented as $\deg(p)$. Specifically, if p is constant 0 or 1, we have $\deg(0) = -1$ and $\deg(1) = 0$.
- $\|$ is the concatenation such as $\vec{x}\|a = (x_0, \ldots, x_{n-1}, a)$. $+$ in a boolean polynomial refers to the XOR operation on bits. $\wedge$ is the bitwise AND of two n-dimensional binary vestors: $\vec{x}, \vec{y} \in \mathbb{F}_2^n$, $\vec{x} \wedge \vec{y} = (x_0 \cdot y_0, \ldots, x_1 \cdot y_1)$
- An MILP model is denoted as $\mathcal{M}$. It contains boolean variables $\mathcal{M}.var$ and constraints imposed on them $\mathcal{M}.con$. It can also have an objective function, denoted as $\mathcal{M}.obj$, maximizing (or minimizing) a linear expression. It can be solved by calling $\mathcal{M}.optimize()$ and it returns either 0 or 1. If $\mathcal{M}.optimize() = 0$, the model is infeasible; otherwise, the model has feasible solutions and we get a proper assignment to all variables in $\mathcal{M}.var$ satisfying all constraints in $\mathcal{M}.con$. For example, $\mathcal{M}.var = \{x_0, x_1, x_2\}$ and $\mathcal{M}.con$ may contain equality or inequalities imposed on them as:

$$\mathcal{M}.con : \begin{cases} x_0 + x_1 + x_2 \geq 2 \\ x_0 + x_1 = x_2. \end{cases}$$

We can also define its objective function as $\mathcal{M}.obj = \max\{x_1 + x_2\}$. We solve the model and find $\mathcal{M}.optimize() = 1$ along with a proper assignment $x_0 = 0, x_1 = 1, x_2 = 1$.

2.1 The Theoretic Basis of Cube Related Cryptanalysis Methods

Consider a stream cipher with n secret key bits $\vec{x} = (x_0, \ldots, x_{n-1})$ and m public initialization vector (IV) bits $\vec{v} = (v_1, \ldots, v_{m-1})$. Then, the first output keystream bit can be regarded as a polynomial of $x_{[0,n)}$ and $v_{[0,m)}$ referred to as $f(\vec{x}, \vec{v})$. For set $I \subseteq [0,m)$ and the corresponding vector $\vec{k}_I \in \mathbb{F}_2^m$, the algebraic normal form (ANF) of $f(\vec{x}, \vec{v}) \in \mathbb{F}_2[\vec{x}, \vec{v}]$ can be uniquely decomposed as

$$f(\vec{x}, \vec{v}) = p(\vec{x}, \vec{v}) \cdot \vec{v}^{\vec{k}_I} + q(\vec{x}, \vec{v}), \tag{2}$$

where the monomials of $q(\vec{x}, \vec{v})$ miss at least one variable from v_I while $p(\vec{x}, \vec{v})$ is the so-called superpoly [DS09] and is on the polynomial ring $\mathbb{F}_2[x_{[0,n)}, v_{\bar{I}}]$. Following [WHT+18], we refer the indices in I as "cube indices", the variables in v_I as cube IV and those of $v_{\bar{I}}$ as non-cube IV. Since the superpoly p in (2) is only related to key bits and non-cube IVs, according to (1), we can represent the ANF of p as

$$p(\vec{x}, \vec{v}) = \sum_{\vec{k} \in \mathbb{F}_2^n} a_{\vec{k}} \vec{x}^{\vec{k}}, \quad \text{where } a_{\vec{k}} \in \mathbb{F}_2[v_{\bar{I}}].$$

Arbitrary binary vector $\vec{IV} \in \mathbb{F}_2^m$ corresponds to a specific assignment to non-cube IVs. With I and $\vec{IV}$, we can define a structure called "cube", denoted as $C_I(\vec{IV})$, consisting of $2^{|I|}$ vectors from $\mathbb{F}_2^m$:

$$C_I(\vec{IV}) := \{\vec{v} \in \mathbb{F}_2^m : \vec{v}[i] = 0/1, i \in I \bigwedge \vec{v}[s] = \vec{IV}[s], s \notin I\}. \tag{3}$$

As is proved in [DS09], the ANF of the superpoly p corresponding to the non-cube IV assignment $\vec{IV}$ can be computed by summing the output bit over the cube $C_I(\vec{IV})$ as:

$$p_{\vec{IV}}(\vec{x}) := p(\vec{x}, \vec{IV}) = \bigoplus_{\vec{v} \in C_I(\vec{IV})} f(\vec{x}, \vec{v}). \tag{4}$$

Since C_I is defined according to I, we may also refer I as the "cube" without causing ambiguities and the size $|I|$, is referred to as the "dimension" of the cube.

All cube related cryptanalysis methods are utilizing the non-random properties of the superpoly $p_{\vec{IV}}$ in (4) (for some $\vec{IV}$). The cube attacks on Keccak [DMP+15, HWX+17, LBDW17, DLWQ17] and Ascon [LDW17] are utilizing the zero-sum property ($p_{\vec{IV}}(\vec{x}) \equiv 0$) The original cube attack in [DS09] can recover secret key bits when $p_{\vec{IV}}(\vec{x})$ is linear ($\deg(p_{\vec{IV}}) = 1$). Todo et al.'s division property based method in [TIHM17a, TIHM17b] determines a small set $J \subseteq [0, n)$ s.t. $p_{\vec{IV}}(\vec{x}) \in \mathbb{F}_2[x_J]$. It precomputes the whole truth table of $p_{\vec{IV}}$ and key bits can only be recovered when $p_{\vec{IV}}$ is non-constant. So it requires $\deg(p_{\vec{IV}}) \geq 1$ and $|J| + |I| < n$. Wang et al.'s improved version [WHT+18] imposes the flag technique for identifying proper $\vec{IV}$ and the degree evaluation technique for upper bounding the algebraic degree of $p_{\vec{IV}}$ as $\deg(p_{\vec{IV}}) \leq d$ for some $d \geq 0$. They relax Todo et al.'s $|J| + |I| < n$ restriction to

$$2^{|I|} \times \binom{|J|}{\leq d} \leq 2^n, \quad \text{where } \binom{|J|}{\leq d} := \sum_{t=0}^{d} \binom{|J|}{t}. \tag{5}$$

The dynamic cube attacks [DGP+11, DS11] as well as the cube testers in [ADH+09] on Grain-128 utilize a quite different property called "bias": for random $\vec{x} \in \mathbb{F}_2^n$ and particular $\vec{IV}$ (usually $\vec{IV} = \vec{0}$ [DS11]), the values of $p_{\vec{IV}}(\vec{x})$ have a bias $\epsilon > 0$ towards 0:

$$\Pr[p_{\vec{IV}}(\vec{x}) = 0] = 2^{-1} + \epsilon, \quad 0 \leq \epsilon \leq 2^{-1}. \tag{6}$$

When ϵ is big enough, it can be used for either distinguishing the correct key guesses from the wrong ones [DGP+11, DS11] or distinguishing the Grain-128 keystreams from truly random bitstreams [ADH+09]. We'll briefly introduce the procedures of Dinur et al.'s dynamic cube attack in Section 2.5.

2.2 Bit-Based Division Property and its MILP Representation

The division property based cube attacks in [TIHM17a, WHT+18] use the bit-based division property first introduced in [TM16]. Its definition is as the following Definition 1:

Definition 1 (Bit-Based Division Property [TM16])**.** Let X be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let K be a set whose elements take an n-dimensional bit vector. When the multiset X has the division property K, it fulfils the following conditions:

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

According to Definition 1, the division property of multiset $X \subseteq \mathbb{F}_2^n$ is grasped by the set $K \subseteq \mathbb{F}_2^n$. For a input multiset X with division property K, when the basic bitwise operations COPY, XOR, AND are applied to all the elements in X and acquire X', the division property of X', denoted by K', can be deduced following the propagation rules `copy`, `xor`, `and` proved in [Tod15b, TM16]. Specifically, let $I \subseteq [0, n)$ be the set containing all the indices of active IV bits; the $2^{|I|}$ initial states compose a set X^0 with division property $K^0 = \{\vec{k}_I\}$. Since the round functions of all cryptographic primitives can be represented as the combinations of the three basic operations, the division property propagation for an R-round encryption can be evaluated as

$$\{\vec{k}_I\} = K^0 \to K^1 \to \cdots \to K^R. \tag{7}$$

If $\vec{e}_i \notin K^R$ for some $i \in [0, n)$, we know the ith bit of the ciphertext has zero-sum property; otherwise, the summation of the ith bit is *unknown*. But the sizes of K^r ($r \in [R]$) expand exponentially to $O(2^n)$ consuming huge memory resources [TM16]. So Xiang et al.

proposed the MILP modeling method and solved the memory crisis [XZBL16]. They first transform the set operations in (7) into a vector propagation, referred to as the "division trail", as follows:

Definition 2 (Division Trail [XZBL16])**.** Consider the division propagation in (7), for any vector $\vec{k}^{i+1} \in K^{i+1}$ $(i \in [0, R-1))$, there must exist a vector $\vec{k}^i \in K^i$ s.t. $\vec{k}^i$ can propagate to $\vec{k}^{i+1}$ by the propagation rules of the division property. For $(\vec{k}^0, \ldots, \vec{k}^R) \in (K^0 \times \cdots \times K^R)$, if $\vec{k}^i$ can propagate to $\vec{k}^{i+1}$ for all $i \in [R-1)$, we call $\vec{k}^0 \to \cdots \to \vec{k}^r$ an R-round division trail.

Xiang et al. also prove that all division trails can be covered by a MILP model $\mathcal{M}$: all entries in $\vec{k}^i$ $(i \in [R])$ are binary variables in $\mathcal{M}.var$; the propagation rules `copy`, `xor`, `and` are equivalent to linear constraints in $\mathcal{M}.con$; to check whether $\vec{e}_j \notin K^R$, we need to impose constraints $\mathcal{M}.con \leftarrow \vec{e}_j = \vec{k}^R$ and call $\mathcal{M}.optimize()$: if it returns 1, we know $\vec{e}_j \in K^R$, the summation of the jth bit is *unknown*; otherwise, $\vec{e}_j \notin K^R$ and the jth bit has zero-sum property. The model solving completely relies on the highly efficient MILP solver like Gurobi [GRB]. Some simplifications have been made to the MILP descriptions of `copy`, `xor`, `and` [SWW16, TIHM17a] in order to reduce the workload of the solvers.

2.3 The Division Property Based Cube Attacks

In the context of cube attacks, the following structure of a stream cipher is assumed. The n-bit key $\vec{x}$ and m-bit IV $\vec{v}$ are first used to initialize a large register $\vec{s}^0$ of length $l > m+n$ containing key, IV and constant 0/1 bits as $\vec{s}^0 = \vec{k}\|\vec{v}\|\vec{c}$. Then, the stream cipher starts an R-round initialization phase and acquires $\vec{s}^1, \ldots, \vec{s}^R$ by iteratively calling the updating function $\vec{s}^{i+1} = \mathtt{Upd}(\vec{s}^i)$ $(i \in [0, R))$. Finally, an output bit z^R is generated from $\vec{s}^R$ by calling the output function $z^R = \mathtt{Output}(\vec{s}^R)$.

Therefore, the division property values for describing cube attacks on stream ciphers should make modifications accordingly [TIHM17a, WHT$^+$18]. For cube index set I, the division property values for $\vec{x}$ and $\vec{v}$ should first be decided as $\vec{k}_x\|\vec{k}_v \in \mathcal{M}.var^{n+m}$. More specifically, $\vec{k}_v = \vec{k}_I$ for cube I. After that, the division property for $\vec{s}^0$ is decided as $\vec{k}^0$ according to $\vec{k}_x, \vec{k}_v$ as $\vec{k}^0$. For constant state bits in $\vec{s}^0$, the division values are set to 0 ($\mathcal{M}.con \leftarrow \vec{k}^0[j] = 0$ if $\vec{s}^0[j]$ is set to constant 0 or 1). The following R updating and output function calls will result in division values $\vec{k}^1, \ldots, \vec{k}^R \in \mathcal{M}.var^l$, $o \in \mathcal{M}.var$ corresponding to the division property values of $\vec{s}^0, \ldots, \vec{s}^R$ and z respectively.

The critical division trail for division property based cube attack is $\vec{k}_x\|\vec{k}_v \to \vec{k}^R\|o$ (equivalent to the $\vec{k}^0 \to o$ expressions in [TIHM17a, WHT$^+$18]). Todo et al. [TIHM17a] proved Proposition 1 and Wang et al. [WHT$^+$18] gave a generalization as Proposition 2.

Proposition 1. *([TIHM17a]) For $j \in [0, n)$, if $\vec{e^j}\|\vec{k}_I \to \vec{0}\|1$ does NOT exist, for arbitrary $\vec{IV} \in \mathbb{F}_2^m$, the corresponding superpoly $p_{\vec{IV}}$ does NOT involve x_j. On the contrary, if $\vec{e^j}\|\vec{k}_I \to \vec{0}\|1$ exists, there might be a $\vec{IV} \in \mathbb{F}_2^m$ s.t. x_j involved in the superpoly $p_{\vec{IV}}$.*

Proposition 2. *([WHT$^+$18]) For $W \in [0, n)$, if $\vec{k}_W\|\vec{k}_I \to \vec{0}\|1$ does NOT exist, for arbitrary $\vec{IV} \in \mathbb{F}_2^m$, the corresponding superpoly $p_{\vec{IV}}$ does NOT involve the monomial $\vec{x}^{\vec{k}_W}$. If $\vec{k}_W\|\vec{k}_I \to \vec{0}\|1$ exists, there might be an $\vec{IV} \in \mathbb{F}_2^m$ s.t. the monomial $\vec{x}^{\vec{k}_W}$ is involved in the superpoly $p_{\vec{IV}}$.*

Based on Proposition 1, they let j traverse $[0, n)$ and decide a key index set $J \subseteq [0, n)$ s.t. the superpoly $p_{\vec{IV}}$ of I can only be a function of key bits x_J [TIHM17a]. They prove that for proper $\vec{IV}$ making $p_{\vec{IV}} \not\equiv 0$, the whole truth table can be constructed offline with complexity $2^{|I|+|J|}$ with which 1 key bit can be recovered online with a cube summation and a table lookup. Proposition 2 enables the adversaries to draw upper bound for the algebraic degree of $p_{\vec{IV}}$: they can find an integer $d \geq 0$ s.t. $\deg(p_{\vec{IV}}) \leq d$. With such an upper bound d, the complexity decreases from $2^{|I|+|J|}$ to no more than $2^{|I|} \cdot \binom{|J|}{\leq d}$ (5). For $t = 1, \ldots, d$, Proposition 2 also enables to find a set $J^t \subseteq [0, n)^t$ corresponding to all t-degree monomial terms possibly appearing in $p_{\vec{IV}}$ (Note: $J^1 = J$ as is explained in [WHT$^+$18]). Considering that the determining J^t is too time consuming for Gurobi, Wang et al. also proposed a relaxed version that determines a relaxed set $\tilde{J}^t \subseteq [0, n)$ s.t. $J^t \subseteq \left(\tilde{J}^t\right)^t \subseteq [0, n)^t$ and the complexities of some key-recovery attacks can be further reduced to (8) according to [WHT$^+$18]:

$$Comp = \max \left\{ 2^{|I|} \cdot (1 + \sum_{t=1}^{d} \binom{|\tilde{J}^t|}{t}), 2^{|I|} + 2^{|J|} \cdot (1 + \sum_{t=1}^{d} \binom{|\tilde{J}^t|}{t}) \right\}. \tag{8}$$

Note that in [WHT$^+$18],$\tilde{J}^t \supseteq \tilde{J}^{t+1}$ for $t = 1, \ldots, d-1$: indicating that key bits involved in nonlinear parts should always be assumed to appear in the linear parts. We remove such restriction in Section 8.2 and acquire improved key-recovery results.

