Observations on the Dynamic Cube Attack of 855-Round TRIVIUM from Crypto'18

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
[2] imec-COSIC, Dept. Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium
[3] FHNW, Windisch, Switzerland
[4] NTT Secure Platform Laboratories, Tokyo 180-8585, Japan
[5] SnT, University of Luxembourg, Luxembourg
haoyonglin@yeah.net, jiaolin_jl@126.com, chaoyun.li@esat.kuleuven.be
willi.meier@fhnw, qingju.wang@uni.lu, todo.yosuke@lab.ntt.co.jp

Abstract. Recently, another kind of dynamic cube attack is proposed by Fu et al.. With some key guesses and a transformation in the output bit, they claim that, when the key guesses are correct, the degree of the transformed output bit can drop so significantly that the cubes of lower dimension can not exist, making the output bit vulnerable to the zero-sum cube tester using slightly higher dimensional cubes. They applied their method to 855-round TRIVIUM. In order to verify the correctness of their result, they even proposed a practical attack on 721-round TRIVIUM claiming that the transformed output bit after 721-rounds of initialization does not contain cubes of dimensions 31 and below. However, the degree evaluation algorithm used by Fu et al. is innovative and complicated, and its complexity is not given. Their algorithm can only be implemented on huge clusters and cannot be verified by existing theoretic tools.

In this paper, we theoretically analyze the dynamic cube attack method given by Fu et al. using the division property and MILP modeling technique.

Firstly, we draw links between the division property and Fu et al.'s dynamic cube attack so that their method can be described as a theoretically well founded and computationally economic MILP-aided division-property-based cube attack. With the MILP model drawn according to the division property, we analyzed the 721-round TRIVIUM in detail and find some interesting results:

1. The degree evaluation using our MILP method is more accurate than that of Fu et al.'s. Fu et al. prove that the degree of pure z^{721} is 40 while our method gives 29. We practically proved the correctness of our method by trying thousands of random keys, random 30-dimensional cubes and random assignments to non-cube IVs finding that the summations are constantly 0.
2. For the transformed output bit $(1 + s_1^{290}) \cdot z^{721}$, we proved the same degree 31 as Fu et al. and we also find 32-dimensional cubes have zero-sum property for correct key guesses. But since the degree of pure z^{721} is only 29, the 721-round practical attack on TRIVIUM is

violating the principle of Fu et al.'s work: after the transformation in the output bit, when the key guesses are correct, the degree of the transformed output bit has not dropped but risen.

3. Now that the degree theoretic foundation of the 721-round attack has been violated, we also find out that the key-recovery attack cannot be carried out either. We theoretically proved and practically verified that no matter the key guesses are correct or incorrect, the summation over 32-dimensional cube are always 0. So, no key bit can be recovered at all.

All these analysis on 721-round Trivium can be verified practically and we open our C++ source code for implementation as well.

Secondly, we revisit their 855-round result. Our MILP model reveal that the 855-round result suffers from the same problems with its 721-round counterpart. We provide theoretic evidence that, after their transformation, the degree of the output bit is more likely to rise rather than drop. Furthermore, since Fu et al.'s degree evaluation is written in an unclear manner and no complexity analysis is given, we rewrite the algorithm according to their main ideas and supplement a detailed complexity analysis. Our analysis indicates that a precise evaluation to the degree requires complexities far beyond practical reach. We also demonstrate that further abbreviation to our rewritten algorithm can result in wrong evaluation. This might be the reason why Fu et al. give such a degree evaluation. This is also an additional argument against Fu et al.'s dynamic cube attack method.

Thirdly, the selection of Fu et al.'s cube dimension is also questionable. According to our experiments and existing theoretic results, there is high risk that the correct key guesses and wrong ones share the same zero-sum property using Fu et al.'s cube testers. As a remedy, we suggest that concrete cubes satisfying particular conditions should be identified rather than relying on the IV-degree drop hypothesis.

To conclude, Fu et al.'s dynamic cube attack on 855-round Trivium is questionable. 855-round as well as 840-and-up-round Trivium should still be open for further convincible cryptanalysis.

Keywords: Dynamic Cube attack, Division Property, MILP, Stream Cipher, Trivium

1 Introduction

Cube attack, as well as its variants, is one of the general cryptanalytic techniques of analyzing symmetric-key cryptosystems. It can be regarded as a generalization of the chosen IV statistical attack on stream ciphers [1,2,3] or a combination of higher order differential cryptanalysis and AIDA [4]. Cube attack is based on the algebraic essence of ciphers. For a cipher with n secret variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and m public variables $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$, we can regard the algebraic normal form (ANF) of output bits as a polynomial of $\boldsymbol{x}$ and $\boldsymbol{v}$, denoted as $f(\boldsymbol{x}, \boldsymbol{v})$. For a randomly chosen set $I = \{i_1, i_2, ..., i_{|I|}\} \subset \{1, \ldots, m\}$, $f(\boldsymbol{x}, \boldsymbol{v})$

can be represented uniquely as

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

where $t_I = v_{i_1} \cdots v_{i_{|I|}}$, $p(\boldsymbol{x}, \boldsymbol{v})$ only relates to v_s's ($s \notin I$) and the secret key bits $\boldsymbol{x}$, and $q(\boldsymbol{x}, \boldsymbol{v})$ misses at least one variable in t_I. When v_s's ($s \notin I$) and $\boldsymbol{x}$ are assigned statically, the value of $p(\boldsymbol{x}, \boldsymbol{v})$ can be computed by summing the output bit $f(\boldsymbol{x}, \boldsymbol{v})$ over a structure called *cube*, denoted as C_I, consisting of $2^{|I|}$ different $\boldsymbol{v}$ vectors with $v_i, i \in I$ being active (traversing all 0-1 combinations) and non-cube indices $v_s, s \notin I$ being static constants. Due to the close link between I and C_I, the index set I is also referred as *cube* without causing ambiguity. $p(\boldsymbol{x}, \boldsymbol{v})$, also referred as the ***superpoly*** [5], may have non-random properties that can be utilized for cryptanalysis. A direct key-recovery attack can be launched when the superpoly is linear [5]. Other non-randomness such as constantness, neutrality, linearity, bias et al. can also be used for distinguishing attacks [6]. After years' development, various cube-based cryptanalysis methods have been proposed and successfully applied to all kinds of ciphers, including stream ciphers [6,7,8,9,10], hash functions [11,12,13], and authenticated encryptions [14,15].

At the beginning, the non-random properties of the superpoly can only be detected with practical experiments, which largely limits its widespread. At Crypto 2017, Todo et al. draw links among integral property, higher-order differential and cube attack, and further introduced the bit-based division property, a tool for conducting integral attacks, into the realm of cube attack [16]. The division property enable us to evaluate the non-random properties of the superpoly even if large I's of sizes far beyond practical computation are used.

Division property is a generalization of the integral property and was first proposed at EUROCRYPT 2015 [17]. With division property, the propagation of the integral characteristics can be deduced more accurately and thus result in the first theoretic key recovery attack on full MISTY1 [18]. The original division property in [17,18] can only be applied to word-oriented primitives. At FSE 2016, the bit-based division property [19] was proposed. It enables the first theoretic proof to the integral characteristics for bit-based block ciphers namely SIMON32 and Simeck32. With division property, the propagation of the integral characteristics can be represented by the operations on a set of 0-1 vectors identifying the bit positions with the zero-sum property. However, the sizes of the 0-1 vector sets are exponential to the block size of the ciphers making the deduction of the bit-based division property quite memory-consuming. Such a storage crisis has been solved by Xiang et al. [20] at ASIACRYPT 2016 by utilizing the MILP model. The operations on 0-1 vector sets are transformed to imposing division property values (0 or 1) to MILP variables, and the corresponding integral characteristics are acquired by solving the models with MILP solvers like Gurobi [21]. With this method, they are able to give integral characteristics for block ciphers with large block sizes. Xiang et al.'s method has now been applied to many other ciphers for improved integral attacks [22,23,24,25].

Todo et al.'s division property based cube attack in [16] adapt Xiang et al.'s method by taking key bits into the MILP model. With this technique, a set of key indices $J = \{j_1, j_2, \ldots, j_{|J|}\} \subset \{1, \ldots, n\}$ is deduced for the cube I s.t. its corresponding superpoly $p(\boldsymbol{x}, \boldsymbol{v})$ can only be related to the key bits x_j's ($j \in J$). With the knowledge of I and J, the ANF as well as the whole truth table of the superpoly can be recovered in the offline phase with complexity $2^{|I|+|J|}$. Then, in the online phase, the adversary only need to sum over the cube and identify the candidate secret keys by referring to the precomputed truth table. Due to division property and the power of MILP solver, cubes of larger dimension can now be used for key recoveries. By using a 72-dimensional cube, Todo et al. propose a theoretic cube attack on 832-round TRIVIUM. They also largely improve the previous best attacks on other primitives namely ACORN, Grain-128a and Kreyvium [16,26]. Based on Todo et al.'s work, Wang et al. exploite several algebraic properties of superpoly which are useful for better key-recovery cube attacks [27]. More specifically, they proposed three new techniques namely: flag technique, degree evaluation and term enumeration. With these new techniques, the results in [27] are largely improved in comparison with those in [26]. Besides the division property based cube method, it is also noticeable that Liu et al. has also invented a new key recovery scenario on TRIVIUM referred as the on TRIVIUM using a method referred as *correlation* cube attack [10]. Their method can mount to 835-round TRIVIUM using small dimensional cubes. Since our work is based on the work of [27] so we will give more detailed descriptions later.

The division property based cube attack, correlation cube attack as well as most of the traditional method are purely cultivating the property of the stream cipher. There is another kind of cube attack named the "dynamic cube attack" where the non-random property of the transformed output is utilized. The most famous dynamic cube attack results are aiming on full-round Gran-128 [28] and 855-round TRIVIUM [29]. However, the theoretic foundation of the dynamic cube attack is not so steady as that of the division property. Both [28] and [29] are acquired by pure experiments, and there are plenty of theoretically unsolved questions in them. In this paper, we only focus on [29].

The dynamic cube attack in [29] impose direct transformation to the output bit. The transformation requires correct guess to some key bits. They claim that 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 with particular dimensions. However, such a claim, similar to that of [28], has never been theoretically proved. According to [29], the degree of the (transformed) output bit is evaluated by enumerating IV-monomials using a huge cluster. But the degree evaluation algorithm is written in an unclear manner, barricading reader from realizing their algorithm and verifying its correctness. Furthermore, the number of IV-monomials enumerated is also omitted in [29] so the complexity of its degree evaluation is also unclear. In order to verify the correctness of their method, the authors of [29] propose a practical attack on 721-round TRIVIUM.

Our Contributions. In this paper, we introduce the division property to the realm of dynamic cube attack and construct corresponding MILP model to evaluate the division property propagation for concrete dynamic cube attacks on specific stream ciphers. For the method of [29], we describe its procedure using division property and MILP model, which enable us to evaluate the degree of the (transformed) output bit in a theoretically reliable manner. We give a detailed analysis to their practical result on 721-round Trivium. It is surprising that the practical example provided by themselves are violating almost all of their claims.

1. The transformed output bit has an algebraic degree higher than the original one, violating the degree drop claim. This also reveals that their strategy for constructing the transformation is questionable.
2. The degree evaluation towards the original output bit is wrong making the degree evaluation algorithm questionable (this is also the reason why they regarded the 721-round result as a support).
3. The key recovery is a complete failure: both correct and incorrect key guesses have the same zero-sum property using their cube tester.

All of the disproofs listed above are verified practically with C++ programs and we open the source codes[6] for verifications.