Besides degree evaluation and term enumeration, Wang et al. also find that some division trails are impossible when specific constant 0/1 bits are involved. So they introduce the "flag technique" for finding proper $\vec{IV}$'s [WHT$^+$17, WHT$^+$18]. According to [WHT$^+$18], the division property of arbitrary bit s has 2 parameters $s.val$ and $s.F$:

- $s.val \in \mathcal{M}.var$ is the division property value as before so it is permanently binary variables taking values either 0 or 1.
- $s.F \in \{0_c, 1_c, \delta\}$ is the "flag" value where $0_c, 1_c, \delta$ specifies whether the state bit is constant 0, constant 1 or variable (active IV, unknown key bits, cube IV's or non-cube IV bits with arbitrary value are all corresponding to flag value δ).

The flag value helps to track the constant bits and control the division property propagations. Corresponding to the bitwise EQUAL, XOR and AND operations, the flag values $0_c, 1_c, \delta$ has $=$, $\oplus$ and $\times$ operations. The $=$ operation is naturally $1_c = 1_c$, $0_c = 0_c$ and $\delta = \delta$. The $\oplus$ and $\times$ operations follow the rules:

$$\begin{cases} 1_c \oplus 1_c = 0_c \\ 0_c \oplus x = x \oplus 0_c = x \\ \delta \oplus x = x \oplus \delta = \delta \end{cases} \text{and} \begin{cases} 1_c \times x = x \times 1_c = x \\ 0_c \times x = x \times 0_c = 0_c, \\ \delta \times \delta = \delta \end{cases} \tag{9}$$

for arbitrary $x \in \{1_c, 0_c, \delta\}$. We refer to such a structure s with both division property value and flag value as "DP structure" hereafter and its propagation rules corresponding to the COPY, XOR and AND operations have become `copyf, xorf, andf` described as Proposition 3, 4 and 5 respectively. With the DP structure and its propagation rules, the degree evaluation (`DegEval`), term enumeration (`TermEnum`) and its relaxed version (`RTermEnum`) are uniformly described in the framework in Algorithm 1 where we add an additional parameter "etc.", because additional parameters are required for such models to describe the dynamic cube attacks in [DGP$^+$11] as well as our new methods. It is also noticeable that when, `DegEval` returns 0, it only proves that the superpoly is a constant (either 0 or 1). This is a constant-sum property and can also serve as a cube tester [ADMS09]. If `DegEval` returns -1, we can fully convince that the cube has a zero-sum property (a special case of the constant-sum property) because its superpoly is constant 0 according to Definition 1. Such a denotation of constant 0 is in accordance with the definition of deg($\cdot$) at the beginning of Section 2. As distinguishers, both the constant-sum and zero-sum can be used for identifying the correct key guesses. But, since all the cubes used in our dynamic attacks have the zero-sum property (`DegEval` returns -1) for the correct key guess, we only use the term *zero-sum* hereafter. The specific effectiveness of utilizing 0, rather than 1, biasing property has already been noticed by Dinur et al..

They have pointed out in [DS11, DGP+11] that Grain-128 superpolies are usually sparse polynomials: for randomly chosen keys, the values of Grain-128 superpolies are more likely to 0.

Proposition 3 (MILP Model for COPY with Flag [WHT+18]). *Let $a.val \rightarrow (b_1.val, \dots, b_m.val)$ be a division trail of COPY. The DP structure propagation rule* `copyf` *is:*

$$\begin{cases} \mathcal{M}.con \leftarrow a.val = b_1.val + \cdots + b_m.val \\ a.F = b_1.F = \ldots = b_m.F \end{cases}$$

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

Proposition 4 (MILP Model for XOR with Flag [WHT+18]). *Let $(a_1.val, \dots, a_m.val) \rightarrow b.val$ be a division trail of XOR. Its DP structure propagation rule* `xorf` *is:*

$$\begin{cases} \mathcal{M}.con \leftarrow a_1.val + \cdots + a_m.val = b.val \\ b.F = a_1.F \oplus a_2.F \oplus \cdots \oplus a_m.F \end{cases}$$

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

Proposition 5 (MILP Model for AND with Flag [WHT+18]). *Let $(a_1.val, \dots, a_m.val) \rightarrow b.val$ be a division trail of AND. Its DP structure propagation rule* `andf` *is:*

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

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

Algorithm 1 The general framework for `DegEval`, `TermEnum` and `RTermEnum` in [WHT+17] where `RTermEnum` can be acquired from `TermEnum` by replacing the blue parts with the corresponding comments.

```
procedure DegEval($I \subseteq [0, m)$, $\vec{IV} \in \mathbb{F}_2^m$, round $R$, etc.)
    Declare an empty MILP model $\mathcal{M}$.
    Declare $\vec{k}_x, \vec{k}_v$ as the DP structures for key and IV bits and assign their division property values and flag values according to $I$, $\vec{IV}$ etc. Denote the procedure as $(\vec{k}_x, \vec{k}_v) \leftarrow \texttt{IniDP}(\mathcal{M}, I, \vec{IV}, R, \text{ etc.})$.
    Add constraint $\mathcal{M}.con \leftarrow \sum_{i \in [0,n)} x_i.val \geq 0$.
    Set objective of $\mathcal{M}$ as $\mathcal{M}.obj = \max \sum_{i \in [0,n)} x_i.val$.
    With $\vec{k}_x, \vec{k}_v$ etc., update the model $\mathcal{M}$ according to $R$ Upd calls and 1 Output call describing the division trail $\vec{k}^0 \rightarrow \cdots \rightarrow \vec{k}^R \| o$. Denote the procedure as $(\mathcal{M}, \vec{k}^0, \dots, \vec{k}^R, o) \leftarrow \texttt{ModelConstruct}(\vec{k}_x, \vec{k}_v, R, \text{ etc.})$.
    If $1 = \mathcal{M}.optimize()$, set the degree $d = \mathcal{M}.obj$; otherwise, set $d = -1$, indicating a constant 0 superpoly and the zero-sum property.
    Return $d$.
end procedure
```

```
procedure TermEnum($I \subseteq [0, m)$, $\vec{IV} \in \mathbb{F}_2^m$, round $R$, targeted term degree $t$, etc.)        ▷ RTermEnum.
    Declare an empty MILP model $\mathcal{M}$.
    Declare a set $J^t = \phi \subseteq [0, n)^t$.                                                ▷ $\tilde{J}^t = \phi \subseteq [0, n)$.
    Declare $\vec{k}_x, \vec{k}_v$ as the division properties of key and IV bits and assign their division property values and flag values according to $I$, $\vec{IV}$ etc.
    Add constraint $\mathcal{M}.con \leftarrow \sum_{i \in [0,n)} x_i.val = t$.
    Call $(\mathcal{M}, \vec{k}^0, \dots, \vec{k}^R, o) \leftarrow \texttt{ModelConstruct}(\vec{k}_x, \vec{k}_v, R, \text{ etc.})$.
    while $\mathcal{M}.optimize() = 1$ do
        Identify $0 \leq i_1 \leq \ldots \leq i_t \leq n - 1$ s.t. $x_{i_j}.val = 1$ for $j = 1, \dots, t$.
        Update $J^t \leftarrow J^t \cup \{(i_1, \dots, i_t)\}$.                                      ▷ $\tilde{J}^t \leftarrow \tilde{J}^t \cup \{i_1, \dots, i_t\}$.
        Add $\mathcal{M}.con \leftarrow \sum_{j=1}^{t} x_{i_j}.val \leq t - 1$.                          ▷ $\mathcal{M}.con \leftarrow \sum_{i \notin \tilde{J}^t} x_i.val \geq 1$.
    end while
    Return $J^t$.                                                                                   ▷ $\tilde{J}^t$.
end procedure
```

2.4 The Accuracy of the Division Property based Cube Attacks

The division property can prove the zero-sum property of cubes theoretically. When division trails show the superpoly non-constant, it is a function of key bits with high probability. But such a probability is not 100% in some cases even with the flag technique [WHT+18]. The reason is obvious: the division property itself cannot handle the effect of monomial cancellation (such as $x_1x_2 + x_1x_2 = 0$). The theoretic remedy has already been proposed in [TM16] referred to as the bit-based division property with three subsets (BDPTS). But the implementation is far beyond practical reach since it requires to enumerate almost all division trails.

The issue of accuracy is discussed extensively recently [WHG+18, YT19, WHG+19, HW19]. Various compromises are given and the general frameworks for R-round stream ciphers can all be summarized as follows:

$$0 \xrightarrow{\vec{k}} R \Rightarrow 0 \xrightarrow{\text{accurate}} R_m \xrightarrow{\vec{k}} R. \tag{10}$$

The whole cipher is split into 2 stages: the $0 \to R_m$ stage is accurately evaluated either using BDPS [WHG+18] or precise ANF deductions [YT19]. Then, all division trails in $R_m \to R$ are enumerated in a conventional manner and the validity of each trail is checked according to the accurately deduced information in $0 \to R_m$. In this way, the cancellation effects in $0 \to R_m$ is handled so that the accuracy is improved. The limitations are also obvious:

1. R_m cannot be large;
2. The trails within $R_m \to R$ should be very few, suitable for practical evaluations.
3. The cancellation effects in $R_m \to R$ is still not handled.

The most significant finding is that, for some key-recovery attacks on Trivium, there is only 1 division trail within $R_m \to R$ and such a division trail cannot add key bits to the superpolies according to their analysis to the ANFs of the intermediate state bits at round R_m [YT19]. We are to use the ANF deduction method in [YT19] for double-check our results.

2.5 Brief Introduction to Dinur et al.'s Dynamic Cube Attacks

In this part, we briefly introduce Dinur et al.'s dynamic cube attack on full Grain-128 [DGP+11]. In a cryptographic primitive with n key bits $\vec{x} = (x_0, \ldots, x_{n-1})$ and m public IV bits $\vec{v} = (v_0, \ldots, v_{m-1})$, the ANFs of its arbitrary bit can be regarded as a polynomial $f(\vec{x}, \vec{v})$ as follows:

$$f(\vec{x}, \vec{v}) = \sum_{\vec{u} \in \mathbb{F}_2^m} a_{\vec{u}} \vec{v}^{\vec{u}} \text{ where } a_{\vec{u}} \in \mathbb{F}_2[x_{[0,n)}]. \tag{11}$$

Dinur et al. [DGP+11] affect the output bit by replacing $\vec{v}$ by some $\vec{\hat{v}}$. For Grain-128, after expressing the output bit z as a polynomial of intermediate state bits, Dinur et al. find that the algebraic degree of the output bit can be decided by only 1 high-degree monomial. Therefore, if particular intermediate state bits are nullified (set to 0), the high-degree monomial can be largely simplified so that the degree of the output bit drops making the output bit vulnerable to bias cube testers. A state bit s_i^r (generated at round r at position i) can be nullified by assigning particular IV bit to a dynamic value decided by the ANF of s_i^r. For example, we start from $\vec{v} = (v_0, \ldots, v_{m-1})$, if

$$s_i^r = s_i^r(\vec{x}, \vec{v}) = v_l + f \text{ where } f \in \mathbb{F}_2[x_{[0,n)}, v_{[0,m)\setminus\{l\}}],$$

s_i^r can be nullified by setting the l-th entry of $\vec{v}$ to f. Let $\vec{\hat{v}}$ be a vector deduced from $\vec{v}$ as

$$\vec{\hat{v}}[j] = \begin{cases} \vec{v}[j], & j \neq l \\ f, & j = l \end{cases}$$

and s_i^r is therefore nullified for $s_i^r = s_i^r(\vec{x}, \hat{\vec{v}}) = 0$. The expression f is referred to as dynamic value and the ith IV bit is referred to as the dynamic IV. If there are multiple crucial bits $s_{i_1}^{r_1}, \ldots, s_{i_t}^{r_t}$ (we constantly assume $r_1 \leq \ldots \leq r_t$ hereafter) to be nullified, we also start from the same $\vec{v}$ and repeat the procedure t times: acquire $\hat{\vec{v}}_1, \ldots, \hat{\vec{v}}_t$ by recursively replacing l_1−th, …, l_t−th IV bits with the corresponding dynamic values $f_1, \ldots, f_t$. Let $\hat{\vec{v}} := \hat{\vec{v}}_t$, we know $s_{i_1}^{r_1}, \ldots, s_{i_t}^{r_t}$ are successfully nullified. As can be seen, the dynamic values can be uniquely decided by the pairs $N_S = \{(l_1, s_{i_1}^{r_1}), \ldots, (l_t, s_{i_t}^{r_t})\}$, so we refer to the set N_S as the "nullification strategy" hereafter. The combination of $(N_S, I, \vec{IV})$ can decide particular key bits that need to be guessed for assignment to dynamic values. The correct key guess is supposed to be uniquely identified using cube testers dedicated for particular non-randomness (such as bias phenomenon in [DGP+11]) in cube summations.