Besides anatomizing the practical 721-round result, we revisit Fu et al.'s dynamic cube attack on 855-round Trivium. Our division property method indicate that no degree drop occurs. In addition to the division property based evaluations, we also provide other arguments against the 855-round result. Considering that the degree evaluation algorithm in [29] is described in a complicated and unclear manner so that the complexity of IV-monomial enumeration is unable to determined. As an additional contribution, we provide a detailed description to the general process of the IV-monomial enumeration algorithm. We provide detailed analysis and available reduction to the complexities. According to detailed analysis, the enumeration requires complexities far beyond practical reach. We also demonstrate that further abbreviation will result in error in degree evaluation. This is also an argument against the method of [29]. Such arguments along with the failure of the 721-round result indicates that the dynamic cube attack on 855-round Trivium is unlikely to be true. Since the previous best theoretically reliable key-recovery result on Trivium has mount to 839 rounds, the key recoveries on 840 and more rounds should still be open for further cryptanalysis unless other theoretic evidence are provided by the authors of [29].

In addition, according to our experiments and the existing theoretic findings in [30], the selection of cube dimension in [29] is also questionable. The correct key guess might share the same zero-sum property with the wrong ones following the cube dimension selecting strategy in [29]. We also propose our remedy of finding concrete cubes whose summations have different properties before and after particular transformations. This can be regarded as a special case of IV-degree drop.

Organizations. Section 2 provides the basic description to the division property. For the Fu et al.'s dynamic cube method, we first introduce the idea of IV-degree in Section 3. With the concept of IV-degree, the theoretic foundation of

[6] `https://github.com/peterhao89/Analyze721Trivium`

their method can be described in a formal manner in Section 4. Based on such a theoretic foundation, the MILP model derived from the propagation of the division property is constructed in Section 4. With such theoretic preparations, we give detailed analysis to [29]'s practical attacks on 721-round Trivium in Section 6. We also detail their degree evaluation algorithm and analyze their result on 855-round Trivium in Section 7. The discussion on cube dimension selection strategy is in Section 8. Finally, we conclude the paper and point out some future work in Section 9.

2 Preliminaries

2.1 Static Cube Attacks and Cube Testers

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

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

where the monomials of $q(\boldsymbol{x}, \boldsymbol{v})$ miss at least one variable from $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$. Furthermore, $p(\boldsymbol{x}, \boldsymbol{v})$, referred as the superpoly in [5], is irrelevant to $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$. The value of $p(\boldsymbol{x}, \boldsymbol{v})$ can only be affected by the secret key bits $\boldsymbol{x}$ and the assignment to the non-cube IV bits v_s $(s \notin I)$. For a secret key $\boldsymbol{x}$ and an assignment to the non-cube IVs $\boldsymbol{IV} \in \mathbb{F}_2^m$, we can define a structure called cube, denoted as $C_I(\boldsymbol{IV})$, consisting of $2^{|I|}$ 0-1 vectors as follows:

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

It has been proved by Dinur and Shamir [5] that the value of superpoly p corresponding to the key $\boldsymbol{x}$ and the non-cube IV assignment $\boldsymbol{IV}$ can be computed by summing over the cube $C_I(\boldsymbol{IV})$ as follows:

$$p(\boldsymbol{x}, \boldsymbol{IV}) = \bigoplus_{\boldsymbol{v} \in C_I(\boldsymbol{IV})} f(\boldsymbol{x}, \boldsymbol{v}). \tag{2}$$

According to [5], some secret key bits can be recovered when the superpoly $p(\boldsymbol{x}, \boldsymbol{IV})$ is linear and the cube dimension $|I|$ is small enough so that the cube summation in (2) is practically implementable. Recently, both the superpoly-linearity and cube-size limitations are now conquered. Theoretic key recoveries can be carried out using a combination of the division property and the MILP modeling technique [16,26]. These new techniques are able to identify the involved key bits and upper bound the algebraic degree of the superpoly as we detail later in subsection 2.2.

In order to launch successful key-recovery attackes, the division-property-based cube attacks only requires the superpoly to be sufficiently simple: only being related to very few key bits and having a comparatively lower algebraic degree. But even if the superpoly becomes complicated, according to [6], some non-random properties (such as zero-sum, bias, presence of neutral bits etc) are also detectable using cube summations in (2) and can be used as efficient distinguishers, referred as the "cube testers".

It is noticeable that the both original cube attack and the latest division-property-based versions are only extracting properties in pure output stream bits. Such kinds of cube attacks are sometimes referred as the "static" cube attack. It is also possible to impose some transformations to the output s.t. the successfully transformed output bits are vulnerable to particular cube testers. The success of the transformation is decided by the correct guess of some key guesses while for wrong guesses, no randomness can be detected. Such kinds of cube attacks are referred as the "dynamic cube attack". There are only 2 currently formally published applications of this method: on full Grain-128 given by Dinur et al. in [28]; on 855-round Trivium given by Fu et al. in [29]. The transformation used in the former application is by nullifying crucial state bits with dynamic IV assignments while that of the latter is just multiply the output bit with intermediate state bit itself. We focus on Fu et al.'s method in this paper.

In the remainder of this paper, we refer to the value of the superpoly corresponding to the assignment $\boldsymbol{IV}$ in Eq. (2) as $p_{\boldsymbol{IV}}(\boldsymbol{x})$ for short. We use C_I as the cube corresponding to arbitrary $\boldsymbol{IV}$ setting in Eq. (1). Since C_I is defined according to I, we may also refer I as the "cube" without causing ambiguities. The size of I, denoted as $|I|$, is also referred as the "dimension" of the cube.

2.2 Bit-Based Division Property and its MILP Representation

The (Bit-Based) Division Property At 2015, the division property, a generalization of the integral property, was proposed in [17] with which better integral characteristics for word-oriented cryptographic primitives have been detected. Later, the bit-based division property was introduced in [19] so that the propagation of integral characteristics can be described in a more precise manner. The definition of the bit-based division property is as follows:

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

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

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

When the basic bitwise operations COPY, XOR, AND are applied to the elements in $\mathbb{X}$, transformations should also be made following the propagation corresponding rules `copy`, `xor`, `and` proved in [17,19]. Since round functions of cryptographic primitives are combinations of bitwise operations, we only need to determine the division property of the chosen plaintexts, denoted by $\mathcal{D}^{1^n}_{\mathbb{K}_0}$. Then, after r-round encryption, the division property of the output ciphertexts, denoted by $\mathcal{D}^{1^n}_{\mathbb{K}_r}$, can be deduced according to the round function and the propagation rules. More specifically, when the plaintext bits at index positions $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{1, 2, \ldots, n\}$ are active (the active bits traverse all $2^{|I|}$ possible combinations while other bits are assigned to static 0/1 values), the division property of such chosen plaintexts is $\mathcal{D}^{1^n}_{\boldsymbol{k}}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Then, the propagation of the division property from $\mathcal{D}^{1^n}_{\boldsymbol{k}}$ is evaluated as

$$\{\boldsymbol{k}\} := \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r,$$

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

However, when round r gets bigger, the size of $\mathbb{K}_r$ expands exponentially towards $O(2^n)$ requiring huge memory resources. So the bit-based division property has only been applied to block ciphers with tiny block sizes, such as SIMON32 and Simeck32 [19]. This memory-crisis has been solved by Xiang et al. [20] using the MILP modeling method.

Mixed Integer Linear Programming MILP is an optimization or feasibility program whose variables are restricted to integers. A MILP model $\mathcal{M}$ consists of variables $\mathcal{M}.var$, constraints $\mathcal{M}.con$, and an objective function $\mathcal{M}.obj$. MILP models can be solved by solvers like Gurobi [21]. If there is no feasible solution at all, the solver simply returns *infeasible*. If no objective function is assigned, the MILP solver only evaluates the feasibility of the model.

The application of MILP model to cryptanalysis dates back to the year 2011 [31], and has been widely used for searching characteristics corresponding to various methods. Besides integral characteristics with division property [20], there are also differential [32,33], linear [33], impossible differential [34,35], zero-correlation linear [34] characteristics etc.

Represent the Propagation of Division Property with MILP. In order to describe the bit-based division property with an MILP model, Xiang et al. first introduced a new concept *division trail* defined as follows:

Definition 2 (Division Trail [20]). *Let us consider the propagation of the division property* $\{\boldsymbol{k}\} \overset{\text{def}}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r$*. Moreover, for any vector* $\boldsymbol{k}^*_{i+1} \in \mathbb{K}_{i+1}$*, there must exist a vector* $\boldsymbol{k}^*_i \in \mathbb{K}_i$ *such that* $\boldsymbol{k}^*_i$ *can propagate to* $\boldsymbol{k}^*_{i+1}$ *by the propagation rule of the division property. Furthermore, for*

$(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r)$ if $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for all $i \in \{0, 1, \ldots, r-1\}$, we call $(\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r)$ an r-round division trail.

Let E_k be the target r-round iterated cipher. Then, if there is a division trail $\boldsymbol{k}_0 \xrightarrow{E_k} \boldsymbol{k}_r = \boldsymbol{e}_j$ $(j = 1, ..., n)$, the summation of jth bit of the ciphertexts is unknown; otherwise, if there is no division trial s.t. $\boldsymbol{k}_0 \xrightarrow{E_k} \boldsymbol{k}_r = \boldsymbol{e}_j$, we know the ith bit of the ciphertext is balanced (the summation of the ith bit is constant 0). Therefore, we have to evaluate all possible division trails to verify whether each bit of ciphertexts is balanced or not. Xiang et al. proved that the basic propagation rules `copy`, `xor`, `and` of the division property can be translated as some variables and constraints of an MILP model. With this method, all possible division trials can be covered with an MILP model $\mathcal{M}$ and the division property of particular output bits can be acquired by analyzing the solutions of the $\mathcal{M}$. After Xiang et al.'s work, some simplifications have been made to the MILP descriptions of `copy`, `xor`, `and` in [22,16]. Recently, Wang et al. [27] improve `copy`, `xor`, `and` by introducing the flag technique and name their improved versions as `copyf`, `xorf`, `andf`. We only use `copyf`, `xorf`, `andf` in this paper and they are detailed in Section 2.3.

2.3 The Bit-Based Division Property for Cube Attack

When the number of initialization rounds is not large enough for a thorough diffusion, the superpoly $p(\boldsymbol{x}, \boldsymbol{v})$ defined in Eq. (1) may not be related to all key bits $x_1, \ldots, x_n$ corresponding to some high-dimensional cube I. Instead, there is a set of key indices $J \subseteq \{1, \ldots, n\}$ s.t. for arbitrary $\boldsymbol{v} \in \mathbb{F}_2^m$, $p(\boldsymbol{x}, \boldsymbol{v})$ can only be related to x_j's $(j \in J)$. In CRYPTO 2017, Todo et al. proposed a method for determining such a set J using the bit-based division property [16]. They further showed that, with the knowledge of such J, cube attacks can be launched to recover some information related to the secret key bits. More specifically, they proved the following Lemma 1 and Proposition 1.

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

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

When f represents the first output bit after the initialization iterations, we can identify J by checking whether there is a division trial $(\boldsymbol{e}_\lambda, \boldsymbol{k}_I) \xrightarrow{f} 1$ for

$\lambda = 1, \ldots, n$ using the MILP modeling method introduced in Sect. 2.2. If the division trial $(\boldsymbol{e}_\lambda, \boldsymbol{k}_I) \xrightarrow{f} 1$ exists, we have $\lambda \in J$; otherwise, $\lambda \notin J$.