3 Formalize Proof 1 and 2 in Dynamic Cube Attacks

With the nullification strategy N_S defined in Section 2.5 along with cube I and non-cube IV assignment $\vec{IV}$, we can deduce the ANF of the dynamic values $f_1, \ldots, f_t \in \mathbb{F}_2[x_{[0,n)}, v_I]$ for assigning the IV bits at positions $l_1, \ldots, l_t$ during cube summations. Detailed analysis to $f_1, \ldots, f_t$ enables us to determine the to-be-guessed key bits for computing dynamic values. We denote to-be-guessed key bits as $\vec{G} = (g_0, \ldots, g_{\kappa-1})$ where $g_j \in \mathbb{F}_2[x_{[0,n)}]$ for $j \in [\kappa)$. Such $\vec{G}$ is determined with the method in [HJL+18] making sure that only the correct key guess can nullify all targeted intermediate state bits. We also follow [HJL+18] and denote the ANF vectors of the 2^κ possible key guesses as $\vec{G}_0, \ldots, \vec{G}_{2^\kappa-1}$, where $G_w[j] = g_j + w[j] \in \mathbb{F}_2[x_{[0,n)}]$ for $w \in [0, 2^\kappa)$ and $j \in [0, \kappa)$. Apparently, $w = 0$ is the correct key guess. We also specify $w = 1$ as the particular key guess s.t. $s_{i_t}^{r_t} = 1$ and all targeted intermediate state bits are successfully nullified.

In dynamic cube attacks, the correct key guess $\vec{G}_0$ is supposed to be identified using cube summations over several different qualified cube I's dedicated for detecting the bias phenomenon in Definition 3. Such a definition corresponds to the simple fact that the truth table of the boolean polynomial $p_{\vec{IV}}(\vec{x})$ contains more 0 than 1 entries.

Definition 3. (General Bias Phenomenon In Cube Summations) Let $p_{\vec{IV}}(\vec{x})$ be the superpoly of cube I with non-cube IVs assigned to $\vec{IV}$. The bias phenomenon is that the cube summation over $C_I(\vec{IV})$ has higher probability of being 0 than 1 for randomly chosen key $\vec{x}$:

$$\Pr_{\vec{x}}[p_{\vec{IV}}(\vec{x}) = 0] = 2^{-1} + \epsilon, \text{ where } \epsilon > 0. \tag{12}$$

The parameter ϵ is referred to as "bias", measuring the significance of the bias phenomenon. Specifically, zero-sum property is also a bias phenomenon with $\epsilon = 2^{-1}$; if $\epsilon = 0$, the summation is random 0-1 without bias.

In dynamic cube attacks, different key guesses $w \in [0, 2^\kappa)$ result in different superpolies $p_{\vec{IV}}^w(\vec{x})$ in the 1st output bit with different biases ϵ_w. A qualified cube I should detect a bias phenomenon satisfying both Proof 1 and 2 for distinguishing the correct key guess from the wrong ones. Therefore, we let ϵ_w be the bias of key guess $w \in [0, 2^\kappa)$, Dinur et al.'s bias phenomenon in [DGP+11] can be formalized as Definition 4.

Definition 4. (Dinur et al.'s Bias Phenomenon [DGP+11])

Proof 1 For the correct guess $w = 0$, the superpoly $p_{\vec{IV}}^0$ has bias $0 < \epsilon_0 < 2^{-1}$.

Proof 2 For all wrong guesses $w = [1, 2^\kappa)$, no bias can be detected: $\epsilon_w = 0$.

They find 51 qualified cube I's of dimension 49 or 50. For verifying Proof 1, they randomly picked 107 keys and, under the correct key guess $w = 0$, there are 8 of them have more than 50 zero-summations, indicating $0 < \epsilon_0 < 2^{-1}$. Although $\epsilon_0 > 0$ seems rational, a success probability of $8/107 \approx 7.5\%$ is far from good. Even for the 8 keys with significant bias, Fu et al.'s lesson is still reminding us that there is a possibility that such non-randomness remains for wrong guesses [FWDM18, HJL+18]. Proof 2 is only an idealized wrong-key assumption whose rationality deserves further discussions.

In our dynamic cube attack on full Grain-128 in Section 6, we are going to use large cube I's of dimension 90 satisfying: for correct key guess, the cubes have zero-sum property ($\epsilon = 2^{-1}$), a non-randomness provable with division property; for wrong guesses $w \in [1, 2^\kappa)$, the cube summations will result in superpoly $p^w_{\vec{IV}}$'s of extremely high algebraic degrees. Furthermore, we proposed division property based methods for evaluating the corresponding biases ϵ_w. Therefore, we can formalize our bias phenomenon with Proof 1 and 2 as Definition 5.

Definition 5. (Bias Phenomenon for Our Dynamic Cube Attacks)

Proof 1 For the correct key guess, there is zero-sum property: $\epsilon_0 = 2^{-1}$.

Proof 2 For the wrong guesses $w \in [1, 2^\kappa)$, the superpoly is a high-degree polynomial whose bias is smaller: $\max\left\{0, \epsilon_{[1,2^\kappa)}\right\} < 2^{-1} = \epsilon_0$.

Our evaluation of $\epsilon_{[1,2^\kappa)}$ indicates that the Proof 2 in Definition 4 doesn't hold for Grain-128. The last bit $s^{r_t}_{i_t}$ can be nullified only if all other state bits have been nullified before hand. Therefore, if 1 bit involved in $s^{r_1}_{i_1}$ is wrongly guessed, the nullification of many other state bits will fail even if all other $\kappa - 1$ key bits are correctly guessed. In this situation, the bias can be very small. On contrary, for $w = 1$, all other state bits will be nullified and the only not nullified bit $s^{r_t}_{i_t}$ will be constant 1, also of extremely low algebraic degree. In such a situation, the corresponding bias ϵ_1 can be quite significant, approaching ϵ_0. Therefore, we will regard an attack as successful if there is a theoretically evaluatable success probability of distinguishing the correct key guess $\vec{G}_0$ with key guess $\vec{G}_1$. In other words, in the remainder of this paper, we regard $w = 1$ as the representation of wrong key guesses for theoretic evaluations.

4 Evaluating the Bias Phenomenon

The theoretic evaluation of the bias phenomenon (Definition 5) requires a quantitative measurement to the ϵ in (6). Let $\mathbb{F}_2^n$ be the key space of $\vec{x}$ in (6). We first introduce a method for dividing $\mathbb{F}_2^n$ into a weak-key class and its complement so that the bias ϵ in (6) can be computed directly. Then, the concept of "split set" is introduced to identify a specific kind of weak-key classes so that the bias evaluation can be largely simplified based on a hypothesis. A method for constructing split sets heuristically is also introduced utilizing some algebraic properties of the superpolies.

4.1 Divide the Key Space with a Weak-Key Class

We suppose that the whole key space $\mathbb{F}_2^n$ can be divided into a weak-key class W and its complement $\overline{W}$ s.t.: for $\vec{x} \in W$, the superpoly $p_{\vec{IV}}(x)$ is constant 0; otherwise, for randomly chosen $\vec{x} \in \overline{W}$, $p_{\vec{IV}}(x)$ is randomly 0 or 1. Such a weak-key-W-division can be summarized as (13) as follows:

$$\Pr[p_{\vec{IV}}(\vec{x}) = 0 | \vec{x} \in T] = \begin{cases} 1, & T = W \\ 2^{-1}, & T = \overline{W} \end{cases} \tag{13}$$

With such a weak-key class W, we can easily prove Proposition 6.

Proposition 6. *For the superpoly $p_{\vec{IV}}(\vec{x})$ as defined in (4) and with key space $\mathbb{F}_2^n$ divided by a weak-key class W satisfying (13), the bias ϵ in (6) can be quantitatively measured as $\epsilon = |W|/2^{n+1}$ where $|W|$ is the size of W.*

Proof. Since the key space is $\mathbb{F}_2^n$, we know the complement $|\overline{W}| = 2^n - |W|$. With the knowledge of (13), (6) can be computed as

$$\Pr_{\vec{x}}[p_{\vec{IV}}(\vec{x}) = 0] = \frac{|W|}{2^n} \Pr_{\vec{x}\in W}[p_{\vec{IV}}(\vec{x}) = 0] + \frac{|\overline{W}|}{2^n} \Pr_{\vec{x}\in \overline{W}}[p_{\vec{IV}}(\vec{x}) = 0] = 2^{-1} + \frac{|W|}{2^{n+1}},$$

which completes the proof. □

4.2 The Bias Evaluation using the Split Set

Let $\Lambda = \{i_1, \ldots, i_\lambda\} \subseteq [0, n)$ be an index set. We can refer such a subset as the "split set" if it can define a specific weak-key class W_Λ of size $2^{n-|\Lambda|}$ as follows:

Definition 6. (Split Set) The superpoly $p_{\vec{IV}}$ is of the same setting as (4). We let Λ be an index set s.t. $\Lambda \subseteq [0, n)$. It defines a key class $W_\Lambda \subset \mathbb{F}_2^n$ as follows:

$$W_\Lambda = \{\vec{x} \in \mathbb{F}_2^n : \vec{x}[j] = 0 \text{ for all } j \in \Lambda\}. \tag{14}$$

Such Λ is referred to as a "split set" if the superpoly $p_{\vec{IV}}(\vec{x})$ in Definition 5 satisfies $p_{\vec{IV}}(\vec{x}) \equiv 0$ for all $\vec{x} \in W_\Lambda$.

The split sets with the smallest size are of specific importance for evaluating the bias phenomenon so we define them as the *minimal split set* in Definition 7.

Definition 7. (Minimal Split Set) Let Λ be a split set satisfying Definition 6. We call this Λ a minimal split set if, for all Λ' with size $|\Lambda'| < |\Lambda|$ and the corresponding weak-key class $W_{\Lambda'}$ defined as (14), there is $p_{\vec{IV}}(\vec{x}) \not\equiv 0$ for $\vec{x} \in W_{\Lambda'}$.

Definition 6 and 7 only guarantee half of (13). To make an evaluation to the bias, we make a further hypothesis that, for a minimal split set Λ, the weak-key class W_Λ satisfies the whole (13) as Assumption 1.

Assumption 1. *For a minimal split set Λ in Definition 7 and its corresponding weak-key class W_Λ as (13), we assume* $\Pr[p_{\vec{IV}}(\vec{x}) = 0 | \vec{x} \in \overline{W_\Lambda}] = 2^{-1}$.

With Assumption 1, the bias of $p_{\vec{IV}}(\vec{x})$ can be directly acquired using Proposition 6, which is formalized as Corollary 1.

Corollary 1. *Let the superpoly $p_{\vec{IV}}$ have a minimal split set Λ as in Definition 7 where Λ also satisfies Assumption 1. The bias phenomenon of $p_{\vec{IV}}(\vec{x})$ in (6) can be evluated as follows:*

$$\Pr_{\vec{x}}[p_{\vec{IV}}(\vec{x}) = 0] = 2^{-1} + 2^{-(|\Lambda|+1)}. \tag{15}$$

Proof. With Definition 6, we know that $\Pr[p_{\vec{IV}}(\vec{x}) = 0 | \vec{x} \in W_\Lambda] = 1$. With Assumption 1, we know that $\Pr[p_{\vec{IV}}(\vec{x}) = 0 | \vec{x} \in \overline{W_\Lambda}] = 2^{-1}$. Therefore W_Λ satisfies (13) and $|W_\Lambda| = 2^{n-|\Lambda|}$. According to Proposition 6, we have $\epsilon = 2^{-(|\Lambda|+1)}$ which completes the proof. □

Our theoretical deductions are mostly based on Assumption 1 and Corollary 1. In fact, we will see in Example 3 of Section 6 that the evaluation in Corollary 1 is more of a lower bound. The reason is quite simple. For a superpoly $p_{\vec{IV}}$, the minimal split set Λ is not necessarily unique. As a result, one minimal split set only can capture the largest zero-sum keys while the rest of the key-space may still contain weak-key x's corresponding to other split sets s.t. $p_{\vec{IV}}(\vec{x}) = 0$. Therefore, for one minimal split set Λ, there should be more $\vec{x} \in \overline{W_\Lambda}$ making $p_{\vec{IV}}(\vec{x}) = 0$ than those making $p_{\vec{IV}}(\vec{x}) = 1$. Their rationalities for using such Assumption 1 and Corollary 1 on Grain-128 will be discussed in detail in Section 7.

4.3 Algebraic Properties of Superpolies

The division property method given by Wang et al. in [WHT+18] exploits the algebraic degree of the superpoly $p_{I\vec{V}}(\vec{x})$. Therefore, we introduce a (heuristic) method for finding a split set Λ utilizing similar algebraic properties so that Wang et al.'s method can be applied directly.

As a polynomial over the ring $\mathbb{F}_2[x_{[0,n)}]$, the superpoly $p_{I\vec{V}}(\vec{x})$ can be split into sub-polynomials over subrings according to an arbitrary set $\Lambda = \{i_1, \ldots, i_\lambda\} \subseteq [0, n)$. With such Λ, the corresponding vector $\vec{k}_\Lambda \in \mathbb{F}_2^n$ has divided the space $\mathbb{F}_2^n$ into 2 parts:

$$\mathcal{U}_\Lambda := \left\{\vec{u} \in \mathbb{F}_2^n : \vec{u} \wedge \vec{k}_\Lambda = \vec{0}\right\} \text{ and } \mathcal{W}_\Lambda := \left\{\vec{w} \in \mathbb{F}_2^n : \vec{w} \wedge \vec{k}_\Lambda \neq \vec{0}\right\}. \tag{16}$$

Following (1), the ANF of the superpoly $p_{I\vec{V}}(\vec{x})$ can be represented as (17).

$$p_{I\vec{V}}(\vec{x}) = f_\Lambda + p_\Lambda = \sum_{\vec{k}\in\mathbb{F}_2^n} a_{\vec{k}}\vec{x}^{\vec{k}} = \sum_{\vec{u}\in\mathcal{U}_\Lambda} a_{\vec{u}}\vec{x}^{\vec{u}} + \sum_{\vec{w}\in\mathcal{W}_\Lambda} a_{\vec{w}}\vec{x}^{\vec{w}}, \tag{17}$$

where $f_\Lambda = \sum_{\vec{u}\in\mathcal{U}_\Lambda} a_{\vec{u}}\vec{x}^{\vec{u}}$ and $p_\Lambda = \sum_{\vec{w}\in\mathcal{W}_\Lambda} a_{\vec{w}}\vec{x}^{\vec{w}}$. Apparently, $f_\Lambda(\vec{x}) = 0$ for $\vec{x} \in W_\Lambda$ and $p_\Lambda \in \mathbb{F}_2^n[x_{[0,n)\setminus\Lambda}]$ is irrelevant to the variables x_Λ. If the Λ in (17) is further restricted to a split set, we know $p_\Lambda \equiv 0$ according to Definition 6 and $f_\Lambda = p_{I\vec{V}}$ accordingly. In order to utilize Assumption 1 and Corollary 1 for bias evaluations, we need to construct a split set with size as small as possible. So we propose a heuristic split set construction method as follows.