When J is determined, we know that for some proper assignment to the non-cube $\boldsymbol{IV} \in \mathbb{F}_2^m$, the corresponding superpoly $p_{\boldsymbol{IV}}(\boldsymbol{x})$ is a polynomial of $x_j, j \in J$ rather than constant 0. In order to find such a proper $\boldsymbol{IV}$, Wang et al. [27] introduce the "flag technique" so that the different models are constructed for different $\boldsymbol{IV}$ and the proper one can be determined by solving the corresponding MILP model.

Flag Technique. In the flag technique of [27], the intermediate state bit s correspond to 2 parameters $s.val$ and $s.F$:

- $s.val \in \mathcal{M}.var$ is the bit-based division property value described as a binary variable of the MILP model.
- $s.F \in \{0_c, 1_c, \delta\}$ 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).

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 natually $1_c = 1_c$, $0_c = 0_c$ and $\delta = \delta$. The $\oplus$ operation follows the rules:

$$\begin{cases} 1_c \oplus 1_c = 0_c \\ 0_c \oplus x = x \oplus 0_c = x \ \text{ for arbitrary } x \in \{1_c, 0_c, \delta\} \\ \delta \oplus x = x \oplus \delta = \delta \end{cases} \tag{3}$$

The $\times$ operation follows the rules:

$$\begin{cases} 1_c \times x = x \times 1_c = x \\ 0_c \times x = x \times 0_c = 0_c \ \text{ for arbitrary } x \in \{1_c, 0_c, \delta\} \\ \delta \times \delta = \delta \end{cases} \tag{4}$$

Considering the effect of flag, the MILP model description to the division property propagation rules corresponding to COPY, XOR and AND operations have become `copyf`, `xorf`, `andf` defined as follows:

Proposition 2 (MILP Model for COPY with Flag [27]). *Let $a \xrightarrow{COPY} (b_1, b_2, \ldots,$
$b_m)$ be a division trail of COPY. The following inequalities are sufficient to describe the propagation of the division property for* `copyf`*.*

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

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

Proposition 3 (MILP Model for XOR with Flag [27]). *Let* $(a_1, a_2, \ldots, a_m) \xrightarrow{XOR} b$ *be a division trail of XOR. The following inequalities are sufficient to describe the propagation of the division property for* `xorf`.

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

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

Proposition 4 (MILP Model for AND with Flag [27]). *Let* $(a_1, a_2, \ldots, a_m) \xrightarrow{AND} b$ *be a division trail of AND. The following inequalities are sufficient to describe the propagation of the division property for* `andf`.

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

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

With `copyf`, `xorf`, `andf`, the propagations of the division property corresponding to any cryptographic primitives can be described in a precise manner.

Degree Evaluation. Another contribution of [27] is the degree evaluation technique using the division property and MILP modeling. For an $\boldsymbol{IV} \in \mathbb{F}_2^m$ s.t. $p_{\boldsymbol{IV}}(\boldsymbol{x}) \neq 0$, the ANF of $p_{\boldsymbol{IV}}(\boldsymbol{x})$ can be represented as

$$p_{\boldsymbol{IV}}(\boldsymbol{x}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{u}} \tag{5}$$

where $a_{\boldsymbol{u}}$ is determined by the values of the non-cube IVs. [27]'s degree evaluation technique can determine an integer d s.t. for all $\boldsymbol{u}$'s with Hamming weight satisfying $hw(\boldsymbol{u}) > d$, there is constantly $a_{\boldsymbol{u}} = 0$. Such a degree evaluation is based on the following Proposition 5, which can be regarded a generalization of Proposition 1.

Proposition 5. *[27] Let* $f(\boldsymbol{x}, \boldsymbol{v})$ *be a polynomial, where* $\boldsymbol{x}$ *and* $\boldsymbol{v}$ *denote the secret and public variables, respectively. For a set of indices* $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{1, 2, \ldots, m\}$, *let* C_I *be a set of* $2^{|I|}$ *values where the variables in* $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$ *are taking all possible combinations of values. Let* $\boldsymbol{k_I}$ *be an* m*-dimensional bit vector such that* $\boldsymbol{v}^{\boldsymbol{k_I}} = t_I = v_{i_1} v_{i_2} \cdots v_{i_{|I|}}$. *Let* $\boldsymbol{k_\Lambda}$ *be an* n*-dimensional bit vector. Assuming there is no division trail such that* $(\boldsymbol{k_\Lambda} || \boldsymbol{k_I}) \xrightarrow{f} 1$, *the monomial* $\boldsymbol{x}^{\boldsymbol{k_\Lambda}}$ *is not involved in the superpoly of the cube* C_I.

With such theoretic basis, the degree d of the superpoly $p_{\boldsymbol{IV}}(\boldsymbol{x})$, can be evaluated using Algorithm 1. Note that $\boldsymbol{IV}$ can be concrete 0-1 vectors or `NULL` indicating arbitrary $\boldsymbol{IV} \in \mathbb{F}_2^m$. Therefore, for $\boldsymbol{IV} = \texttt{NULL}$, Algorithm 1 will give the largest possible d for all $\boldsymbol{IV} \in \mathbb{F}_2^m$ ($d = \max_{\boldsymbol{IV}\in\mathbb{F}_2^m}\{\deg(p_{\boldsymbol{IV}}(\boldsymbol{x}))\}$). Such degree evaluation technique plays an important role in constructing the theoretic foundation of dynamic cube attacks.

Algorithm 1 Evaluate upper bound of algebraic degree on the superpoly [27]

```
1: procedure DegEval(Cube indices $I$, specific assignment to non-cube IVs $\boldsymbol{IV}$ or $\boldsymbol{IV} = \texttt{NULL}$)
2:     Declare an empty MILP model $\mathcal{M}$.
3:     Declare $\boldsymbol{x}$ be $n$ MILP variables of $\mathcal{M}$ corresponding to secret variables.
4:     Declare $\boldsymbol{v}$ be $m$ MILP variables of $\mathcal{M}$ corresponding to public variables.
5:     $\mathcal{M}.con \leftarrow v_i.val = 1$ and assign the flags $v_i.F = \delta$ for all $i \in I$
6:     $\mathcal{M}.con \leftarrow v_i.val = 0$ for $i \in (\{1,\dots,m\} - I)$
7:     if $\boldsymbol{IV} = \texttt{NULL}$ then
8:         Assign the flags $v_i.F = \delta$ for $i \in (\{1,\dots,m\} - I)$
9:     else
10:        Assign the flags of $v_i$, $i \in (\{1,2,\dots,m\} - I)$ as:
               $v_i.F = \begin{cases} 1_c & \text{if } \boldsymbol{IV}[i] = 1 \\ 0_c & \text{if } \boldsymbol{IV}[i] = 0 \end{cases}$
11:    end if
12:    Set the objective function $\mathcal{M}.obj \leftarrow \max\{\sum_{i=1}^n x_i\}$
13:    Update $\mathcal{M}$ according to round functions and output functions
14:    Solve MILP model $\mathcal{M}$
15:    return The solution of $\mathcal{M}$.
16: end procedure
```

2.4 Unify the Specification of Trivium

In [29], the description of Trivium is slightly different from the traditional one. In this part, we first unify the symbols of traditional Trivium description with that of [29]. We first introduce Trivium in a traditional manner.

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

$$\begin{aligned}
(s_1^0, s_2^0, \dots, s_{93}^0) &= (x_0, x_1, \dots, x_{79}, 0, \dots, 0),\\
(s_{94}^0, s_{95}^0, \dots, s_{177}^0) &= (v_0, v_1, \dots, v_{79}, 0, \dots, 0),\\
(s_{178}^0, s_{279}^0, \dots, s_{288}^0) &= (0, 0, \dots, 0, 1, 1, 1).
\end{aligned}$$

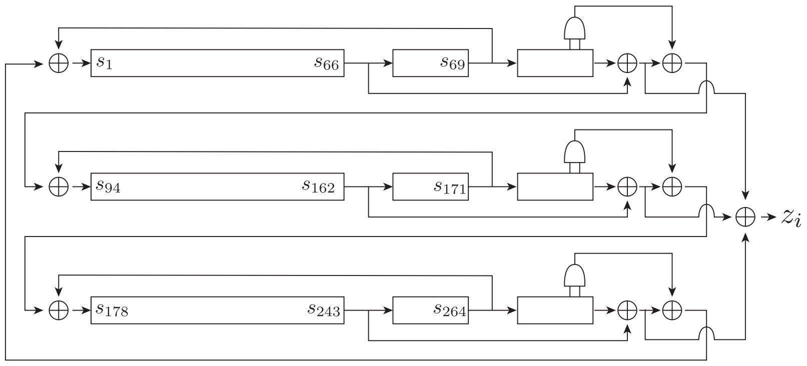

Fig. 1. Structure of Trivium

For round number $r = 1, 2, \ldots$, the state $\boldsymbol{s}^r$ is computed from $\boldsymbol{s}^{r-1}$ by calling the updating function, denoted as $\boldsymbol{s}^r = \texttt{Upd}(\boldsymbol{s}^{r-1})$ and defined as (6):

$$\begin{aligned} t_1^{r-1} &\leftarrow s_{91}^{r-1} \cdot s_{92}^{r-1} \oplus s_{66}^{r-1} \oplus s_{93}^{r-1} \oplus s_{171}^{r-1} \\ t_2^{r-1} &\leftarrow s_{175}^{r-1} \cdot s_{176}^{r-1} \oplus s_{162}^{r-1} \oplus s_{177}^{r-1} \oplus s_{264}^{r-1} \\ t_3^{r-1} &\leftarrow s_{286}^{r-1} \cdot s_{287}^{r-1} \oplus s_{243}^{r-1} \oplus s_{288}^{r-1} \oplus s_{69}^{r-1} \\ (s_1^r, s_2^r, \ldots, s_{93}^r) &\leftarrow (t_3^{r-1}, s_1^{r-1}, \ldots, s_{92}^{r-1}) \\ (s_{94}^r, s_{95}^r, \ldots, s_{177}^r) &\leftarrow (t_1^{r-1}, s_{94}^{r-1}, \ldots, s_{176}^{r-1}) \\ (s_{178}^r, s_{279}^r, \ldots, s_{288}^r) &\leftarrow (t_2^{r-1}, s_{178}^{r-1}, \ldots, s_{287}^{r-1}) \end{aligned} \tag{6}$$

After R initialization rounds, Trivium output 1 key stream bit denoted as $z^R = \texttt{Output}(\boldsymbol{s}^R)$ and computed from $\boldsymbol{s}^R$ according to the output function as

$$z^R = s_{66}^R \oplus s_{93}^R \oplus s_{162}^R \oplus s_{177}^R \oplus s_{243}^R \oplus s_{288}^R \tag{7}$$

Full version of Trivium has $R = 1152$ initialization rounds. After the initialization, one bit key stream is produced by every update function.

For [29], the s_0^r, s_1^r, s_2^r for specifying the three bits newly generated at round r, which are denoted as $s_1^r, s_{94}^r, s_{178}^r$ in (6). Therefore, the updating functions of s_0^r, s_1^r, s_2^r, as is given in [29], are as follows:

$$\begin{aligned} s_0^r &= s_2^{r-66} + s_2^{r-109} s_2^{r-110} + s_2^{r-111} + s_0^{r-69} \\ s_1^r &= s_0^{r-66} + s_0^{r-91} s_0^{r-92} + s_0^{r-93} + s_1^{r-78} \\ s_2^r &= s_1^{r-69} + s_1^{r-82} s_1^{r-83} + s_1^{r-84} + s_2^{r-87} \end{aligned} \tag{8}$$

In most parts in this paper, we unify the symbol back to the traditional manner and use $s_1^r, s_{94}^r, s_{178}^r$ to represent the s_0^r, s_1^r, s_2^r of [29] respectively. We only temporarily use the notations s_0^r, s_1^r, s_2^r of [29] in Section 7.1 for easier demonstration. It is also noticeable that the 80 key/IV bits are indexed as $0, \ldots, 79$ in accordance with [29].