4.4 Heuristic Construction of Split Sets

If $p_{I\vec{V}}(\vec{x})$ itself is constant 0 ($\deg(p_{I\vec{V}}) = -1$), we can simply conclude that the empty set $\Lambda = \phi$ is the minimal split set. If $p_{I\vec{V}}(\vec{x}) \equiv 1$ ($\deg(p_{I\vec{V}}) = -1$), it cannot have a split set. For non-constant $p_{I\vec{V}}(\vec{x})$ ($\deg(p_{I\vec{V}}) \geq 1$), our heuristic method start from an empty set $\Lambda = \phi$ and construct the final split set gradually by adding the most promising element to the current Λ.

We consider the specific case of (17) where Λ only contains 1 element $\theta \in [0, n)$. Then, we know that the f_Λ can be represented as

$$f_\Lambda = f_{\{\theta\}} = x_\theta \cdot q_{\phi\cup\{\theta\}} \text{ where } q_{\phi\cup\{\theta\}} \in \mathbb{F}_2^n[x_{[0,n)\setminus\Lambda}].$$

The superpoly in (17) can be represented

$$p_{I\vec{V}}(\vec{x}) = x_\theta \cdot q_{\{\theta\}} + p_{\{\theta\}},$$

where both $q_{\{\theta\}}, q_{\{\theta\}}$ are polynomials of $\mathbb{F}_2^n[x_{[0,n)\setminus\Lambda}]$. We evaluate the algebraic degree of $p_{\{\theta\}}$. If $\deg(p_{\{\theta\}}) = -1$, we know that $\Lambda = \{\theta\}$ is the minimal split set; if $\deg(p_{\{\theta\}}) = 0$, such superpoly do not have a split set; otherwise, we pick a new θ and rerun the procedure. When the θ traverse all $[0, n)$ and all $p_{\{\theta\}}$'s satisfy $\deg(p_{\{\theta\}}) > 0$, we know that minimal split sets for such superpoly $p_{I\vec{V}}(\vec{x})$ are of size larger than 1. So we need to consider Λ's of size $2, 3, 4 \ldots$. Of course, traversing all possible $2, 3, 4\ldots$-element Λ's can be computationally infeasible. The time saving strategy for our heuristic method is that we select θ s.t. $p_{\{\theta\}}$ having the lowest algebraic degree and add it in Λ. Then, we start from $\Lambda = \{\theta\}$, we further search the minimal split set for $p_{\{\theta\}}$ following the same procedure. As a result, our heuristic for split set constructions can be summarized as follows:

1. Let $\Lambda_0 \leftarrow \phi$ and $p_{\Lambda_0} = p_{I\vec{V}}(\vec{x})$.

2. For $j = 1, 2, \ldots,$

(a) For all $\theta \in [0, n)\backslash\Lambda_{j-1}$, rewrite $p_{\Lambda_{j-1}} = x_\theta \cdot q_{\Lambda_{j-1}\cup\{\theta\}} + p_{\Lambda_{j-1}\cup\{\theta\}}$ and evaluate the algebraic degree $\deg(p_{\Lambda_{j-1}\cup\{\theta\}})$.

(b) $\Lambda_j \leftarrow \Lambda_{j-1} \cup \{i_j\}$ where $i_j \in [0, n)$ satisfies

$$\deg(p_{\Lambda_{j-1}\cup\{i_j\}}) = \min_{\theta\in[0,n)\backslash\Lambda_{j-1}} \deg(p_{\Lambda_{j-1}\cup\{\theta\}}). \tag{18}$$

(c) If $\deg(p_{\Lambda_j})$ equals to -1 or 0, $\Lambda \leftarrow \Lambda_j$ and break.

3. If $\deg(p_\Lambda) = -1$, Λ is a split set[7].

Such a heuristic split set construction algorithm can be carried out easily using Wang et al.'s division property based degree evaluation technique in [WHT+18]. The generated Λ satisfies Definition 6. Furthermore, such Λ is usually of small size and quite likely to be a minimal split set as Definition 7: enabling use to provide theoretically evaluated success probabilities based on Assumption 1 and Corollary 1. Like all heuristic algorithms, our method has its limitation: it is possible that the heuristically generated Λ may not be minimal. Our method can only exhaust all Λ's of size 1 so the result may not be optimal especially for the superpolies having much larger sized minimal split sets. So, in most cases, the Λ's constructed by our heuristic algorithm can only provide an approximation to the bias provided by the minimal split set in Corollary 1. Fortunately, as can be seen in Section 6, all the cubes used in our attacks have very small sized minimal split sets making our heuristic method applicable to Grain-128. We will detail the limitation of our heuristic method as well as the feasibility of our dynamic cube attack in Section 7.2. The idea and limitation of our Λ-construction algorithm are demonstrated in the following Example 1.

Example 1. As an example, we choose $p_{IV}(\vec{x}) = x_0x_1x_2 + x_1x_3 + x_2x_4$. With $i_1 = 1$, we have $p_{\Lambda_1} = x_2x_4$ and $\deg(p_{\Lambda_1}) = 2$, followed by $i_2 = 2$ making $p_{\Lambda_2} \equiv 0$. So we have $\Lambda = \{1, 2\}$ as a split set. The changed order $(i_1, i_2) = (2, 1)$ will have $p_{\Lambda_1} = x_1x_3$ and $\deg(p_{\Lambda_1}) = 2$ but $p_{\Lambda_2} \equiv 0$ remains. Both $(1, 2)$ and $(2, 1)$ satisfy $\deg(p_{\Lambda_1}) = 2$ and can be generated following our heuristic in (18). A different strategy, such as picking 3 or 4 first, may lead $|\Lambda| > 2$. However, our heuristic cannot guarantee a minimal Λ as well: if $i_1 = 0$, we have $\deg(p_{\Lambda_1}) = 2$ but it cannot make $|\Lambda| = 2$.

Note that Example 1 is only a toy case showing the limitation of the heuristic split set construction technique. So we use simple p_{IV} involving very few key bits. In the case of Grain-128, p_{IV}'s are extremely complicated: having high algebraic degree over 39 and involving almost all 128 key bits. The heuristic method can still construct Λ of size 2 or 3 but the corresponding $p_{\Lambda_1}, p_{\Lambda_2}$ will become much more complicated polynomials. Therefore, in the case of our dynamic cube attack on Grain-128, it is rational to assume that, for randomly chosen $\vec{x}$, which is not in the weak-key class captured by the minimal split set, the probability for a $p_{\Lambda_j}(\vec{x})$ being 0 or 1 is approximately 2^{-1}.

5 Division Property Description of Dynamic Cube Attacks

In this section, we introduce the division property description of the dynamic cube attack. According to Section 2.5, the nullification strategy $N_S = \{(l_1, s_{i_1}^{r_1}), \ldots, (l_t, s_{i_t}^{r_t})\}$ affects both the initial DP structure $\vec{k}_x, \vec{k}_v$ and the DP structures $\vec{k}^{r_1}[i_1], \ldots, \vec{k}^{r_t}[i_t]$ corresponding to the nullified state bits $s_{i_1}^{r_1}, \ldots, s_{i_t}^{r_t}$.

[7] If $\deg(p_\Lambda) = 0$, we know that, for randomly chosen $\vec{x}$, $p_{\vec{IV}}(\vec{x})$ can also be constant 1. This is also a non-random property but such a non-randomness has never been detected in any cube attack yet especially for Grain-128, as analyzed in [DS11]. So we do not consider such Λ's here.

5.1 The `ModelConstruct` and `InitDP` for Dynamic Cube Attacks

The init $\vec{k}_x, \vec{k}_v$ is affected because the IVs $\vec{v}[l_j]$ are to be changed from v_{l_j} to a polynomial $f_j^w \in \mathbb{F}_2[x_{[0,n)}, v_I]$ decided by N_S and the key guess $\vec{G}_w$ ($j = 1, \ldots, t$, $w \in [0, 2^\kappa)$). Each polynomial f_j can be regarded as a summation of monomials composed of variables $x_{[0,n)}, v_I$. We encode each monomial (including constant 1's) into a numeric sequence as follows:

$$\begin{array}{c} x_{i_1} x_{i_2} \cdots x_{i_s} v_{j_1} v_{j_2} \cdots v_{j_t} \to (i_1, i_2, \ldots, i_s, n + j_1, n + j_2, \ldots, n + j_t) \\ 1 \to (-1) \end{array}. \tag{19}$$

Therefore, for $f \in \mathbb{F}_2[x_{[0,n)}, v_{[0,m)}]$, we can construct the set S_f containing all the numeric sequences corresponding to all the monomials of f:

$$S_f := \left\{ (i_1, \ldots, i_s, n + j_1, \ldots, n + j_t) : \begin{array}{c} x_{i_1} \cdots x_{i_s} v_{j_1} \cdots v_{j_t} \\ \text{is a monomial of } f \end{array} \right\}. \tag{20}$$

For the wrong guesses $\vec{G}_w$ ($w \in [1, 2^\kappa)$), the corresponding ANF of the dynamic values will become $f_j^w = f_j^0 + g_j^w$, where $g_j^w \in \mathbb{F}_2[v_I]$ for $j = 1, \ldots, t$.

Algorithm 2 The division property of initial key and IV bits.

```
1:  procedure IniDP(MILP model $\mathcal{M}$, cube index set $I$, non-cube IV assignment $\vec{IV}$, initialization round number $R$, nullification strategy $N_S$, guess number $w \in [0, 2^\kappa)$, etc. )    ▷ split set $\Lambda$,
2:      Declare the DP structures $\vec{k}_x^0, \vec{k}_v^0$ of lengths $n$ and $m$ respectively.
3:      For $i \in I$, set $\mathcal{M}.con \leftarrow \vec{k}_v^0[i].val = 1$ and $\vec{k}_v^0[i].F = \delta$.
4:      For $i \notin I$, set $\mathcal{M}.con \leftarrow \vec{k}_v^0[i].val = 0$ and assign the flag value according to $\vec{IV}[i]$: $\vec{k}_v^0[i].F = 1_c$ if $\vec{IV}[i] = 1$ and $\vec{k}_v^0[i].F = 0_c$ if $\vec{IV}[i] = 0$.
5:      For $j \in [0, n)$, set $\vec{k}_x^0[j].F = \delta$.    ▷ Set $\mathcal{M}.con \leftarrow \vec{k}_x^0[\Lambda].val = 0$ and $\vec{k}_x^0[\Lambda].F = 0_c$. Set $\vec{k}_x^0[j].F = \delta$ for $j \notin \Lambda$.
6:      According to $N_S$ and $w$, deduce the ANFs of the dynamic values $f_1^w, \ldots, f_t^w$.
7:      Encode $f_1^w, \ldots, f_t^w$ to the corresponding sets defined in (20) and denote them as $S_1, \ldots, S_t$.
8:      for $\alpha = 1, \ldots, t$ do
9:          Update $(\mathcal{M}, \vec{k}_x^\alpha, \vec{k}_v^\alpha) \leftarrow$ PolyDiv$(\mathcal{M}, \vec{k}_x^{\alpha-1}, \vec{k}_v^{\alpha-1}, l_\alpha, S_\alpha)$.
10:     end for
11:     return $(\mathcal{M}, \vec{k}_x^t, \vec{k}_v^t)$.
12: end procedure
```

With the nullification strategy N_S and integer $w \in [0, 2^\kappa)$ identifying key guess $\vec{G}_w$ along with the split set Λ defined in Definition 6, the procedure $(\vec{k}_x, \vec{k}_v) \leftarrow$ `IniDP`$(\mathcal{M}, I, \vec{IV}, R,$ etc.$)$ used in `DegEval` in Algorithm 1 can now be specified as Algorithm 2. The subroutine `PolyDiv` is defined in Algorithm 3, handling the DP structure of 1 dynamic value.

The DP structures $\vec{k}^{r_1}[i_1], \ldots, \vec{k}^{r_t}[i_t]$ should reflect the key guess $w \in [0, 2^\kappa)$ and the effect of nullification. According to the analysis in Section 3, we only consider the correct key guess $w = 0$ and the wrong guess $w = 1$. During the implementation of the `ModelConstruct` in Algorithm 1, if $w = 0$ makes all $s_{i_1}^{r_1}, \ldots, s_{i_t}^{r_t}$ nullified, we should additionally set their flag values to $\vec{k}^{r_j}[i_j].F = 0_c$ and add constraints $\mathcal{M}.con \leftarrow \vec{k}^{r_j}[i_j].val = 0$ for $j = 1, \ldots, t$. As to the wrong key guesses $w = 1$, we have $s_{i_1}^{r_1} = \ldots = s_{i_{t-1}}^{r_{t-1}} = 0$ and $s_{i_t}^{r_t} = 1$. Reflecting the DP structures, the division property values are $\mathcal{M}.con \leftarrow \vec{k}^{r_j}[i_j].val = 0$ for $j = 1, \ldots, t$ and the flag values are $\vec{k}_{i_1}^{r_1}.F = \ldots = \vec{k}_{i_{t-1}}^{r_{t-1}}.F = 0_c$, $\vec{k}_{i_t}^{r_t}.F = 1_c$. With all factors, the `ModelConstruct` procedure in Algorithm 1 can now be described as Algorithm 4.

5.2 Selecting Qualified Cubes for Dynamic Cube Attacks

With such division-property-based tools, we can theoretically analyze a dynamic cube attack for not only Proof 1 but also Proof 2 in Definition 5. Proof 1 specifies the zero-sum property ($\epsilon_0 = 2^{-1}$). Such a property can be checked by the degree evaluation technique as $-1 = d_0 =$ `DegEval`$(I, \vec{IV}, N_S, w = 0)$. Proof 2 indicates that $p_{\vec{IV}}^w(\vec{x}) \neq 0$ for $w = 1$

Algorithm 3 Evaluate the division property of a polynomial encoded as S_f.

```
procedure PolyDiv(MILP model $\mathcal{M}$, DP structures for key bits $\vec{k}_x$ and current IV values $\vec{k}_v$, the targeted dynamic IV index $l_\alpha$, the encoded ANF of the dynamic value $S_\alpha$ defined as (20). )
    Initialize an empty set $T$.
    for $\vec{\gamma} \in S_\alpha$ do
        Call the subroutine algorithm $(\mathcal{M}, \vec{k}_x, \vec{k}_v, o) \leftarrow$ MonoDiv$(\mathcal{M}, \vec{k}_x, \vec{k}_v, \vec{\gamma})$.
        Add $o$ in $T$.
    end for
    If $|T| = 1$ and $T = \{u\}$, update $\vec{k}_v[l_\alpha] \leftarrow u$; otherwise, call $(\mathcal{M}, u) \leftarrow$ xorf$(\mathcal{M}, T)$ and assign $\vec{k}_v[l_\alpha] \leftarrow u$.
    return $(\mathcal{M}, \vec{k}_x, \vec{k}_v)$.
end procedure
```

```
procedure MonoDiv(MILP model $\mathcal{M}$, DP structures of current key $\vec{k}_x$ and IV $\vec{k}_v$, a monomial encoded as sequence $\gamma = (i_1, i_2, \dots, i_s, n + j_1, n + j_2, \dots, n + j_t)$ or $\gamma = (-1)$.)
    If the sequence is $(-1)$, declare a DP structure $o$ s.t. $\mathcal{M}.con \leftarrow o.val = 0$, $o.F \leftarrow 1_c$ and return $(\mathcal{M}, \vec{k}_x, \vec{k}_v, o)$.
    Initialize an empty set $P$.
    for $\mu = i_1, i_2, \dots, i_s, n + j_1, n + j_2, \dots, n + j_t$ do
        if $\mu < n$ then
            $(\mathcal{M}, x'_\mu, x^\star_\mu) \leftarrow$ copyf$(\mathcal{M}, \vec{k}_x[\mu], 2)$.
            Add $x'_\mu$ into $P$ and update $\vec{k}_x$ as $\vec{k}_x[\mu] \leftarrow x^\star_\mu$.
        else
            $(\mathcal{M}, v'_{\mu-n}, v^\star_{\mu-n}) \leftarrow$ copyf$(\mathcal{M}, \vec{k}_v[\mu - n], 2)$.
            Add $v'_{\mu-n}$ into $P$ and update $\vec{k}_v$ as $\vec{k}_v[\mu - n] \leftarrow v^\star_{\mu-n}$.
        end if
    end for
    If $P$ only contains 1 element, we denote it as $\vec{P} = \{o\}$; otherwise, we call $(\mathcal{M}, o) \leftarrow$ andf$(\mathcal{M}, P)$.
    return $(\mathcal{M}, \vec{k}_x, \vec{k}_v, o)$.
end procedure
```

Algorithm 4 Evaluate the division property of a polynomial encoded as S_f.

```
procedure ModelConstruct($\mathcal{M}$, DP structures for keys and IVs $\vec{k}_x, \vec{k}_v$, round $R$, nullification strategy $N_S = \{(l_1, s_{i_1}^{r1}), \dots, (l_t, s_{i_t}^{rt} s)\}$, guess number $w \in [0, 2^\kappa)$.)
    Initialize the division property of the initial state as $\vec{k}^0$ according to $\vec{k}_x, \vec{k}_v$.
    for $r = 1, \dots, R$ do
        Update $\mathcal{M}$ as $(\mathcal{M}, \vec{k}^r) \leftarrow$ UpdDiv$(\mathcal{M}, \vec{k}^{r-1}, r, N_S, w)$ capturing the division trail $\vec{k}^{r-1} \xrightarrow{\texttt{Upd}} \vec{k}^r$ using copyf, andf, xorf in Proposition 3, 5 and 4.
        if There is $j \in \{1, \dots t\}$ s.t. $r = r_j$ then
            Update $\mathcal{M}$ as $\mathcal{M}.con \leftarrow \vec{k}^r[i_j].val = 0$.
            If $w = 0$ or $w \neq 0 \bigwedge j \neq t$, $\vec{k}^r[i_j].F = 0_c$; otherwise, $\vec{k}^r[i_j].F = 1_c$.
        end if
    end for
    Update model $\mathcal{M}$ according to Output to complete the division trail $\vec{k}^0 \rightarrow \vec{k}^R \| o$.
    Add constraints $\mathcal{M}.con \leftarrow \vec{k}^R \| o = \vec{0} \| 1$.
    Return $(\mathcal{M}, \vec{k}^0, \dots, \vec{k}^R, o)$.
end procedure
```

so the degree evaluation should return $0 < d_1 = \texttt{DegEval}(I, \vec{IV}, N_S, w = 1)$. Proof 2 also requires that $p^1_{\vec{IV}}(\vec{x})$ has insignificant bias phenomenon: $\epsilon_1 < 2^{-1}$. According to Section 4, we need to find the minimum split set Λ for $p^1_{\vec{IV}}(\vec{x})$. Therefore, we realize the heuristic Λ-construction method in Section 4.4 as `DeteLam` defined in Algorithm 5. Note that `DeteLam` use `IniDP` as the subroutine. The additional parameter Λ of Algorithm 5 only affects `IniDP` in Algorithm 2 by replacing the statements colored in blue with the corresponding comments.

The algebraic degrees in (18) are evaluated with Wang et al.'s division property based algorithm `DegEval`. Such `DegEval` is defined in Algorithm 1 and its subroutines `IniDP`, `ModelConstruct` are defined as Algorithm 2 and 4 respectively.

Therefore, with a predefined nullification strategy N_S and non-cube IV assignment to $\vec{IV}$, a qualified cube I suitable for dynamic cube attacks can be constructed as follows:

1. Randomly construct a high-dimensional cube $I \subseteq [0, m) \backslash \{l_1, \dots, l_t\}$.
2. For $w = 0, 1$, evaluate the degree $d_0 \leftarrow \texttt{DegEval}(I, \vec{IV}, R, N_S, 0)$ and $d_1 \leftarrow \texttt{DegEval}(I, \vec{IV}, R, N_S, 1)$.
3. If $d_0 = -1$ and $d_1 > 0$, keep such I as a qualified cube and go to Step 4; otherwise, go back

Algorithm 5 A heuristic construction of the split set Λ.

```
procedure DeteLam(Cube indices $I$, non-cube IV assignment $\vec{IV}$, round $R$, nullification strategy $N_S$, key guess $w \in [0, 2^\kappa)$.)
    Initialize $\Lambda \leftarrow \phi$ and $d \leftarrow$ DegEval($I, \vec{IV}, R, N_S, w, \Lambda$).
    while $d \neq -1$ do
        Initialize a table $T$ containing $n$ entries and all entries are initialized as
            $T[i] = n$ for all $i \in [0, n)$.    (21)
        for $i \in [0, n)\backslash\Lambda$ do
            Compute $d_i =$ DegEval($I, \vec{IV}, R, N_S, w, \Lambda \cup \{i\}$) and set $T[i] \leftarrow d_i$.
        end for
        Search $T$ and pick an $i_j \in [0, n)\backslash\Lambda$ s.t. $T[i_j] = \min_{1\leq i\leq n}\{T[i]\}$.
        Update $\Lambda \leftarrow \Lambda \bigcup \{i_j\}$ and $d \leftarrow T[i_j]$.
    end while
    return $\Lambda$.
end procedure
```

to Step 1.

4. Let $w = 1$, heuristically build a split set Λ as $\Lambda \leftarrow$ `DeteLam`$(I, \vec{IV}, R, N_S, w)$.
5. If $|\Lambda|$ is large, keep such I as a qualified cube; otherwise, go back to Step 1.

The good cubes should be the ones satisfying $d_0 = 0 \ll d_1$ and $\Lambda > 1$, indicating that the superpoly $p^1_{\vec{IV}}$ corresponding to the wrong guess $w = 1$ is a high-degree polynomial rather than constant 0 and the bias of wrong guess is smaller than 2^{-2}. If we are able to find sufficiently many such cube I's, such a collection of cubes makes up a cube tester available for launching dynamic cube attacks. This is exactly the kind of cube we use in our attack on full Grain-128 in Section 6.

6 Dynamic Cube Attack on Full Grain-128 with Theoretic Evidence for Both Proof 1 and 2

The specification of Grain-128 is described in Appendix A. More details can be seen in [HJMM06] In accordance to [DGP+11], the bits in LFSR (NFSR) are denoted as $s_0, s_1, \ldots$ $(b_0, b_1, \ldots)$. For full Grain-128, the 1st output bit is computed after 256 initialization rounds so the keystream bits are $z^{256}, z^{257}, \ldots$. The corresponding division property propagation is described by the MILP models in Appendix B.

The 1st and most important step of the dynamic cube attack is setting the nullification strategy N_S, determining which intermediate state bit is nullified and how. In [DGP+11], Dinur et al.'s terminal goal is to nullify b_{203}. But the ANF of b_{203} is too complicated involving too many key bits to be guessed. Therefore, they nullify other intermediate state bits to simplify b_{203}. At last, their nullification strategy contains 13 pairs requiring to guess $\kappa = 39$ key bits. They use 51 I's consisting of one 50-dimensional cube along with its 49-dimensional sub-cubes. According to [DGP+11], for the correct key guess $\vec{G}_0$, more than 50 out of the 51 cubes can have zero-sum property with probability $8/107 \approx 7.5\%$. But the bias of wrong key guesses is never tested, so whether $\epsilon_0 \gg \max\{0, \epsilon_{[1,2^{39})}\}$ is unknown.

In comparison, our N_S is quite simple only containing 1 pair: $N_S = \{(l_1, b_{158})\}$. We find that when b_{158} is nullified by setting IV bits at position $l_1 = 30$ or $l_1 = 90$ to dynamic values, the superpoly of the cube summations over $I = [0, n)\backslash\{30\}$ (or $I = [0, n)\backslash\{90\}$) at round 256 is constant. Another good news is that the dynamic values for the two l_1's share the same key guesses only involving $\kappa = 3$ bits, since the ANFs of the two dynamic values are as follows:

$$\begin{aligned} f_{30} &= \mathbf{g_0} + \mathbf{x_{42}}v_{38} + \mathbf{x_{125}}v_{72} + v_{43}v_{50} + v_{90}, \\ f_{90} &= \mathbf{g_0} + \mathbf{x_{42}}v_{38} + \mathbf{x_{125}}v_{72} + v_{43}v_{50} + v_{30}, \end{aligned}$$

where g_0 is a polynomial of key bits:

$$g_0 = x_{30} + x_{32} + x_{33}x_{97} + x_{41}x_{43} + x_{42}x_{125} + x_{45} + x_{47}x_{48} + x_{56} + x_{57}x_{89} + x_{66} + \\ x_{70}x_{78} + x_{75} + x_{86} + x_{91}x_{95} + x_{94} + x_{98}x_{114} + x_{103} + x_{119} + x_{121} + x_{126} + 1.$$

So the correct key guess should be defined as $\vec{G}_0 = (g_0, x_{42}, x_{125})$. Furthermore, when the wrong guess number $w = 1$ ($\vec{G}_1[0]$ is wrongly guessed as $g_0 + 1$, x_{42} and x_{125} are correctly guessed), we have $b^1_{158} \equiv 1$. For $w > 1$, the corresponding ANF b^w_{158}'s are even more complicated. Therefore, we run the procedure in Section 5.2 and find some qualified cubes of dimension 90, denoted as $I_1, \ldots, I_{29}$ in Table 3. As can be seen, such cubes not only have large d_1's ($d_1 \geq 39 > d_0 = 0$) but split sets of size $|\Lambda| \geq 2$ as well. This indicates that for wrong guesses $w \in [1, 2^\kappa)$, the superpoly $p^w_{\vec{IV}}(\vec{x})$ is a polynomial of degree $\deg(p_{\vec{IV}}) \geq 39$ and, if Assumption 1 holds, the bias can be evaluated as $\epsilon_w \in [2^{-4}, 2^{-3}]$ according to Corollary 1. Such Λ's are to be used to theoretically evaluate the success probability, denoted as P_S^{TP}, of our dynamic cube attacks. The procedure of our dynamic cube attack is quite simple. For the 29 90-dimensional cubes in Table 3, we select N of them, denoted as $I_{i_1}, \ldots, I_{i_N}$ ($1 \leq i_1 < \cdots < i_N \leq 29$), and do the following steps:

1. Guess g_0, x_{42} and x_{125}.
2. For each guess, sum over $C_{I_{i_1}}(\vec{0}), \ldots, C_{I_{i_N}}(\vec{0})$ while $\vec{v}[l_1]$ is assigned to the corresponding dynamic value.
3. If any of the N summations is non-zero, the key guess is wrong; otherwise, keep the key guess as a correct key guess candidate.

Complexity and Success Probabilities The time and data complexities of the attack are both $Comp = 2^3 \cdot N \cdot 2^{90} = N \cdot 2^{93}$. The memory complexity is only 2^3 counters of $\lceil \log N \rceil$ bits, which is negligible. Since the correct key guess makes sure N 0-summations, evaluating the success probability P_S is equivalent to evaluating the probability for a wrong key guess to generate N zero summations. In Dinur et al.'s idealized setting in Definition 4, $\epsilon_{[1,2^\kappa)} = 0$ so the probability of having N 0-summations is 2^{-N} and such idealized success probability can be easily evaluated as $P_S^{ideal} = 1 - 2^{-N}$ which is demonstrated in the 2nd line of Table 4. On the other hand, our evaluation of P_S takes the split set Λ into account. We use the worst case $\epsilon_{[1,2^\kappa)} = 2^{-3}$, so our theoretically evaluated success probability, based on Assumption 1 and Corollary 1, is $P_S^{TP} = 1 - (5/8)^N$ when N which is demonstrated in the 3rd line of Table 4.