3 The Concept of IV-Degree

For a cryptographic primitive with secret key bits $\boldsymbol{x} = (x_1, \ldots, x_n)$ and IV bits $\boldsymbol{v} = (v_1, \ldots, v_m)$, all its intermediately generated bits (including intermediate state bits and output bits) can be regarded as polynomials on ring $\mathbb{F}_2[\boldsymbol{x}, \boldsymbol{v}]$ whose ANF can be represented as (9)

$$f(\boldsymbol{x}, \boldsymbol{v}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^m} a_{\boldsymbol{u}} \boldsymbol{v}^{\boldsymbol{u}} \tag{9}$$

where $a_{\boldsymbol{u}} \in \mathbb{F}_2[\boldsymbol{x}] = \mathbb{F}_2[x_1, \ldots, x_n]$ is the coefficient of the IV monomial $\boldsymbol{v}^{\boldsymbol{u}}$. We define the IV-degree of $f(\boldsymbol{x}, \boldsymbol{v})$ as Definition 3.

Definition 3. ***(IV-Degree)*** *Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_m)$ be key and IV variables. Define the polynomial $f(\boldsymbol{x}, \boldsymbol{v})$ as (9). The IV-degree of f, denoted as $\deg_{IV}(f)$, is defined as the minimum integer s.t. for arbitrary $\boldsymbol{u} \in \mathbb{F}_2^m$ with hamming weight $hw(\boldsymbol{u}) > \deg_{IV}(f)$, the corresponding coefficient $a_{\boldsymbol{u}}$'s are constant 0 ($a_{\boldsymbol{u}} \equiv 0$).*

Apparently, for a monomial $a_1 a_2 \cdots a_j$, its IV-degree satisfies that

$$\deg_{IV}(a_1 a_2 \cdots a_j) \leq \sum_{t=1}^{j} \deg_{IV}(a_t) \tag{10}$$

We'll show in the following part how such a definition can grasp the idea of [29].

4 Describing the Theoretic Foundation of Fu et al.'s Method using IV-Degree

In [29], the output bit z^R is regarded as the form as follows:

$$z^R = s_i^t P_1 + P_2 \tag{11}$$

where s_i^t refers to an intermediate state bit newly generated at round t (the position i can only have 3 choices $i \in \{1, 94, 178\}$) and P_1, P_2 are other terms with complicated ANF. They find that by multiplying $(1+s_i^t)$ with z^R, the term P_1 can be eliminated since

$$(1 + s_i^t) z^R = (1 + s_i^t) P_2 \tag{12}$$

If the IV-degree of P_1 is much higher than that of $(1 + s_i^t) \cdot P_2$ and satifies

$$\deg_{IV}((1 + s_i^t) P_2) < \deg_{IV}(s_i^t P_1), \tag{13}$$

arbitrary cube I's of dimension $|I| \geq \deg_{IV}((1 + s_i^t) \cdot P_2) + 1$ can be used as zero-sum distinguishers for $(1 + s_i^t) \cdot z^R$.

The inequality (13) is the theoretic foundation of the method in [29]. Once such a degree-drop assumption fails, the key-recovery will simultaneously fail because some wrong key guesses will share the same zero-sum property as the correct ones. We will later show that once such an inequality is violated, the key-recovery attack in [29] is no longer available because some particular wrong key guesses can share the same zero-sum property as the correct ones.

Let the output z^R defined as (11) and the transformed output is defined as (12). We assume that the state bit s_i^t is of IV-degree $\deg_{IV}(s_i^t) = \lambda$ so the ANF of s_i^t can be represented in the form of (9) as:

$$s_i^t = \sum_{\boldsymbol{u}\in\mathbb{F}_2^m, hw(\boldsymbol{u})\le\lambda} g_{\boldsymbol{u}} \boldsymbol{v}^{\boldsymbol{u}} \tag{14}$$

and the secret information need to be guessed are simply a set of coefficient $g_{\boldsymbol{u}}$ satisfying $g_{\boldsymbol{u}} \in \mathbb{F}_2^n[x_1,\dots,x_n]\backslash\{0,1\}$. We define the secret information as a vector

$$\boldsymbol{G} = \left\{ g_{\boldsymbol{u}} : \text{Defined as (14)} \bigwedge g_{\boldsymbol{u}} \in \mathbb{F}_2^n[x_1,\dots,x_n]\backslash\{0,1\} \right\}. \tag{15}$$

Each element of $\boldsymbol{G}$ is simply indexed by the subindex $\boldsymbol{u}$ as $\boldsymbol{G}[\boldsymbol{u}] = g_{\boldsymbol{u}}$. In the key recovery attack, the adversary should first guess $\boldsymbol{G}$ as $\boldsymbol{G'}$. If $\boldsymbol{G'} = \boldsymbol{G}$, the state bit s_i^t can be correctly computed as (14) and the transformed output is exactly (12). For the wrong guess, this simply means that the difference $\boldsymbol{\Delta} := \boldsymbol{G} \oplus \boldsymbol{G'} \neq \boldsymbol{0}$, or in other words, the set $\mathcal{U}(\boldsymbol{G'})$ defined as (16) is not empty.

$$\mathcal{U}(\boldsymbol{G'}) = \{\boldsymbol{u} \in \mathbb{F}_2^m : \boldsymbol{G'}[\boldsymbol{u}] = \boldsymbol{G}[\boldsymbol{u}] + 1\}. \tag{16}$$

Therefore, for the wrong guess $\boldsymbol{G'}$, the state bit recovered can be expressed as

$$\hat{s}_i^t = s_i^t + \sum_{\boldsymbol{u}\in\mathcal{U}(\boldsymbol{G'})} \boldsymbol{v}^{\boldsymbol{u}} = s_i^t + \xi. \tag{17}$$

Apparently, $\deg_{IV}(\xi) \le \deg_{IV}(s_i^t) = \lambda$ and all the IV-monomial $\boldsymbol{v}^{\boldsymbol{u}}$ in ξ should also appear in the ANF of s_i^t. With the $\hat{s}_i^t$ defined in (17), the transformed output bit has become

$$(1+\hat{s}_i^t)z^R = (1+s_i^t)P_2 + \xi P_2 + \xi s_i^t P_1 \tag{18}$$

According to the definition of ξ in (17), we know that the IV-degree of ξP_2 satisfies

$$\deg_{IV}(\xi P_2) \le \deg_{IV}((1+s_i^t)P_2)$$

Therefore, comparing the wrong-key-guess derived transformed output bit in (18) with that derived from the correct key guess in (12), the correct key can only be distinguished when the IV-degrees satsify

$$\deg_{IV}((1+s_i^t)P_2) < \deg_{IV}(\xi s_i^t P_1) \le \lambda + \deg_{IV}(s_i^t P_1) \tag{19}$$

When (13) is true, it is obvious that (19) holds for arbitrary $\boldsymbol{G'} \neq \boldsymbol{G}$. If (13) is violated, we can prove in Proposition 6 that a proportion of wrong key guesses cannot be filtered.

Proposition 6. *Supposing that* (13) *is violated so that*

$$\deg_{IV}((1+s_i^t)P_2) \geq \deg_{IV}(s_i^t P_1). \tag{20}$$

Let the integer η *defined as*

$$\eta = \deg_{IV}((1+s_i^t)P_2) - \deg_{IV}(s_i^t P_1) \tag{21}$$

and assume that $\eta \geq 0$*. Then, all the wrong guess* $\boldsymbol{G}'$*'s satisfying*

$$\max\{hw(\boldsymbol{u}) : \boldsymbol{u} \in \mathcal{U}(\boldsymbol{G}')\} \leq \eta \tag{22}$$

will share the same zero-sum property with the correct key guess $\boldsymbol{G}$ *when using cubes of dimension* $\deg_{IV}((1+s_i^t)P_2)+1$*.*

Proof. According to the definition of $\mathcal{U}(\boldsymbol{G}')$ in (16), (22) simply means that the IV-degree of ξ satisfy that $\deg_{IV}(\xi) \leq \eta$. Therefore, we know that

$$\deg_{IV}(\xi s_i^t P_1) \leq \deg_I V(\xi) + \deg_{IV}(s_i^t P_1) \leq \deg_{IV}((1+s_i^t)P_2)$$

Therefore, the degree of transformed output bit corresponding to the wrong key guess $\boldsymbol{G}'$ share the same IV-degree with the correct guess $\boldsymbol{G}$. Therefore, such wrong guess $\boldsymbol{G}'$'s cannot be distinguished from the correct ones, which complete the proof. $\square$

In fact, $s_i^t P_1$ is simply part of the pure output bit z^R according to (9). Therefore, in practice, as long as we can prove $\deg_{IV}(z^R) \leq \deg_{IV}((1+s_i^t)P_2)$, we have already proved (20) and violated (13) because $\deg_{IV}(s_i^t P_1) \leq \deg_{IV}(z^R)$. We'll use this later in Section 6.

5 MILP Modeling for Fu et al.'s Dynamic Cube Method

As can be seen, the original output z^R is generated the initial state $\boldsymbol{s}^0$ by taking the following steps:

1. For $j = 1, \ldots, R$, call the updating function for new state $\boldsymbol{s}^j = \mathtt{Upd}(\boldsymbol{s}^{j-1})$;
2. Call the output function for $z^R = \mathtt{Output}(\boldsymbol{s}^R)$.

We prove that the degree of $(1+s_i^t) \cdot z^R$ ($i \in \{1, 94, 178\}$) is equal to $(1+s_i^t) \cdot \hat{z}^R$ where $\hat{z}^R$ is alternative output bit generated in a different manner defined in Proposition 7.

Proposition 7. *The evaluated output* $(1+s_i^t) \cdot z^R$ *can be regarded as the following:*

1. *For* $j = 1, \ldots, t$*, call the updating function for new state* $\boldsymbol{s}^j = \mathtt{Upd}(\boldsymbol{s}^{j-1})$*;*
2. *Store* s_i^t *in an additional register and replace* $\boldsymbol{s}^t$ *with* $\hat{\boldsymbol{s}}^t$*, where* $\hat{s}_i^t = 0$ *and* $\hat{s}_j^t = s_j^t$ *for* $j \neq i$*.*
3. *For* $j = t+1, \ldots, R$*, call the updating function for new state* $\hat{\boldsymbol{s}}^j = \mathtt{Upd}(\hat{\boldsymbol{s}}^{j-1})$*;*

4. Call the output function for $\hat{z}^R = \texttt{Output}(\hat{\boldsymbol{s}}^R)$ *and multiply it with* $(1 + s_i^t)$.

Proof. According to (11), after replacing s_i^t with constant 0, the output $\hat{z}^R$ has now become $\hat{z}^R = 0 \cdot P_1 + P_2 = P_2$. Therefore, the multiplication $(1 + s_i^t) \cdot z^R = (1 + s_i^t) \cdot \hat{z}^R = (1 + s_i^t) \cdot P_2$. □

The division property propagation corresponding to this procedure can be described with a MILP model generated from Algorithm 3. It takes as inputs initial round R and the integers (t, i) decided by s_i^t, and returns the MILP model $\mathcal{M}$ along with MILP variables $\boldsymbol{v} = (v_0, \ldots, v_{79})$ and o, corresponding to the division property values of the IV bits and output bit $(1 + s_i^t) \cdot \hat{z}^R$ respectively. So we denote the procedure of Algorithm 3 as $(\mathcal{M}, \boldsymbol{v}, o) = \texttt{TriviumModel}(R, i, t)$.

With $(\mathcal{M}, \boldsymbol{v}, o)$, we should further add constraints and objective functions to the model $\mathcal{M}$ according to different cube selections. The detailed procedure can be summarized as follows:

1. We first check whether the flag $o.F = \delta$. If $o.F = 0_c$ or 1_c, the transformed output $(1 + s_i^t) \cdot z^R$ is constant which has zero-sum property for arbitrary cubes. If $o.F = \delta$, we need to further add constraint $\mathcal{M}.con \leftarrow o.val = 1$.
2. We construct a set of IV indices $\mathcal{I}$ whose elements are available for constructing cubes. In other words, the cube I's should always satisfy $I \subseteq \mathcal{I}$. We assign the flag values $v_j.F = \delta$ for $j \in \mathcal{I}$ and update the MILP model $\mathcal{M}$ by defining the objective function:
$$\mathcal{M}.obj \leftarrow \max \sum_{j \in \mathcal{I}} v_j.val \tag{23}$$
3. For the remaining IV indices $j \in \overline{\mathcal{I}}$, update the model by $\mathcal{M}.con \leftarrow v_j.val = 0$ and assign the flag $v_j.F$'s to 0_c or 1_c according to a 0-1 vector $\boldsymbol{IV} \in \mathbb{F}_2^{80}$ decided by different attacking scenarios.

Finally, we solve the model $\mathcal{M}$ and acquire the IV-degree $\deg_{IV}((1 + s_i^t) \cdot z^R) = \mathcal{M}.obj$ defined in (23). For Trivium, the whole process can be summarized as Algorithm 4.

Algorithm 2 MILP model of division property for the core function of Trivium.

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

Algorithm 3 MILP model of division property for Trivium with transformed output $(1+s_i^t)\cdot z^R$

```
1: procedure TriviumModel(round $R$, integers $(t,i)$ corresponding to $s_i^t$ where $i \in \{1,93,178\}$)
2:     Prepare empty MILP Model $\mathcal{M}$
3:     $\mathcal{M}.var \leftarrow v_j.val$ for $j \in \{0,1,\dots,79\}$.        ▷ Declare Public Modifiable IVs
4:     $\mathcal{M}.var \leftarrow x_j.val$ for $j \in \{0,1,\dots,79\}$.        ▷ Declare Secret Keys
5:     $\mathcal{M}.var \leftarrow s_j^0.val$ for $j \in \{1,2,\dots,288\}$
6:     $s_j^0 = x_{j-1}.val$, $s_{j+93}^0 = v_{j-1}.val$ for $i = 1,\dots,80$.
7:     $\mathcal{M}.con \leftarrow s_j^0.val = 0$ for $j = 1,\dots,93,174,\dots,288$.        ▷ All non-active bits are assigned to division property value 0
8:     $s_j^0.F = \delta$ for $j = 1,\dots,80$.        ▷ Assign the flags for key bits
9:     $s_j^0.F = 0_c$ for $j = 81,\dots,285$ and $s_j^0.F = 1_c$ for $j = 286,287,288$.        ▷ Assign the flags for constant state bits
10:    for $r = 1$ to $t$ do
11:        $(\mathcal{M},\boldsymbol{x}) = $ Core$(\mathcal{M},\boldsymbol{s}^{r-1},66,171,91,92,93)$
12:        $(\mathcal{M},\boldsymbol{y}) = $ Core$(\mathcal{M},\boldsymbol{x},162,264,175,176,177)$
13:        $(\mathcal{M},\boldsymbol{z}) = $ Core$(\mathcal{M},\boldsymbol{y},243,69,286,287,288)$
14:        $\boldsymbol{s}^r = \boldsymbol{z} \ggg 1$
15:    end for
16:    Declare a new variable $\mathcal{M}.var \leftarrow \hat{s}_i^t.val$ and assign its flag value as $\hat{s}_i^t.F = 0_c$.
17:    Impose constraint as $\mathcal{M}.con \leftarrow \hat{s}_i^t.val = 0$.
18:    Store $s_i^t$ and define $\hat{\boldsymbol{s}}^t$ as
           $$\hat{s}_j^t = \begin{cases} s_j^t, & j \neq i \\ \hat{s}_i^t, & j = i \end{cases}$$
19:    for $r = t+1$ to $R$ do
20:        $(\mathcal{M},\boldsymbol{x}) = $ Core$(\mathcal{M},\hat{\boldsymbol{s}}^{r-1},66,171,91,92,93)$
21:        $(\mathcal{M},\boldsymbol{y}) = $ Core$(\mathcal{M},\boldsymbol{x},162,264,175,176,177)$
22:        $(\mathcal{M},\boldsymbol{z}) = $ Core$(\mathcal{M},\boldsymbol{y},243,69,286,287,288)$
23:        $\hat{\boldsymbol{s}}^r = \boldsymbol{z} \ggg 1$
24:    end for
25:    for all $j \in \{1,2,\dots,288\}$ w/o $66,93,162,177,243,288$ do
26:        $\mathcal{M}.con \leftarrow \hat{s}_j^R.val = 0$
27:    end for
28:    $(\mathcal{M},o') \leftarrow$ xorf$(\mathcal{M},\hat{s}_{66}^R,\hat{s}_{93}^R,\hat{s}_{162}^R,\hat{s}_{177}^R,\hat{s}_{243}^R,\hat{s}_{288}^R)$
29:    Declare a new variable $\mathcal{M}.var \leftarrow b.val$ and assign its flag value as $b.F = 1_c$.
30:    Impose constraint as $\mathcal{M}.con \leftarrow b.val = 0$.
31:    $(\mathcal{M},c) \leftarrow$ xorf$(\mathcal{M},s_i^t,b)$
32:    $(\mathcal{M},o) \leftarrow$ andf$(\mathcal{M},c,o')$
33:    return $(\mathcal{M},o,\boldsymbol{v})$
34: end procedure
```

6 Application to the 721-Round Attack in [29]

In this part, we apply our method in Section 5 to the practical dynamic cube attack on 721-round given in [29]. According to [29], all 40 IV bits with odd indices $1,3,\dots,79$ and the 3 IV bits even indices $58,64,72$ should be assigned to constant 0 and should not be selected as cube indices so we have $\mathcal{I}$ of size 37 defined as

$$\mathcal{I} = \{0,2,\dots,56,60,62,66,68,70,74,76,78\} \tag{24}$$

and $\boldsymbol{IV} = (0,0,\dots,0)$. They claimed that the IV-degree of z^{721} is $\deg_{IV}(z^{721}) = 36$ and after the transformation in the output bit, the IV-degree drops to $\deg_{IV}((1+s_{94}^{290})\cdot z^{721}) = 31$. Note that the $\mathcal{I}$ for evaluating $\deg_{IV}(z^{721})$ in [29] is slightly different from that of $\deg_{IV}((1+s_{94}^{290})\cdot z^{721})$, we will detail this later and pointing out that such a difference cannot violate our conclusion.

Algorithm 4 Evaluating the IV-degree $\deg_{IV}((1+s_i^t)\cdot z^R)$ of TRIVIUM

```
1: procedure IVDegTrivium(round $R$, integers $(t,i)$ corresponding to $s_i^t$ where $i \in \{1,94,178\}$, the
   available cube index candidates $\mathcal{I}$, the assignment to non-cube IV bits $\boldsymbol{IV} \in \mathbb{F}_2^{80}$)
2:     $(\mathcal{M}, \boldsymbol{v}, o) = \texttt{TriviumModel}(R, i, t)$.                ▷ Call Algorithm 3
3:     if $o.F = 0_c$ or $1_c$ then
4:         return $\deg_{IV}((1+s_i^t)\cdot z^R) = 0$.
5:     end if
6:     for $j \in \mathcal{I}$ do
7:         Assign the flag value of $v_j$ as $v_j.F = \delta$
8:     end for
9:     Update the model $\mathcal{M}.obj \leftarrow \sum_{j\in\mathcal{I}} v_j.val$.
10:    for $j \in \{0,\ldots,79\}\backslash\mathcal{I}$ do
11:        Assign the flag value of $v_j$ as
               $$v_j.F = \begin{cases} 0_c, & \boldsymbol{IV}[j] = 0 \\ 1_c, & \boldsymbol{IV}[j] = 1 \end{cases}$$
12:    end for
13:    Solve $\mathcal{M}$ and get the value of the objective function $\mathcal{M}.obj$.
14:    return $\deg_{IV}((1+s_i^t)\cdot z^R) = \mathcal{M}.obj$
15: end procedure
```

6.1 Compare the Preciseness of IV-Degree Evaluation

Before they go into the IV-degree evaluation to the dynamic cube attack, the authors of [29] first give an evaluation to the IV degree of pure output bit of z^{721} $\mathcal{I}$ and $\boldsymbol{IV}$. The evaluation they gave $\deg_{IV}(z^{721}) = 36$ with $\mathcal{I}$ containing all the even IV indices as:

$$\mathcal{I}' = \{0, 2, \ldots, 78\} \tag{25}$$

In order to compare the bounds drawn with their method and the with ours, we propose the IV-degree evaluation algorithm to pure z^R of R-round without the transformation in the output bit as Algorithm 5. In fact, the setting of IV bits (both division property values and flag values) in Algorithm 5 and objective function setting is identical to that of Algorithm 4. It only differs in the subroutine Algorithm 3 where the effect of s_i^t is eliminated. In fact, for evaluating $\deg_{IV}(z^{721})$ is equivalent to proving that the zero-sum property exist for all I's with dimension $|I| > \deg_{IV}(z^{721})$.

With $\mathcal{I}'$ defined as (25) and $\boldsymbol{IV} = (0,\ldots,0)$, we call Algorithm 5 that returns $\deg_{IV}(z^{721}) = 31$, lower than [29]'s 36. In order to verify the correctness of our evaluation, we randomly picked thousands of 32-dimensional cube I's using indices in $\mathcal{I}'$ (along with random keys and random assignments to $\boldsymbol{IV}[j]$ for $j \in \mathcal{I}'\backslash I$) and sum z^{721}. We find that all 32-dimensional cubes have zero-sum property when sum z^{721} over them under randomly chosen keys and randomly assigned non-cube IV bit values. This evaluation has proved that our MILP modeling method can provide sharper IV-degree upper bound than the complicated degree evaluation algorithms of [29].

Of course, in order to verify the theoretic foundation (13), we need to verify the degree drop for the same $\mathcal{I}$ and $\boldsymbol{IV}$ settings. With $\mathcal{I}$ defined as (24) and $\boldsymbol{IV} = (0,\ldots,0)$, we call Algorithm 5 that returns $\deg_{IV}(z^{721}) = 29$. We verify the correctness of our degree evaluation with sufficiently many randomly constructed

30-dimensional cube I's using indices in $\mathcal{I}$ and sum z^{721}. All the random settings have given zero-sum property proving the correctness of our degree evaluation $\deg_{IV}(z^{721}) = 29$. But for 29-dimensional cube $I \subseteq \mathcal{I}$, there exist 1-summations for some random keys. This evaluation has not only proved our sharper upper bound, but also violate the theoretic belief in (13). In other words, after the transformation in the output bit, the IV-degree has not decreased but increased. So the 721-round example is not supporting the correctness of [29] but standing against it.