Table 3: The qualified 90-dimensional cubes used for attacking 256-round Grain-128. The nullification strategy is $N_S = \{(l_1, b_{158})\}$ where $l_1 = 90$ for $I_1, \ldots, I_{15}$ and $l_1 = 30$ for the others.

i	I_i for $l_1 = 90$	d_1	Λ	i	I_i for $l_1 = 30$	d_1	Λ
1	$[0,96)\backslash\{39,40,41,43,50,90\}$	42	36,81,85	16	$[0,96)\backslash\{30,39,40,41,45,50\}$	39	38,81,125
2	$[0,96)\backslash\{39,40,41,45,50,90\}$	39	65,81,125	17	$[0,96)\backslash\{21,30,39,40,41,43\}$	45	81,89,125
3	$[0,96)\backslash\{21,39,40,41,43,90\}$	45	38,81,125	18	$[0,96)\backslash\{25,30,39,40,41,43\}$	44	52,81,125
4	$[0,96)\backslash\{25,39,40,41,43,90\}$	44	81,91,125	19	$[0,96)\backslash\{30,39,40,41,43,44\}$	41	81,89,125
5	$[0,96)\backslash\{30,39,40,41,43,90\}$	44	57,81,125	20	$[0,96)\backslash\{30,39,40,41,43,49\}$	44	77,81,125
6	$[0,96)\backslash\{39,40,41,43,44,90\}$	41	38,81,125	21	$[0,96)\backslash\{30,39,40,41,43,50\}$	42	69,81,125
7	$[0,96)\backslash\{39,40,41,43,49,90\}$	44	77,81,125	22	$[0,96)\backslash\{30,39,40,41,43,50\}$	42	58,81,125
8	$[0,96)\backslash\{39,40,41,43,50,90\}$	42	58,81,125	23	$[0,96)\backslash\{30,40,41,43,44,50\}$	42	69,81,125
9	$[0,96)\backslash\{39,40,41,43,54,90\}$	45	57,81,125	24	$[0,96)\backslash\{30,39,40,41,43,54\}$	45	81,106,125
10	$[0,96)\backslash\{39,40,41,43,61,90\}$	44	54,81,125	25	$[0,96)\backslash\{30,39,40,41,43,61\}$	44	50,81,125
11	$[0,96)\backslash\{39,40,41,43,72,90\}$	43	81,125	26	$[0,96)\backslash\{30,39,40,41,43,72\}$	43	81,125
12	$[0,96)\backslash\{39,40,41,43,87,90\}$	44	81,125	27	$[0,96)\backslash\{30,39,40,41,43,87\}$	44	81,125
13	$[0,96)\backslash\{39,40,41,43,89,90\}$	43	81,91,125	28	$[0,96)\backslash\{30,39,40,41,43,89\}$	43	81,91,125
14	$[0,96)\backslash\{39,40,41,43,90,94\}$	43	81,125	29	$[0,96)\backslash\{30,39,40,41,43,94\}$	43	81,125
15	$[0,96)\backslash\{40,41,43,44,50,90\}$	42	69,81,125				

Practical Verifications We verify the correctness of our method with practically computable cube I's for round reduced Grain-128. Following the procedure in Section 4.4,

Table 4: The complexities and success probabilities for attacking 256-round Grain-128 using different cube number N's. P_S^{ideal} is based on Dinur et al.'s idealized wrong-key hypothesis in Definition 4 while P_S^{TP} is our theoretically evaluated success probability based on Assumption 1 and Corollary 1

N	1	2	3	4	5	6	7	8	9	10
Comp	93.00	94.00	94.58	95	95.32	95.58	95.81	96.00	96.17	96.32
P_S^{ideal}	0.78%	13.35%	39.27%	63.65%	80.07%	89.56%	94.66%	97.30%	98.64%	99.32%
P_S^{TP}	0.01%	0.31%	2.16%	6.98%	15.01%	25.36%	36.68%	47.79%	57.90%	66.65%
N	11	12	13	14	15	16	17	18	19	20
Comp	96.46	96.58	96.70	96.81	96.91	97.00	97.09	97.17	97.25	97.32
P_S^{ideal}	99.66%	99.83%	99.91%	99.96%	99.98%	99.99%	99.99%	100.00%	100.00%	100.00%
P_S^{TP}	73.93%	79.83%	84.51%	88.17%	91.01%	93.19%	94.86%	96.12%	97.08%	97.80%
N	21	22	23	24	25	26	27	28	29	
Comp	97.39	97.46	97.52	97.59	97.64	97.70	97.75	97.81	97.86	
P_S^{ideal}	100.00%	100.00%	100.00%	100%	100.00%	100.00%	100.00%	100.00%	100.00%	
P_S^{TP}	98.35%	98.76%	99.07%	99.30%	99.47%	99.61%	99.70%	99.78%	99.83%	

we acquire the cube I's satisfying $d_0 = 0, d_1 > 0$ but with split set Λ of different sizes by calling Algorithm 5. Then, for key guess $w = 0$ and $w = 1$, we practically evaluate the corresponding biases ϵ_0, ϵ_1 respectively with sufficiently many random keys. We find that the zero-sum property $\epsilon_0 = 2^{-1}$ is satisfied whenever $d_0 = 0$. But Dinur et al.'s wrong-key hypothesis $\epsilon_1 = 0$ (Proof 2 of Definition 4) is incorrect. We present Example 2 and 3: the former can be regarded as a disproof to Definition 4 while the latter can be regarded as a support to our theories in Section 4.2: Assumption 1 based Corollary 1 serves as a lower bound to the bias phenomenon. It remains to be discussed whether Corollary 1 based bias evaluations can be used as success probability evaluation to dynamic cube attacks, as we will show immediately in Section 7.

Example 2. For $R = 179$, $\vec{IV} = \vec{0}$ and $N_S = \{(90, b_{158})\}$, following the procedure of Section 5.2, we construct two 9-dimensional cubes $I^1 = \{15, 31, 33, 34, 44, 65, 79, 86, 88\}$ and $I^2 = \{5, 6, 9, 37, 40, 41, 43, 64, 67\}$ sharing $d_0^1 = d_0^2 = 0$ and $d_1^1 = d_1^2 = 18$. The experimentally acquired statistics support Proof 1 in Definition 5 as all randomly chosen keys have zero-sum property for the correct key guess $w = 0$. For Proof 2, we evaluate the actual bias of wrong guesses as $\epsilon_1^1 = 0.165$ and $\epsilon_1^2 = 0.464$: both significantly larger than 0, violating Dinur et al.'s wrong-key hypothesis in Definition 4.

Example 3. For the I_1, I_2 in Example 2 with the same R, $\vec{IV}$ and N_S settings. We call Algorithm 5 only finding that their split set Λ's for wrong key $w = 1$ are different: $\Lambda^1 = \{37, 125\}$ and $\Lambda^2 = \{75\}$. According to Corollary 1, such a difference $|\Lambda^1| = 2 > |\Lambda^2| = 1$ indicates that, for wrong guesses, the bias of I^1 is less significant than that of I^2 making I^1 more suitable for dynamic cube attacks. The experimentally acquired statistics support our deductions: for sufficiently many randomly chosen keys, we evaluate the actual bias of wrong guesses as $\epsilon_1^1 = 0.165 \geq 2^{-3} = 0.125$ and $\epsilon_1^2 = 0.464 \geq 2^{-2} = 0.25$. This is in accordance with our theories in Section 4.2.

On the Accuracy of the Division Property. We use [YT19]'s method to verify the non-constant property of $p_{\vec{0}}^1$. We find that for arbitrary $0 \leq R_m \leq 256$, there are sufficiently many division trails within $R_m \to R$ in (10). We also practically check the ANF of $(\vec{b}^{R_m}, \vec{s}^{R_m})$ for $R_m \leq 210$ finding that all of them can add key bits to the superpolies. So the superpolies are unlikely to be constants. This is an additional theoretic evidence supporting the feasibility of our dynamic cube attack on full Grain-128.

7 Discussions and Comparisons

We discuss the unsolved issues (such as the rationality of Assumption 1, the limitations of our dynamic cube attacks etc.). We also make detailed comparisons between our attack with Dinur et al.'s in [DGP+11] and point out some promising improvements.

7.1 Probability Evaluation based on Assumption 1

Although our dynamic cube attack in Section 6 provides P_S^{TP}, such a theoretic evaluation of the success probability is based on Corollary 1, which can only be valid when Assumption 1 holds. According to the analysis in Section 4.2, for a single cube, the bias evaluation in Corollary 1 is more of a lower bound. However, in our practical experiments on Grain-128, we try many cube I's resembling the I_1, I_2 in Examples 2 and 3. Such I's have $d_0 = 0, d_1 > 0$ and a non-empty Λ. When we practically measure their bias, there is an interesting finding: their bias parameter ϵ in Definition 5 always falls into the range $[2^{-(|\Lambda|+1)}, 2^{-|\Lambda|}]$. If this is the case, the cubes with $|\Lambda| = 3$ will have bias $[2^{-4}, 2^{-3}]$ while those with $|\Lambda| = 2$ will be within $[2^{-3}, 2^{-2}]$. Since most of the cubes in Table 3 have $|\Lambda| = 3$ and the rest have $|\Lambda| = 2$, we believe it rational for us to use the median number 2^{-3} an approximate bias for success probability evaluation.

7.2 Limitation of Heuristic Split Set Construction and the Rationality of the Dynamic Cube Attack

As has been pointed out in Section 4.4 and demonstrated in Example 1, our heuristic split set construction cannot guarantee to generate the theoretically minimum Λ. But, it should also be noticed that the nature of our heuristic method will exhaustively try all Λ of size $|\Lambda| = 1$. Therefore, this at least guarantees that all cubes in Table 3 have split set sizes larger than 1. So we have $|\Lambda| \geq 2$ and the split sets with $|\Lambda| = 2$ in Table 3 should be regarded as optimal. During the Λ-construction process in Algorithm 5, we find that, after nullifying most of the key bits, the degree of the superpoly will not drop at all. Just as x_3, x_4 in Example 1, such key bits should not have priority for entering Λ. For the rest, it will be possible for us to exhaust all Λ's with $|\Lambda| = 2$ only to find that all 2-sized Λ cannot make p_Λ in (17) become constant 0. Therefore, the 3-sized Λ's in Table 3 are also optimal. Therefore, although it is not theoretically perfect, our heuristic split construction method in Section 4.4 works well in our dynamic cube attack on Grain-128.

7.3 Comparison with Dinur et al.'s Result

As can be seen in Section 6, our nullification strategy only requires to guess 3 key bits. Therefore, we are able to recover 3 key bits with complexity bounded by $2^{97.86}$ and success probability 99.83%. The remaining 125 key bits are supposed to be recovered through exhaustive search with complexity 2^{125}. On the contrary, Dinur et al.'s nullification strategy in [DGP+11] requires 39 bits of key guesses. They use a 50-dimensional cube along with its 49-dimensional subcubes so they are able to recover the 39 key bits with complexity $2^{93.71}$ and success probability only 7.5%. The remaining 89 key bits should also be exhausted with complexity 2^{89}, lower than that of their dynamic cube partial-key recovery. Our partial-key recovery has overwhelmingly better success probability but only 3 bits are recovered, which is obviously a limitation making the overall complexity high. To sum up, our attack can be regarded as a special kind of tradeoff of Dinur et al.'s: we trade a higher complexity for a better success probability and theoretic reliability.

7.4 Additional Comments on Dinur et al.'s Attack

Dinur et al. also claimed in [DGP+11] that some unspecified tradeoffs can be made to their attacks to improve the efficiency of their attack, but our analysis reveals many underlying problems. Using the same modeling method in Section 5, we are also able to evaluate Dinur et al.'s result with division property techniques. For each of the 51 cubes (a 50-dimensional one along with its 49-dimensional subcubes), we search for minimal split sets for both correct $w = 0$ and incorrect key guess $w = 1$. We find that such a cube-subcube combination is not optimal: there are unqualified cubes included. Examples can be seen in $I\backslash\{79\}$: both correct and incorrect key guess share the same Λ of size 78 indicating an insignificant bias for the correct key guess. Such subcube can be a distraction for identifying the correct key guess from wrong ones.

If a larger cube is used to make a more significant bias for the correct key guess, more key bits will be involved so that the number of key guesses will be skyrocketing because, in Dinur et al.'s nullification strategy in [DGP+11], the ANFs of the dynamic IVs have already mixed the key and IV bits quite thoroughly. Besides, larger cubes still cannot avoid the bad-subcube distractions.

On the other hand, if they abandon the cube-subcube strategy and use our random cube picking strategy in Section 5.2, they may still be confronted with an increased number of key guesses because different cubes will involve different key guesses. Furthermore, Dinur et al.'s complicated nullification strategy may result in that some of the key guesses are involved in very few randomly constructed cubes. When these key bits are wrongly guessed while others are correctly guessed, there will still be high probability of witnessing significant bias so that the success probability can be even lower. So abandoning the cube-subcube strategy can be hard for Dinur et al.'s attack.

Besides, according to our division property evaluations, with Dinur et al.'s nullification strategy, even the largest possible cube cannot provide zero-sum property for the correct key guess. So, it is hard for them to draw the threshold for the bias parameter to distinguish correct key guess from the wrong ones.

Faced with these problems, the unspecified tradeoffs in [DGP+11] may not bring too much improvements to the success probability or the average efficiency.

7.5 Future Improvements

According to the analysis above, we can see that Dinur et al.'s dynamic cube attack has lower complexity along with a limited success probability. Their complicated nullification strategy also makes it hard to find a good tradeoff. On the other hand, our attack has a much higher success probability for recovering the 3 key bits determined by a quite simple nullification strategy. But such a simple nullification strategy also makes the complexity for recovering all 128 key bits quite high. In other words, an extremely simple nullification strategy also has its disadvantage for involving too few key guesses. As a promising compromise, there might be a nullification strategy N_S that involve more than 3 key guesses. Along with such a N_S, there should also be highly qualified cubes that maintain a significant bias phenomenon for the correct key guess and far less significant one for the wrong guesses. Of course, designing the N_S and finding the corresponding cubes can be quite challenging as well.

8 Other Applications of the Split Sets

We now focus on n split sets each contain $n - 1$ elements denoted as $\Lambda_i := [0, n)\backslash\{i\}$ for $i \in [0, n)$. For round number R and some cube I, if $1 = \texttt{DegEval}(I, \vec{IV}, R, \Lambda_i)$, we know that x_i can appear in the linear part of the superpoly $p_{\vec{IV}}(\vec{x})$ of the output bit z_R; otherwise, x_i can only appear in the non-linear part. Therefore, we have two directions for extension:

1. Draw secure bounds for stream ciphers against the bias cube tester.

2. Minimize the $\tilde{J}^1$ generated by Algorithm 1.

The 1st extension is applied to the Grain-like primitives while the latter is applied to Kreyvium and Acorn for better key-recovery attacks.

8.1 Upper Bounds for Grain-Like Stream Ciphers Against Bias Testers

In [Liu17], Liu et al. draw upper bounds for Trivium-like stream ciphers against the zero-sum cube tester: they give a round number R s.t. the summation over the largest possible cube $I = [0, m)$ of the output bits after $(R-1)$-round initialization still have the zero-sum property but UNKNOWN for R and more rounds. Todo et al. verified and expanded the bounds soon afterwards in [TIHM17b].