Algorithm 5 Evaluating the IV-degree $\deg_{IV}((1 + s_i^t) \cdot z^R)$ TRIVIUM

```
1: procedure IVDegTriviumPure(round $R$, the available cube index candidates $\mathcal{I}$, the assignment to non-cube IV bits $\boldsymbol{IV} \in \mathbb{F}_2^{80}$)
2:     Prepare empty MILP Model $\mathcal{M}$
3:     $\mathcal{M}.var \leftarrow v_j.val$ for $j \in \{0, 1, \ldots, 79\}$.        ▷ Declare Public Modifiable IVs
4:     $\mathcal{M}.var \leftarrow x_j.val$ for $j \in \{0, 1, \ldots, 79\}$.        ▷ Declare Secret Keys
5:     for $j \in \{0, \ldots, 79\} \backslash \mathcal{I}$ do        ▷ Initialize the flags of IV bits
6:         Assign the flag value of $v_j$ as
               $$v_j.F = \begin{cases} 0_c, & \boldsymbol{IV}[j] = 0 \\ 1_c, & \boldsymbol{IV}[j] = 1 \end{cases}$$
7:     end for
8:     Update the model $\mathcal{M}.obj \leftarrow \sum_{j \in \mathcal{I}} v_j.val$.
9:     $\mathcal{M}.var \leftarrow s_j^0.val$ for $j \in \{1, 2, \ldots, 288\}$
10:    $s_j^0 = x_{j-1}$, $s_{j+93}^0 = v_{j-1}$ for $i = 1, \ldots, 80$.
11:    $\mathcal{M}.con \leftarrow s_j^0.val = 0$ for $j = 1, \ldots, 93, 174, \ldots, 288$.    ▷ All non-active bits are assigned to division property value 0
12:    $s_j^0.F = \delta$ for $j = 1, \ldots, 80$.        ▷ Assign the flags for key bits
13:    $s_j^0.F = 0_c$ for $j = 81, \ldots, 285$ and $s_j^0.F = 1_c$ for $j = 286, 287, 288$.    ▷ Assign the flags for constant state bits
14:    for $r = 1$ to $R$ do
15:        $(\mathcal{M}, \boldsymbol{x}) = \texttt{Core}(\mathcal{M}, \boldsymbol{s}^{r-1}, 66, 171, 91, 92, 93)$
16:        $(\mathcal{M}, \boldsymbol{y}) = \texttt{Core}(\mathcal{M}, \boldsymbol{x}, 162, 264, 175, 176, 177)$
17:        $(\mathcal{M}, \boldsymbol{z}) = \texttt{Core}(\mathcal{M}, \boldsymbol{y}, 243, 69, 286, 287, 288)$
18:        $\boldsymbol{s}^r = \boldsymbol{z} \ggg 1$
19:    end for
20:    for all $j \in \{1, 2, \ldots, 288\}$ w/o $66, 93, 162, 177, 243, 288$ do
21:        $\mathcal{M}.con \leftarrow s_j^R = 0$
22:    end for
23:    $(\mathcal{M}, o) \leftarrow \texttt{xorf}(\mathcal{M}, s_{66}^R, s_{93}^R, s_{162}^R, s_{177}^R, s_{243}^R, s_{288}^R)$
24:    if $o.F = 0_c$ or $1_c$ then
25:        return $\deg_{IV}((1 + s_i^t) \cdot z^R) = 0$.
26:    end if
27:    for $j \in \mathcal{I}$ do
28:        Assign the flag value of $v_j$ as $v_j.F = \delta$
29:    end for
30:    Solve $\mathcal{M}$ and get the value of the objective function $\mathcal{M}.obj$.
31:    return $\deg_{IV}((1 + s_i^t) \cdot z^R) = \mathcal{M}.obj$
32: end procedure
```

6.2 Revisiting Key-Recovery Attack on 721-Round TRIVIUM Given in [29]

By calling Algorithm 4, we are able to prove the same IV-degree evaluation $\deg_{IV}((1+s_{94}^{290})z^{721})=31$. But the degree drop assumption (13) has already been violated, the theoretic foundation to the key-recovery attack is no longer steady. We further show that the 721-round result cannot recovery any key bit because all wrong key guess $\boldsymbol{G}'$'s share the same zero-sum property with the correct $\boldsymbol{G}$ when using 32-dimensional cubes. This is a direct application to Proposition 6.

We regard z^{721} according to (9) as $z^{721}=s_{94}^{290}P_1+P_2$. As has been proved in Section 6.1, we have $\deg_{IV}(z^{721})=29$. By calling Algorithm 4, we know that $\deg_{IV}((1+s_{94}^{290})z^{721})=31$. Apparently, we have

$$\deg_{IV}(s_{94}^{290}P_1)\le\deg_{IV}(z^{721})=29\le\deg_{IV}((1+s_{94}^{290})z^{721})=\deg_{IV}((1+s_{94}^{290})P_2)=31$$

Therefore, we know that the $\eta=2$ according to its definition as (21) in Proposition 6. By analyzing the ANF of s_{94}^{290}, its IV-degree is exactly $\deg_{IV}(s_{94}^{290})=\eta=2$. Therefore, according to Proposition 6, all of the wrong key guesses and indistinguishable using 32-dimensional cubes. The key-recovery attack on 721-round TRIVIUM in [29] is a complete failure with no key bit recovered at all. We verify this with sufficiently many experiments. Their claim "***for wrong guesses, the result is 1 with probability*** $\frac{1}{2}$" is definitely delusional as can be experimentally disproved[7].

Although [29]'s method draws the correct evaluation $\deg_{IV}((1+s_{94}^{290})\cdot z^{721})=31$, we believe it's only a coincidence and that the IV-degree of pure z^{721} is sufficiently low. We'll show in Section 7.2 that when the IV-degree grows sufficiently high, a precise evaluation to the IV-degree following [29]'s idea requires impractical complexities and further abbreviations can result in serious mistakes.

7 Analyze the 855-Round TRIVIUM Result of [29]

According to the analysis in Section 6, three lessons should be learnt:

1. The strategy used in [29] for picking s_i^t is incorrect. The selected s_i^t cannot bring decrease but increase to the IV-degree of output bit.
2. The degree evaluation algorithm of [29] is questionable. The evaluation to pure output is wrong.
3. The availability of key-recovery attack is questionable. Because the IV-degree drop assumption has been violated, wrong keys guesses can have the same zero-sum property as the correct ones.

Therefore, the 855-round TRIVIUM has already become doubtful. For 855-round TRIVIUM, [29] select s_{94}^{210} for output bit transformation. The cube index candidates are $\mathcal{I}=\{0,\dots,79\}\backslash\{30,48,60,74,75\}$ so we have $|\mathcal{I}|=75$. The assignment

[7] `https://github.com/peterhao89/Analyze721Trivium`

to $\{30, 48, 60, 74, 75\}$ is also constant 0 so $\boldsymbol{IV} = (0, \ldots, 0)$. With such settings, we run Algorithm 4 only to find $\deg_{IV}((1 + s_{94}^{210}) \cdot z^{885}) = 75$. We also run Algorithm 5 and find $\deg_{IV}(z^{885}) = \deg_{IV}((1 + s_{94}^{210}) \cdot z^{885}) = 75$. Both evaluations, along with the three lessons learnt from Section 6, indicate that the 855-round attack in [29] is unlikely to be true.

In subsection 7.1, we give theoretic evidence that, for 855-round Trivium, the IV-degree of the output bit is likely to increase rather than decrease with their transformation. In subsection 7.2, we revisit Fu et al.'s degree evaluation algorithm and rewrite it in a more clear manner, showing that a precise evaluation requires complexities far beyond practical reach and further abbreviations will bring incorrect evaluation.

7.1 The IV-Degree is More Likely to Increase Rather than Decrease

We temporarily use the notations of [29] (also in Section 2.4) in this part for simpler demonstration. Since s_1^{210} is used to transform z^{855}, we can represent z^{855} according to (11) as follows:

$$z^{855} = s_1^{210} P_2 + P_3$$

Therefore, we should focus on the propagation of s_1^{210}. According to (8), s_1^{210} can only propagate to 5 intermediate state bits generated in further rounds, namely s_1^{288}, s_2^{279}, s_2^{292}, s_2^{293}, , s_2^{294}. Among the 5 bits, s_1^{210} propagate to s_1^{288}, s_2^{279}, s_2^{294} through XOR operation and s_2^{292}, s_2^{293} through AND operation. Therefore, we can denote them as $\boldsymbol{V} = (V_0, \ldots, V_4)$ defined as follows:

$$\begin{aligned} &V_0 = s_1^{279} = s_1^{210} + L_0, \quad V_1 = s_2^{288} = s_1^{210} + L_1, \quad V_2 = s_2^{294} = s_1^{210} + L_2, \\ &V_3 = s_2^{292} = a \cdot s_1^{210} + L_3, \quad V_4 = s_2^{293} = b \cdot s_1^{210} + L_4 \end{aligned} \tag{26}$$

where $a, b, L_0, \ldots, L_4$ are all functions of intermediate state bits whose ANFs can be explicitly expressed according to (8). Furthermore, with detailed study of their ANFs, we find that we find that $a, b, L_1, \ldots, L_4$ have higher IV-degrees than s_1^{210} (Lemma 2).

Lemma 2. *When IV bits at positions 30,48,60,74,75, the IV-degrees of $a, b, L_0, \ldots, L_4$ defined in* (26) *are all larger than that of s_1^{210}.*

Proof. We explicitly deduced the ANFs of $a, b, L_0, \ldots, L_4, s_1^{210}$ using Sage program [36] and evaluate their IV-degrees. We find that $\deg_{IV}(a) = 3$, $\deg_{IV}(b) = 4$, $\deg_{IV}(L_0) = 5$, $\deg_{IV}(L_1) = 6$, $\deg_{IV}(L_2) = 8$, $\deg_{IV}(L_3) = 5$, $\deg_{IV}(L_4) = 5$ and $\deg_{IV}(s_1^{210}) = 2$, which complete the proof. □

With $\boldsymbol{V}$ defined as (26), we can denote the ANF of z^{885} as follow:

$$z^{855} = \sum_{\boldsymbol{u} \in \mathbb{F}_2^5} Q_{\boldsymbol{u}} \cdot \boldsymbol{V}^{\boldsymbol{u}} \tag{27}$$

where the coefficient $Q_{\boldsymbol{u}}$'s are irrelevant to $V_0, \ldots, V_4$ and since s_1^{210} is only propagated to $V_1, \ldots, V_4$, it is equivalent to say that $Q_{\boldsymbol{u}}$'s are irrelevant to s_1^{210}.

We denote $\boldsymbol{L} = (L_0, \ldots, L_4)$ and, according to Proposition 7, after the transformation, the output becomes

$$(1 + s_1^{210}) \cdot z^{855} = (1 + s_1^{210}) \cdot \sum_{\boldsymbol{u} \in \mathbb{F}_2^5} Q_{\boldsymbol{u}} \cdot \boldsymbol{L}^{\boldsymbol{u}} \tag{28}$$

With Lemma 2 and detailed analysis to the ANFs, we find that for all $\boldsymbol{u} \in \mathbb{F}_2^5$, the IV-degree of each monomial satisfies $\deg_{IV}((1 + s_1^{210}) \cdot \boldsymbol{L}^{\boldsymbol{u}}) \geq \deg_{IV}(\boldsymbol{V}^{\boldsymbol{u}})$. Therefore, the transformation in [29] is unlikely to lower the IV-degree of z^{855} due to the well designed updating function of Trivium. On the contrary, since $Q_{\boldsymbol{u}}$'s are irrelevant to s_1^{210}, the IV-degree is more likely to increase rather than decrease after the transformation of [29].

7.2 The Precise IV-Degree of 855-Round Cannot be Practically Evaluated

In this part, we analyze the degree evaluation algorithms in [29]. We prove that the precise evaluation to the IV-degree of 855-round Trivium is much higher than practical reach and further abbreviations will result in false evaluations.

In fact, in [29], the IV-degree is evaluated by enumerating all the monomial $\boldsymbol{v}^{\boldsymbol{u}}$ appearing in the ANF of the output bit denoted as z (either $z = z^R$ or $z = (1 + s_i^t) \cdot z^R$) hereafter. When we read [29], the description of the degree evaluation process is too complicated to follow and there are a lot of important details omitted by simply using ambiguous sentences such as
"Further degree reduction for $t > 4$ ***is hard to be obtained using PC for loop executing Algorithm 3. Some man-made work should be involved to obtain further degree reduction. "***
Furthermore, the enumeration of monomials require huge memory to store the IV-monomials and a lot of time to evaluate the IV-monomials in the newly generated bits. But [29] did not provide the exact upper bound to the memory/time complexity and how many IV-monomials are computed. They simply claim that a huge cluster is used. The source code of their experiment is also unavailable making it hard to verify the correctness of their implementation.

According to the idea of [29]'s degree evaluation algorithms, we provide a clearer description to how to do the IV-monomial enumeration and give the correct IV-degree. Furthermore, the time/memory complexities of the enumeration can be evaluated in a theoretic way. According to the theoretic analysis, the enumeration cost is far beyond practical reach for 855-round Trivium. We also provide example that further abbreviation towards our described method will result in ignorance to particular high degree monomials and therefore wrongly evaluated IV-degrees.

Since the ANF of z is extremely complicated, we have to represent it as the summation of many monomials composed of intermediate state bits so that the

IV-degree evaluations can be carried out on each monomials independently. For example, when z is represented as

$$z = b_1 b_2 b_3 + c_1 c_2 c_3 c_4 \tag{29}$$

where b_i, c_j are all intermediate state bits and their IV monomials are practically computable.[8] Then, we have $\deg_{IV}(z) = \max\{\deg_{IV}(b_1 b_2 b_3), \deg_{IV}(c_1 c_2 c_3 c_4)\}$ So the evaluation of $\deg_{IV}(z)$ is equivalent to evaluating $\deg_{IV}(b_1 b_2 b_3)$ and $\deg_{IV}(c_1 c_2 c_3 c_4)$ separately. Note that for z after sufficiently many rounds of initializations, some monomials will contain more than 20 practically representable state bits (such as $z = b_1 \cdots b_{20} + c_1 \cdots c_{21} + \ldots$). But in this part, we only use (29) to demonstrate the main ideas of IV-monomial enumeration.

In [29], instead of direct evaluation to $\deg_{IV}(z)$, an targeted integer d is predefined so that the following proofs can concentrate on $\deg_{IV}(b_1 b_2 b_3) \le d$ and $\deg_{IV}(c_1 c_2 c_3 c_4) \le d$.[9] To be more specific, they use $d = 69$ for 855-round Trivium. We take the evaluation of $\deg_{IV}(b_1 b_2 b_3) \le d$ as an example. When the ANF of the monomial $b_1 b_2 b_3$ is practically computable, the IV-degree is acquired explicitly. When $\sum_{i=1}^{3} \deg_{IV}(b_i) \le d$, we know $\deg_{IV}(b_1 b_2 b_3) \le d$ according to (10). When $\sum_{i=1}^{3} \deg_{IV}(b_i) > d$, a more precise evaluation to $\deg_{IV}(b_1 b_2 b_3)$, as well as an IV-degree reduction are required. This is the critical technique but [29] simply state as *"Some man-made work should be involved to obtain further degree reduction"*. Therefore, we provide a detailed description here on how degree reduction can be carried out with as low complexity as possible.

Let the number of available IV bits is m. Denote the IV bits as $\boldsymbol{v} = (v_1, \ldots, v_m)$. The state bits b_1, b_2, b_3 are first represented as a list of 0-1 strings in $\mathbb{F}_2^m$, corresponding to the IV-monomial $\boldsymbol{v}^{\boldsymbol{u}}$'s in (9) satisfying $a_{\boldsymbol{u}} \neq 0$. We define the list of 0-1 strings corresponding to state bit $b = f(\boldsymbol{x}, \boldsymbol{v})$ as

$$\mathcal{T}(b) := \{\boldsymbol{u} \in \mathbb{F}_2^m : \text{the coefficient } a_{\boldsymbol{u}} \text{ in (9) is non-zero}\} \tag{30}$$

Apparently, the IV-degree $\deg_{IV}(b)$ is equivalent to the largest hamming weight of $\boldsymbol{u} \in \mathcal{T}(b)$:

$$\deg_{IV}(b) = \max_{\boldsymbol{u} \in \mathcal{T}(b)} \{hw(\boldsymbol{u})\} \tag{31}$$

The the multiplication of two state bits b_1 and b_2, the corresponding $\mathcal{T}(bc)$ can be computed from $\mathcal{T}(b_1)$ and $\mathcal{T}(b_2)$ as

$$\mathcal{T}(b_1 \cdot b_2) = \{\boldsymbol{u} \in \mathbb{F}_2^m : \boldsymbol{u} = \boldsymbol{u}_1 \vee \boldsymbol{u}_2 \text{ where } \boldsymbol{u}_1 \in \mathcal{T}(b_1), \boldsymbol{u}_2 \in \mathcal{T}(b_2)\} \tag{32}$$

As can be seen, if there is $\boldsymbol{u}_1, \boldsymbol{u}_1' \in \mathcal{T}(b_1)$ satisfying $\boldsymbol{u}_1 \succeq \boldsymbol{u}_1'$, then for arbitrary $\boldsymbol{u}_2 \in \mathbb{F}_2^m$, there is $(\boldsymbol{u}_1 \vee \boldsymbol{u}_2) \succeq (\boldsymbol{u}_1' \vee \boldsymbol{u}_2)$. Since $\deg_{IV}(b_1 b_2)$ only relate to the

[8] If the ANF of b_i (or c_j) is too complicated so that the IV monomials are not practically computable, they will further represent b_i (c_j) as superpoly of intermediate state bits generated in earlier stages of initialization. After such a representation, the number of intermediate state monomials of z increases but each monomial can be regarded as the multiplication of practically representable intermediate state bits.

[9] The d for two degree monomial $b_1 b_2$ here is equivalent to the $\mathcal{DEG}(b_1) + \mathcal{DEG}(b_2) - d_t(b_1 b_2)$ in [29].

elements with larger hamming weight, it is safe for us to remove $\boldsymbol{u}'_1$ and only keep $\boldsymbol{u}_1$ in $\mathcal{T}(b_1)$ so that the size of the $\mathcal{T}$ can be reduced. This is done by Fu et al. with the "Repeated-IV term Removing Algorithm" (Algorithm 4 in [29]). We can rewrite such "Repeated-IV term Removing Algorithm as Algorithm" as 6. Apparently, the complexity of Algorithm 6 is $O(|\mathcal{T}|^2)$.

Algorithm 6 Reducing the size of the list $\mathcal{T}(b)$

```
1: procedure ReduceT(The list $\mathcal{T} = \mathcal{T}(b)$ corresponding to state bit $b$ defined as (30))
2:     Reorder the elements in $\mathcal{T}$ as $\mathcal{T} = \{\boldsymbol{u}_1, \boldsymbol{u}_2 \ldots, \boldsymbol{u}_L\}$ according to hamming weight s.t. $hw(\boldsymbol{u}_1) \geq hw(\boldsymbol{u}_2) \geq \ldots \boldsymbol{u}_L$.
3:     Assign $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_L$ a flag initially assigned to value 0: $\boldsymbol{u}_i.F = 0$, $i = 1, \ldots, L$.
4:     for $i = 1, \ldots, (s-1)$ do
5:         if $\boldsymbol{u}_i.F \neq 0$ then
6:             continue.
7:         end if
8:         for $j = i+1, \ldots, L$ do
9:             If $\boldsymbol{u}_i \succeq \boldsymbol{u}_j$, assign the flag value $\boldsymbol{u}_j.F = 1$.
10:        end for
11:    end for
12:    Eliminate all $\boldsymbol{u}_j \in \mathcal{T}$ having non-zero flag values
           $\mathcal{T}' \leftarrow \mathcal{T} \backslash \{\boldsymbol{u} \in \mathcal{T} : \boldsymbol{u}.F = 1\}$
13:    return $\mathcal{T}'$.
14: end procedure
```

[29] has only considered (or at least only explicitly demonstrated) the situation when two state bits are multiplied believing the multiplication of more state bits are the same thing, which result in the mistake in degree evaluation. We find that when more bits are multiplied, we many evaluations can be ignored $\mathcal{T}'s$. For example, when computing $\mathcal{T}(b_1b_2b_3)$ and the predefined degree evaluation d satisfies that

$$d - \deg_{IV}(b_2) - \deg_{IV}(b_3) \geq 0$$

then, all elements $\boldsymbol{u} \in \mathcal{T}(b_1)$ with hamming weight lower than $d - \deg_{IV}(b_2) - \deg_{IV}(b_3)$ can be safely ignored. Therefore, for the general situation, we show in Algorithm 7 how to make the correct judgement to whether $\deg_{IV}(b_1 \cdots b_s) = d$ with the knowledge of predefined evaluation d and (size reduced)[10] lists $\mathcal{T}(b_1), \ldots, \mathcal{T}(b_s)$. Here, we require that $s \geq 2$. With the $\mathcal{T}'_s$ computed by Algorithm 7, we know that

$$\deg_{IV}(b_1 \cdots b_s) = \max\{hw(\boldsymbol{u}) : \boldsymbol{u} \in \mathcal{T}'_s\}$$

so the judgement f is correct. In fact, Algorithm 7 can not only make the correct judgement on whether $\deg_{IV}(b_1 \cdots b_s) = d$ but obtaining a proportion of $\mathcal{T}(b_1 \cdots b_s)$ containing the largest hamming weight elements as well.

However, the complexities of the term enumeration are still high even with the size reductions above. For arbitrary m, when the degree of a intermediate

[10] It means that the Repeated-IV term Removing Algorithm of [29] has been carried out on all the lists $\mathcal{T}(b_1), \ldots, \mathcal{T}(b_s)$. This only affects the efficiency rather than the accuracy of Algorithm 7.