According to existing practical results on Trivium-like primitives [ADMS09, LYWL18], as long as the superpoly is non-zero, the biases are usually not so significant as those for Grain-128 [ADH+09, DGP+11]. Therefore, drawing zero-sum bounds is quite meaningful. But for Grain-like stream ciphers, simply drawing zero-sum bounds is not enough: as can be seen in Section 6 as well as in previous cryptanalysis results [DGP+11, DS11, ADH+09], even if the superpoly is not constant 0, the cube summation may still have significant bias that can be utilized as an efficient cube tester. Therefore, it is necessary to draw bounds for Grain-like stream ciphers against the bias cube tester.

Corollary 1 has revealed that significant biases can be detected when there is a small split set Λ found by calling Algorithm 5. Therefore, for the largest cube $I = [0, m)$, the round number R should be the secure bound against bias testers only if it is the smallest integer satisfying $1 = \texttt{DegEval}(I, \vec{IV}, R, \Lambda_i)$ for all $\Lambda_i = [0, n) \setminus \{i\}$ where $i \in [0, n)$. This indicates that, even for the largest cube $I = [0, m)$ and the largest possible Λ's ($|\Lambda| = n - 1$), the corresponding superpoly may still have a non-constant p_Λ which is defined in (17).

Table 5: The Upper Bounds for Grain Family Against Bias Testers

Grain-128				Grain-128a			
Bound	Full	#IV	Type	Bound	Full	#IV	Type
254	256	96	0-sum	184	256	96	0-sum
265			bias	190			bias
Grain-V1				Plantlet			
Bound	Full	#IV	Type	Bound	Full	#IV	Type
80	160	64	0-sum	101	320	90	0-sum
82			bias	138			bias

Using this method, we are able to draw bias bounds for Grain-like stream ciphers, namely Grain-128, Grain-128a, Grain-V1 and Plantlet, in Table 5. As can be seen, 256-round initialization is not enough for Grain-128. Such a method can be regarded as an effective general tool for new stream ciphers for evaluating its security against bias cube testers. Note: since all IV indices are included in the cube I, there is no non-cube IVs making such results an application of manipulating the key flag values only.

On the Accuracy of the Division Property. Same with the situation of Grain-128, for arbitrary R_m's and the bound R, there are sufficiently many division trails within $R_m \to R$ and, using the ANF checking method in [YT19], we find most of such division trails are related to monomials involving key bits. Such monomials are in fact adding key bits to the superpolies so the bounds we draw are reasonable.

8.2 Improved Relaxed Term Enumeration with Smaller $\tilde{J}^1$

In [WHT+18], for predefined $I, \vec{IV}, R$, the degree d is first evaluated as $1 \leq d = \texttt{DegEval}(I, \vec{IV}, R)$. Then, the relaxed term enumeration in Algorithm 1 is called and acquired $\tilde{J}^1, \ldots, \tilde{J}^d$ as $\tilde{J}^t \leftarrow \texttt{RTermEnum}(I, \vec{IV}, R, t)$ for $t = 1, \ldots, d$. According to [WHT+18], the sets have relationship $\tilde{J}^1 \supseteq \cdots \supseteq \tilde{J}^d$. But, in fact, there might be key bit x_j's only appearing in non-linear part of the superpoly $p_{\vec{IV}}(\vec{x})$. Such a case can be seen in the practical example on 576-round Kreyvium in [WHT+17]: for $I = \{0, 10, 20, 30, 40, 50, 60, 70\}$[8] and $\vec{IV} = (\texttt{0x613fa9ca}, \texttt{0x5068e953}, \texttt{0xe0f73db6}, \texttt{0xc8c3491f})$, the corresponding superpoly is

$$p_{\vec{IV}}(\vec{x}) = x_{47} + x_{72}x_{73} + x_{74}. \tag{22}$$

Using the relaxed term enumeration method in [WHT+18], we have $\tilde{J}^1 = \{47, 72, 73, 74\} \supseteq \tilde{J}^2 = \{72, 73\}$. But in fact, x_{72} and x_{73} only appear in the nonlinear part of $p_{\vec{IV}}$ in (22) and should be excluded from $\tilde{J}^1$. In order to reflect such a property of $p_{\vec{IV}}$, we borrow the idea in Section 8.1: if x_j does not appear in the linear part of $p_{\vec{IV}}$, the degree evaluation should have $0 = \texttt{DegEval}(I, \vec{IV}, R, \Lambda_j)$ where $\Lambda_j = [0, n)\backslash\{j\}$ resembling Section 8.1. Therefore, we can do the following steps to diminish $\tilde{J}^1$:

1. For $j \in \tilde{J}^2$, evaluate $d_j \leftarrow \texttt{DegEval}(I, \vec{IV}, R, \Lambda_j)$ where $\Lambda_j = [0, n)\backslash\{j\}$,
2. If $d_j = 0$, update $\tilde{J}^1$ by excluding j: $\tilde{J}^1 \leftarrow \tilde{J}^1\backslash\{j\}$.

We apply this method to Kreyvium and Acorn, and improve the results in [WHT+18] by 1 and 13 rounds respectively. The detailed parameters are listed in Table 6. As can be seen, both attacks have $d = 2$ and the complexities are computed according to (8) where J is the set including all involved key bits deduced with Algorithm 1 as $J \leftarrow \texttt{TermEnum}(I, \vec{IV}, R, 1)$. Apparently, smaller $\tilde{J}^1$ can bring down the complexity *Comp*. Note that, for Acorn, there is the relationship $J = \tilde{J}^1 = \tilde{J}^2$, indicating that all involved key bits appear both in linear and non-linear parts of the superpoly. So the diffusion of Acorn round function is quite smooth. The whole process needs to manipulate the flag values of both non-cube IV bits and key bits for a precise description of $\tilde{J}^1$.

Table 6: The detailed parameters for our improved attacks on 892-round Kreyvium and 763-round Acorn.

Kreyvium: $R = 892$, $\vec{IV} = \vec{1}$, $d = 2$, Complexity $2^{121.19}$.	
I	0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 62, 63, 64, 65, 67, 68, 69, 70, 71, 74, 75, 76, 77, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 102, 103, 104, 105, 107, 108, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127 ($\lvert I\rvert = 115$)
J	2, 12, 17, 26, 27, 34, 37, 40, 47, 50, 51, 52, 59, 60, 61, 62, 63, 71, 72, 73, 74, 76, 77, 78, 84, 85, 86, 88, 89, 90, 96, 109, 121 ($\lvert J\rvert = 33$)
$\tilde{J}^1$	$J\backslash\{60, 76, 77, 84, 88, 89\}$ ($\lvert\tilde{J}^1\rvert = 27$)
$\tilde{J}^2$	59, 60, 72, 73, 76, 77, 84, 85, 88, 89 ($\lvert\tilde{J}^2\rvert = 10$)

Acorn: $R = 763$, $\vec{IV} = \vec{1}$, $d = 2$, Complexity $2^{125.54}$.	
I	0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 83, 84, 85, 86, 87, 89, 90, 91, 92, 93, 94, 95, 96, 97, 99, 100, 102, 103, 104, 105, 106, 107, 108, 109, 110, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 125, 126, 127 ($\lvert I\rvert = 116$)
J	0, 1, 2, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 34, 35, 39, 44, 46, 49, 50, 54, 56, 59, 60, 83, 93 ($\lvert J\rvert = 38$)
$\tilde{J}^1$	J ($\lvert\tilde{J}^1\rvert = 38$)
$\tilde{J}^2$	J ($\lvert\tilde{J}^2\rvert = 38$)

On the Accuracy of the Division Property. For Kreyvium, we choose R_m as large as 312. We find that there are multiple trails in $R_m \rightarrow R$ and all of them can add key bits to the superpoly. Such a situation is also true for Acorn: there are multiple division trails making non-constant superpolies for arbitrary R_m's and the practical ANF analysis (can only be carried out within $R_m < 256$) also supports the non-constant property of the superpoly. These supportive verifications are showing not only the feasibility of our attacks but the better linear diffusions of such stream ciphers as well.

[8] The index in [WHT+17] starts from 1 while ours from 0.

9 Conclusion and Future Works

In this paper, we draw links between the division property and the bias phenomenon, the non-random property used in dynamic cube attack method in [DGP+11]. Based on such a theoretic finding, we are able to give the 1st dynamic cube attack on full Grain-128 with a theoretically evaluated success probability 99.83%. There are also other applications namely: drawing secure bounds for Grain-like stream ciphers against bias cube testers, and improved cube attack results on Kreyvium and Acorn.

As to future works, although most of the existing cube attack methods have been described by the division property bringing better results and steadier theoretic foundations, there are still cube attack variants, such as the correlation cube attack in [LYWL18], not included in the division property based models. Besides, as is pointed out in Section 7.5, a compromised dynamic cube attack that combines the advantages of both our attack and Dinur et al.'s is also a promising direction that deserves pursuing.

Acknowledgement. The authors thank the anonymous reviewers and the shepherd, Dr. Zhenzhen Bao, for careful readings and helpful comments. This work is supported National Key Research and Development Program of China (No. 2018YFA0306404), National Natural Science Foundation of China (Grant No. 61902030), the University of Luxembourg Internal Research Project (IRP) FDISC. Chaoyun Li is supported in part by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058.

References

[ADH+09] Jean-Philippe Aumasson, Itai Dinur, Luca Henzen, Willi Meier, and Adi Shamir. Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128. *SHARCS'09 Special-purpose Hardware for Attacking Cryptographic Systems*, page 147, 2009.

[ADMS09] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 1–22. Springer, 2009.

[ÅHJM11] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *IJWMC*, 5(1):48–59, 2011.

[CCF+16] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 313–333. Springer, 2016.

[CJF+16] Tingting Cui, Keting Jia, Kai Fu, Shiyao Chen, and Meiqin Wang. New automatic search tool for impossible differentials and zero-correlation linear approximations, 2016. http://eprint.iacr.org/2016/689.

[DGP+11] Itai Dinur, Tim Güneysu, Christof Paar, Adi Shamir, and Ralf Zimmermann. An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 327–343. Springer, 2011.

[DLWQ17] Xiaoyang Dong, Zheng Li, Xiaoyun Wang, and Ling Qin. Cube-like attack on round-reduced initialization of ketje sr. *IACR Trans. Symmetric Cryptol.*, 2017(1):259–280, 2017.

[DMP$^+$15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 733–761. Springer, 2015.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.

[DS11] Itai Dinur and Adi Shamir. Breaking Grain-128 with dynamic cube attacks. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 167–187. Springer, 2011.

[EJT07] Håkan Englund, Thomas Johansson, and Meltem Sönmez Turan. A framework for chosen IV statistical analysis of stream ciphers. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT 2007*, volume 4859 of *LNCS*, pages 268–281. Springer, 2007.

[FKM08] Simon Fischer, Shahram Khazaei, and Willi Meier. Chosen IV statistical analysis for key recovery attacks on stream ciphers. In Serge Vaudenay, editor, *AFRICACRYPT 2008*, volume 5023 of *LNCS*, pages 236–245. Springer, 2008.

[FTIM17] Yuki Funabiki, Yosuke Todo, Takanori Isobe, and Masakatu Morii. Improved integral attack on HIGHT. In Josef Pieprzyk and Suriadi Suriadi, editors, *ACISP 2017, Part I*, volume 10342 of *LNCS*, pages 363–383. Springer, 2017.

[FV13] Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 502–517. Springer, 2013.

[FWC$^+$17] Ximing Fu, Xiaoyun Wang, Jiazhe Chen, Marc Stevens, and Xiaoyang Dong. Improved attack on full-round Grain-128. Cryptology ePrint Archive, Report 2017/412, 2017. https://eprint.iacr.org/2017/412.

[FWD$^+$18] Ximing Fu, Xiaoyun Wang, Xiaoyang Dong, Willi Meier, Yonglin Hao, and Boxin Zhao. A refinement of “a key-recovery attack on 855-round Trivium" from crypto 2018. Cryptology ePrint Archive, Report 2018/999, 2018. https://eprint.iacr.org/2018/999.

[FWDM18] Ximing Fu, Xiaoyun Wang, Xiaoyang Dong, and Willi Meier. A key-recovery attack on 855-round Trivium. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 160–184, 2018.

[GRB] Zonghao Gu, Edward Rothberg, and Robert Bixby. Gurobi optimizer. http://www.gurobi.com/.

[HJL$^+$18] Yonglin Hao, Lin Jiao, Chaoyun Li, Willi Meier, Yosuke Todo, and Qingju Wang. Observations on the dynamic cube attack of 855-round TRIVIUM from Crypto’18. Cryptology ePrint Archive, Report 2018/972, 2018. https://eprint.iacr.org/2018/972.

[HJM07] Martin Hell, Thomas Johansson, and Willi Meier. Grain: A stream cipher for constrained environments. *IJWMC*, 2:86–93, 01 2007.

[HJMM06] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *IEEE International Symposium on Information Theory*, 2006.

[HW19] Kai Hu and Meiqin Wang. Automatic search for a variant of division property using three subsets. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 412–432. Springer, 2019.

[HWX$^+$17] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional Cube Attack on Reduced-Round Keccak Sponge Function. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 259–288. Springer, 2017.

[KMN10] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of NLFSR-based cryptosystems. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 130–145. Springer, 2010.

[LBDW17] Zheng Li, Wenquan Bi, Xiaoyang Dong, and Xiaoyun Wang. Improved conditional cube attacks on keccak keyed modes with MILP method. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 99–127. Springer, 2017.

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

[Liu17] Meicheng Liu. Degree evaluation of NFSR-based cryptosystems. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 227–249. Springer, 2017.

[LLW15] Meicheng Liu, Dongdai Lin, and Wenhao Wang. Searching cubes for testing boolean functions and its application to Trivium. In *ISIT 2015*, pages 496–500. IEEE, 2015.

[LYWL18] Meicheng Liu, Jingchun Yang, Wenhao Wang, and Dongdai Lin. Correlation cube attacks: From weak-key distinguisher to key recovery. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 715–744. Springer, 2018.

[MAM16] Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On ciphers that continuously access the non-volatile key. *IACR Trans. Symmetric Cryptol.*, 2016:52–79, 2016.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *INSCRYPT 2011*, volume 7537 of *LNCS*, pages 57–76. Springer, 2011.

[Saa06] Markku-Juhani Olavi Saarinen. Chosen-iv statistical attacks on estream ciphers. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT 2006*, pages 260–266. INSTICC Press, 2006.

[SBD$^+$16] Md. Iftekhar Salam, Harry Bartlett, Ed Dawson, Josef Pieprzyk, Leonie Simpson, and Kenneth Koon-Ho Wong. Investigating cube attacks on the authenticated encryption stream cipher ACORN. In Lynn Batten and Gang Li, editors, *ATIS*, volume 651 of *CCIS*, pages 15–26. Springer, 2016.

[SHW+14a] Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, and Ling Song. Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties, 2014. http://eprint.iacr.org/2014/747.

[SHW+14b] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 158–178. Springer, 2014.

[SMB17] Santanu Sarkar, Subhamoy Maitra, and Anubhab Baksi. Observing biases in the state: Case studies with Trivium and Trivia-SC. *Des. Codes Cryptography*, 82(1-2):351–375, January 2017.

[ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT2017, Part III*, volume 10212 of *LNCS*, pages 185–215. Springer, 2017.

[SWW16] Ling Sun, Wei Wang, and Meiqin Wang. MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. IACR Cryptology ePrint Archive, Report 2016/811, 2016. http://eprint.iacr.org/2016/811.

[SWW17] Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for ARX ciphers and word-based division property. *IACR Cryptology ePrint Archive, Report 2017/860*, 2017. http://eprint.iacr.org/2017/860.

[TIHM17a] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 250–279. Springer, 2017.

[TIHM17b] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. IACR Cryptology ePrint Archive, Report 2017/306, 2017. http://eprint.iacr.org/2017/306.

[TIM+18] Yosuke Todo, Takanori Isobe, Willi Meier, Kazumaro Aoki, and Bin Zhang. Fast correlation attack revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 129–159, Cham, 2018. Springer International Publishing.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to SIMON family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 357–377. Springer, 2016.

[Tod15a] Yosuke Todo. Integral cryptanalysis on full MISTY1. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 413–432. Springer, 2015.

[Tod15b] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.

[Vie07] Michael Vielhaber. Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. IACR Cryptology ePrint Archive, Report 20107/413, 2007. http://eprint.iacr.org/2007/413.

[WGR18] Qingju Wang, Lorenzo Grassi, and Christain Rechberger. Zero-sum partitions of PHOTON permutations. In Nigel Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 279–299. Springer, 2018.

[WHG+18] Senpeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. MILP method of searching integral distinguishers based on division property using three subsets. Cryptology ePrint Archive, Report 2018/1186, 2018. https://eprint.iacr.org/2018/1186.

[WHG+19] SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang, and TaiRong Shi. A practical method to recover exact superpoly in cube attack. Cryptology ePrint Archive, Report 2019/259, 2019. https://eprint.iacr.org/2019/259.

[WHT+17] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. Cryptology ePrint Archive, Report 2017/1063, 2017. https://eprint.iacr.org/2017/1063.

[WHT+18] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 275–305, 2018.

[Wu16] Hongjun Wu. ACORN v3, 2016. Submission to CAESAR competition.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 648–678. Springer, 2016.

[YT19] Chen-Dong Ye and Tian Tian. Revisit division property based cube attacks: Key-recovery or distinguishing attacks? *IACR Trans. Symmetric Cryptol.*, 2019(3):81–102, 2019.

A Specification of Grain-128

Grain-128 [HJMM06] is a NLFSR-based stream cipher. It takes as input 128 key bits $x_{[128)}$ and 96 IV bits $v_{[96)}$. Its internal state consists of an LFSR and an NFSR, both of length 128 bits. The NFSR and LFSR, denoted as $\vec{b}^0, \vec{s}^0$, are initialized by the key and IV bits respectively as follows:

$$\vec{b}^0 = (b_0, b_1, \ldots, b_{127}) = (x_0, \ldots, x_{127}), \vec{s}^0 = (s_0, s_1, \ldots, s_{127}) = (v_0, \ldots, v_{95}, 1, \ldots, 1).$$

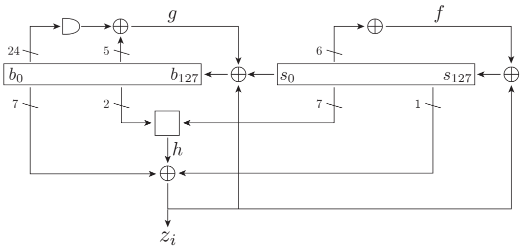

Figure 1: Structure of Grain-128

After that, for $r = 0, \ldots, 255$, the NLFSRs are updated by calling `Upd` as follows:

$$\begin{aligned}
&h^r \leftarrow b_{r+12}s_{r+8} + s_{r+13}s_{r+20} + b_{r+95}s_{r+42} + s_{r+60}s_{r+79} + b_{r+12}b_{r+95}s_{95}\\
&z^r \leftarrow h^r + s_{r+93} + \sum_{j\in A} b_{r+j} \text{ where } A = \{2, 15, 36, 45, 64, 73, 89\} && (23)\\
&g^r \leftarrow b_r + b_{r+26} + b_{r+56} + b_{r+91} + b_{r+96} + b_{r+3}b_{r+67} + b_{r+11}b_{r+13} + b_{r+17}b_{r+18}\\
&\qquad + b_{r+27}b_{r+59} + b_{r+40}b_{r+48} + b_{r+61}b_{r+65} + b_{r+68}b_{r+84}\\
&b_{r+128} \leftarrow g^r + z^r + s_r && (24)\\
&f^r \leftarrow s_r + s_{r+7} + s_{r+38} + s_{r+70} + s_{r+81} + s_{r+96}\\
&s_{r+128} \leftarrow f^r + z^r && (25)\\
&\vec{b}^r \leftarrow (b_{r+1}, \ldots b_{r+128}) && (26)\\
&\vec{s}^r \leftarrow (s_{r+1}, \ldots s_{r+128}) && (27)
\end{aligned}$$

Finally, the first keystream bit z^{256} is output, computed as (23) with parameter $r = 256$. Such a procedure can be reflected by Fig. 1.

B The Division Property Propagation of Grain-128

The division property propagation corresponding to the whole procedure of updating function can be constructed by calling `UpdDiv` Algorithm 6. It is the subroutine of Algorithm 4. The subroutines `funcZ`, `funcG`, `funcF` are defined in Algorithm 7, where `funcZ` can also describe the DP structure of the output bit, denoted as o in Algorithm 4.

Algorithm 6 MILP model for the `Upd` of Grain-128 under dynamic cube attack

```
procedure UpdDiv(the current MILP model $\mathcal{M}$, the DP structure $\vec{k}^{r-1} = (\vec{k}_b^{r-1}, \vec{k}_s^{r-1})$ where $\vec{k}_b$ and $\vec{k}_s$ are of length 128 corresponding to the DP structures of $\vec{b}^{r-1}, \vec{s}^{r-1}$ respectively, the round number $r$ ($r = 1, 2 \ldots$), the nullification strategy $N_S = \{(l_1, \alpha_{r_1}), \ldots, (l_t, \alpha_{r_t})\}$ where $\alpha_x \in \{b_x, s_x\}$, the key guess number $w \in [0, 2^\kappa)$.)
    $(\mathcal{M}, \vec{k}_b', \vec{k}_s', z) \leftarrow$ funcZ($\mathcal{M}, \vec{k}_b^{r-1}, \vec{k}_s^{r-1}$)
    $(\mathcal{M}, z_g, z_f) \leftarrow$ copyf($\mathcal{M}, z$)
    $(\mathcal{M}, \vec{k}_b'', g) \leftarrow$ funcG($\mathcal{M}, \vec{k}_b'$)
    $(\mathcal{M}, \vec{k}_s'', f) \leftarrow$ funcF($\mathcal{M}, \vec{k}_s'$)
    $(\mathcal{M}, s_0^\star, s_0^{\star\star}) \leftarrow$ copyf($\mathcal{M}, \vec{k}_s''[0], 2$)
    $(\mathcal{M}, b_{r+127}) \leftarrow$ xorf($\mathcal{M}, g, s_0^\star, z_g$)
    $(\mathcal{M}, s_{r+127}) \leftarrow$ xorf($\mathcal{M}, f, s_0^{\star\star}, z_f$)
    Assign $\vec{k}_b^r \leftarrow \vec{k}_b''[1, \ldots, 127] \| b_{r+127}$ and $\vec{k}_s^r \leftarrow \vec{k}_s''[1, \ldots, 127] \| s_{r+127}$
    if $r + 127 \in \{r_1, \ldots, r_t\}$ of $N_S$ then
        if $\alpha_{r+127} = s_{r+127}$ then
            Add constraint $\mathcal{M}.con \leftarrow \vec{k}_s[127].val = 0$.
            If $w = 0$, set flag value $\vec{k}_s[127].F = 0_c$; otherwise, $\vec{k}_s[127].F = 1_c$.
        else
            Add constraint $\mathcal{M}.con \leftarrow \vec{k}_b[127].val = 0$.
            If $w = 0$, set flag value $\vec{k}_b[127].F = 0_c$; otherwise, $\vec{k}_b[127].F = 1_c$.
        end if
    end if
    Assign $\vec{k}^r = (\vec{k}_b^r, \vec{k}_s^r)$.
end procedure
```

Algorithm 7 MILP model for NLFSR and LFSR in Grain-128

```
procedure funcZ($\mathcal{M}, \vec{b}, \vec{s}$)
    $(\mathcal{M}, \vec{b}_1, \vec{s}_1, a_1) \leftarrow$ CAND($\mathcal{M}, \vec{b}, \vec{s}, \{12\}, \{8\}$)
    $(\mathcal{M}, \vec{b}_2, \vec{s}_2, a_2) \leftarrow$ CAND($\mathcal{M}, \vec{b}_1, \vec{s}_1, \phi, \{13, 20\}$)
    $(\mathcal{M}, \vec{b}_3, \vec{s}_3, a_3) \leftarrow$ CAND($\mathcal{M}, \vec{b}_2, \vec{s}_2, \{95\}, \{42\}$)
    $(\mathcal{M}, \vec{b}_4, \vec{s}_4, a_4) \leftarrow$ CAND($\mathcal{M}, \vec{b}_3, \vec{s}_3, \phi, \{60, 79\}$)
    $(\mathcal{M}, \vec{b}_5, \vec{s}_5, a_5) \leftarrow$ CAND($\mathcal{M}, \vec{b}_4, \vec{s}_4, \{12, 95\}, \{95\}$)
    $(\mathcal{M}, \vec{b}_6, \vec{s}_6, x) \leftarrow$ CXOR($\mathcal{M}, \vec{b}_5, \vec{s}_5, \{2, 15, 36, 45, 64, 73, 89\}, \{93\}$)
    $(\mathcal{M}, z) \leftarrow$ xorf($\mathcal{M}, x, a_1, \ldots, a_5$)
    return $(\mathcal{M}, \vec{b}_6, \vec{s}_6, z)$
end procedure
```

```
procedure funcF($\mathcal{M}, \vec{s}$)
    $(\mathcal{M}, \phi, \vec{s}_1, f) \leftarrow$ CXOR($\mathcal{M}, \phi, \vec{s}, \phi, \{7, 38, 70, 81, 96\}$)
    return $(\mathcal{M}, \vec{s}_1, f)$
end procedure
```

```
procedure funcG($\mathcal{M}, \vec{b}$)
    $(\mathcal{M}, \vec{b}_1, \phi, a_1) \leftarrow$ CAND($\mathcal{M}, \vec{b}, \phi, \{3, 67\}, \phi$)
    $(\mathcal{M}, \vec{b}_2, \phi, a_2) \leftarrow$ CAND($\mathcal{M}, \vec{b}_1, \phi, \{11, 13\}, \phi$)
    $(\mathcal{M}, \vec{b}_3, \phi, a_3) \leftarrow$ CAND($\mathcal{M}, \vec{b}_2, \phi, \{17, 18\}, \phi$)
    $(\mathcal{M}, \vec{b}_4, \phi, a_4) \leftarrow$ CAND($\mathcal{M}, \vec{b}_3, \phi, \{27, 59\}, \phi$)
    $(\mathcal{M}, \vec{b}_5, \phi, a_5) \leftarrow$ CAND($\mathcal{M}, \vec{b}_4, \phi, \{40, 48\}, \phi$)
    $(\mathcal{M}, \vec{b}_6, \phi, a_6) \leftarrow$ CAND($\mathcal{M}, \vec{b}_5, \phi, \{61, 65\}, \phi$)
    $(\mathcal{M}, \vec{b}_7, \phi, a_7) \leftarrow$ CAND($\mathcal{M}, \vec{b}_6, \phi, \{68, 84\}, \phi$)
    $(\mathcal{M}, \vec{b}_{11}, \phi, x) \leftarrow$ CXOR($\mathcal{M}, \vec{b}_{10}, \phi, \{0, 26, 56, 91, 96\}, \phi$)
    $(\mathcal{M}, g) \leftarrow$ xorf($\mathcal{M}, x, a_1, \ldots, a_{10}$)
    return $(\mathcal{M}, \vec{b}_{11}, g)$
end procedure
```

Algorithm 8 MILP model for COPY+XOR and COPY+AND in Grain-128

```
procedure CAND($\mathcal{M}, \vec{b}, \vec{s}, I, J$)
    $(\mathcal{M}, b'_i, x_i) \leftarrow$ copyf($\mathcal{M}, b_i$) for all $i \in I$
    $(\mathcal{M}, s'_j, y_j) \leftarrow$ copyf($\mathcal{M}, s_j$) for all $j \in J$
    for all $i \in \{0, 1, \ldots, 127\} \backslash I$ do
        $b'_i = b_i$
    end for
    for all $j \in \{0, 1, \ldots, 127\} \backslash J$ do
        $s'_i = s_i$
    end for
    $(\mathcal{M}, z) \leftarrow$ andf($\mathcal{M}, {b'}_{i,i \in I}, {s'}_{j,j \in J}$)
    return $(\mathcal{M}, \vec{b}', \vec{s}', z)$
end procedure
```

```
procedure CXOR($\mathcal{M}, \vec{b}, \vec{s}, I, J$)
    $(\mathcal{M}, b'_i, x_i) \leftarrow$ copyf($\mathcal{M}, b_i$) for all $i \in I$
    $(\mathcal{M}, s'_j, y_j) \leftarrow$ copyf($\mathcal{M}, s_j$) for all $j \in J$
    for all $i \in \{0, 1, \ldots, 127\} \backslash I$ do
        $b'_i = b_i$
    end for
    for all $j \in \{0, 1, \ldots, 127\} \backslash J$ do
        $s'_i = s_i$
    end for
    $(\mathcal{M}, z) \leftarrow$ xorf($\mathcal{M}, {b'}_{i,i \in I}, {s'}_{j,j \in J}$)
    return $(\mathcal{M}, \vec{b}', \vec{s}', z)$
end procedure
```