Algorithm 7 Computation of $\mathcal{T}(b_1 b_2 \cdots b_s)$ from $\mathcal{T}(b_1), \ldots, \mathcal{T}(b_s)$

```
1: procedure CompT(predefined integer $d$ regarded as the evaluation to $\deg_{IV}(b_1 \cdots b_s)$; the (size
   reduced) lists $\mathcal{T}(b_1), \ldots, \mathcal{T}(b_s)$)
2:     Compute the IV-degrees $\deg_{IV}(b_j)$ for $j = 1, \ldots, s$ as $d_j = \max_{\boldsymbol{u} \in \mathcal{T}(b_j)}\{hw(\boldsymbol{u})\}$
3:     Initialize $\mathcal{T}'_1 \leftarrow \mathcal{T}(b_1)$.
4:     for $j = 1, \ldots, (s-1)$ do
5:         Initialize $\mathcal{T}'_{j+1} \leftarrow \phi$
6:         for $(\boldsymbol{u}, \boldsymbol{w}) \in \mathcal{T}'_j \times \mathcal{T}(b_{j+1})$ do
7:             Compute $\boldsymbol{u}' = \boldsymbol{u} \vee \boldsymbol{w}$.
8:             if $j + 2 \leq s$ and $hw(\boldsymbol{u} \vee \boldsymbol{w}) \geq \max\{0, d - \sum_{t=j+2}^{s} d_t\}$ then
9:                 Update $\mathcal{T}'_{j+1} \leftarrow \mathcal{T}'_{j+1} \bigcup \{\boldsymbol{u}'\}$
10:            end if
11:            if $j + 2 > s$ and $hw(\boldsymbol{u}') \geq d$ then
12:                Update $\mathcal{T}'_{j+1} \leftarrow \mathcal{T}'_{j+1} \bigcup \{\boldsymbol{u}'\}$
13:            end if
14:        end for
15:        Reduce the size $\mathcal{T}'_{j+1} \leftarrow$ ReduceT($\mathcal{T}'_{j+1}$) by calling Algorithm 6.
16:    end for
17:    Define a flag {, f = 1 if $\max_{\boldsymbol{u} \in \mathcal{T}'_s}\{hw(\boldsymbol{u})\} \leq d$ and f = 0 otherwise.
18:    return $(\mathcal{T}'_s, \mathrm{f})$.
19: end procedure
```

state bit b grows to $\rfloor m/2 \lfloor$, the size of $\mathcal{T}(b)$ can grow to its peak of $\binom{m}{\rfloor m/2 \lfloor}$ (all $\rfloor m/2 \lfloor$-degree monomials are involved). For the 855-round [29], we have $m = 75$ and the peak is $2^{71.55}$. When higher degree terms are generated afterwards, Algorithm 6 still requires the same time complexity to remove the redundant IV monomials. Therefore, a precise evaluation of IV-degree using IV-monomial enumeration as Fu et al. seems impractical in the first sight, and further evidence are required to verify the correctness of their degree evaluation algorithms, or "man-made work" in their own words .

Although [29] does not reveal their "man-made work" to further lowering the complexities, according to the evaluation in the 721-round attack, we believe that it involves the removal of low-degree monomials. However, once low-degree monomials are removed in earlier stage, the number of high-degree monomials generated in later rounds will decrease. For the 721-round version where the original output has not reached the highest degree, the degree evaluation may still be correct because the correct judgement can be made as long as 1 31-degree monomial is generated and there are $\binom{37}{31}$ candidates to choose from. But for 855-round version, the original output has already reached the highest degree so there is only 1 highest degree candidate to choose from. Therefore, the low-degree monomial removal will become wrong judgement to the final IV-degree. For example, we consider the situation when $d = 70$, b_1, b_2, b_3 are defined as:

$$b_1 = x_1 v_0 v_1 \cdots v_{66} v_{69} + x_2 v_1 v_2 \cdots v_{68}, \quad b_2 = x_3 v_0, \quad b_3 = x_4 v_{67} v_{68}$$

So the corresponding list $\mathcal{T}$'s are

$$\mathcal{T}(b_1) = \{(\bigoplus_{j=0}^{66} \boldsymbol{e}_j) \oplus \boldsymbol{e}_{69}, \bigoplus_{j=1}^{68} \boldsymbol{e}_j\}, \quad \mathcal{T}(b_2) = \{\boldsymbol{e}_0\}, \quad \mathcal{T}(b_3) = \{\boldsymbol{e}_{67} \oplus \boldsymbol{e}_{68}\}$$

Apparently, we have $\deg_{IV}(b_1 b_2 b_3) = 70 = d$ and $\mathcal{T}(b_1 b_2 b_3)$ contains an element of hamming weight 70: $\bigoplus_{j=0}^{69} \boldsymbol{e}_j$. But the multiplication of $b_1 b_2$ will generate a

69-degree IV-monomial $v_0 v_1 \cdots v_{68}$ and a 68-degree term $v_0 \cdots v_{66} v_{69}$. If the low-degree term is removed for saving memory complexity, the final $\mathcal{T}(b_1 b_2 b_3)$ will result in only 1 element $\bigoplus_{j=0}^{68} \boldsymbol{e}_j$ making the degree evaluation $\deg_{IV}(b_1 b_2 b_3) < d = 70$. Therefore, further size reduction to list $\mathcal{T}$'s will result in risk of wrong evaluation to the IV-degree. This might be the reason why the authors of [29] thought the key-recovery attack on 855-round were to be available.

Note that the analysis of the part has not only made the degree evaluation in [29] questionable. It is also noticeable that such degree evaluation technique has also been used in another dynamic cube attack result in [37]. Therefore, the result of [37] might be equally questionable.

8 Other Comments on Fu et al..'s Dynamic Cube Method

Besides the problems demonstrated in Section 6 and 7, the randomly picking cubes of dimension $\lambda = \deg_{IV}((1 + s_i^t) z^R) + 1$ for filtering wrong key guesses is also a questionable practice, even if (13) were satisfied. Following the notations in Section 4, for correct guess $\boldsymbol{G}$ and wrong guess $\boldsymbol{G}'$, we can rewrite the transformed output in (18) as (33) and (34) as follows:

$$(1 + s_i^t) z^R = f(\boldsymbol{x}, \boldsymbol{v}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^m} a_{\boldsymbol{u}} \boldsymbol{v}^{\boldsymbol{u}} \tag{33}$$

$$(1 + \hat{s}_i^t) z^R = \hat{f}(\boldsymbol{x}, \boldsymbol{v}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^m} \hat{a}_{\boldsymbol{u}} \boldsymbol{v}^{\boldsymbol{u}} \tag{34}$$

Even if $\deg_{IV}((1 + \hat{\hat{s}}_i^t) z^R) \geq \lambda$, it only means that for some of the $\binom{m}{\lambda}$ possible $\boldsymbol{u} \in \mathbb{F}_2^m$ of hamming weight $hw(\boldsymbol{u}) = \lambda$ satisfying $\hat{a}_{\boldsymbol{u}} \neq 0$. It is quite probable that $\hat{a}_{\boldsymbol{u}} = 0$ for most λ-hamming-weight $\boldsymbol{u}$'s in (34). In this case, randomly picked λ-dimensional cubes will have zero-sum property with high probability for both wrong and correct key guesses. Such a phenomenon has already been pointed out and studied in [30] where the proportion of λ-degree monomial $\boldsymbol{v}^{\boldsymbol{u}}$'s having non-zero coefficients are defined as the "density" of λ-dimensional cubes. The authors of [30] proved theoretically and practically that the higher-dimensional cubes have lower density than the lower-dimensional ones. Take the 721-round Trivium as an example: even if the wrong guess make the transformed output having IV-degree reaches 32, the randomly chosen 32-dimensional cube summations should make zero-summations with probability much larger than $\frac{1}{2}$.

Therefore, when (13) is satisfied and $\deg_{IV}((1 + \hat{s}_i^t) z^R)$ is only slightly higher than $\lambda = \deg_{IV}((1 + s_i^t) z^R) + 1$, the density of λ-dimensional cube in 34 is likely to be low and there is high probability that the correct key guess cannot be distinguished from the wrong ones. **The best solution is to specify sufficiently many cubes satisfying the following conditions simultaneously:**

1. **For the correct guess, the cube summation of the transformed output is constant 0.**
2. **For the wrong guess, the cube summation of the transformed output is randomly 0 or 1.**

This is equivalent to prove that there is $\boldsymbol{u} \in \mathbb{F}_2^m$ satisfying $a_{\boldsymbol{u}} = 0$ in (33) and $\hat{a}_{\boldsymbol{u}} \neq 0$ in (34). Such a phenomenon can be regarded as a algebraic degree drop in the superpoly corresponding to a concrete cube rather than a IV-degree drop: once an IV-degree drop happen, all cubes of particular dimensions can be used for zero-sum distinguishers in key-recovery attacks; for algebraic degree drop in superpoly, we need to find such "good" cubes one by one and there should be evidence showing that the algebraic degree drop do happen in the superpolies, for example, using the degree evaluation of [27] (Algorithm 1). Our remedy can only be regarded as a "*special case of IV-degree drop*" when all IV candidates are involved in a cube I.

9 Conclusion

In this paper, we describe the dynamic cube attack on Trivium given in [29] using division property and MILP model. We detail the practical example given in [29] on 721-round Trivium only to find that such a practical example is not supporting but violating the theoretic basis of their dynamic cube method. It also prove that the new complicated degree evaluation technique with "some man-made work" given in [29] is questionable. In order to simplify the degree evaluation technique in [29], we rewrite the main algorithms in a more readable manner revealing that the theoretic complexities of their method is higher than practical reach. Therefore, unless further evidence are provided, the key-recovery attack on 855-round (as well as lower-round) Trivium should still be open for further cryptanalysis. Furthermore, future works should focus on specific cube selections making correct key guesses distinguishable from wrong ones, rather than simply analyzing the IV-degrees.

Acknowledgments. We would like to thank the authors of [29], namely Xiaoyang Dong, Ximing Fu and Xiaoyun Wang for their helpful sharing. Although it's a pity that they do not agree all of our deductions and refuse to be co-authors, we are still willing to convey our sincere gratefulness to them.

References

1. Saarinen, M.O.: Chosen-iv statistical attacks on estream ciphers. In Malek, M., Fernández-Medina, E., Hernando, J., eds.: SECRYPT 2006, Proceedings of the International Conference on Security and Cryptography, Setúbal, Portugal, August 7-10, 2006, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications, INSTICC Press (2006) 260–266
2. Fischer, S., Khazaei, S., Meier, W.: Chosen IV statistical analysis for key recovery attacks on stream ciphers. In Vaudenay, S., ed.: Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings. Volume 5023 of Lecture Notes in Computer Science., Springer (2008) 236–245

3. Englund, H., Johansson, T., Turan, M.S.: A framework for chosen IV statistical analysis of stream ciphers. In Srinathan, K., Rangan, C.P., Yung, M., eds.: Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings. Volume 4859 of Lecture Notes in Computer Science., Springer (2007) 268–281
4. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. IACR Cryptology ePrint Archive, Report 20107/413 (2007) `http://eprint.iacr.org/2007/413`.
5. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In Joux, A., ed.: EUROCRYPT. Volume 5479 of LNCS., Springer (2009) 278–299
6. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In Dunkelman, O., ed.: FSE. Volume 5665 of LNCS., Springer (2009) 1–22
7. Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In Joux, A., ed.: FSE. Volume 6733 of LNCS., Springer (2011) 167–187
8. Fouque, P., Vannet, T.: Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In Moriai, S., ed.: FSE. Volume 8424 of LNCS., Springer (2013) 502–517
9. Salam, M.I., Bartlett, H., Dawson, E., Pieprzyk, J., Simpson, L., Wong, K.K.: Investigating cube attacks on the authenticated encryption stream cipher ACORN. In Batten, L., Li, G., eds.: ATIS. Volume 651 of CCIS., Springer (2016) 15–26
10. Liu, M., Yang, J., Wang, W., Lin, D.: Correlation Cube Attacks: From Weak-Key Distinguisher to Key Recovery. In Nielsen, J.B., Rijmen, V., eds.: EUROCRYPT 2018 Part II. Volume 10821 of Lecture Notes in Computer Science., Springer (2018) 715–744
11. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In Oswald, E., Fischlin, M., eds.: EUROCRYPT Part I. Volume 9056 of LNCS., Springer (2015) 733–761
12. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional Cube Attack on Reduced-Round Keccak Sponge Function. In Coron, J., Nielsen, J.B., eds.: EUROCRYPT Part II. Volume 10211 of Lecture Notes in Computer Science., Springer (2017) 259–288
13. Li, Z., Bi, W., Dong, X., Wang, X.: Improved conditional cube attacks on keccak keyed modes with milp method. In: ASIACRYPT, Springer (2017) (to appear).
14. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ASCON. IACR Trans. Symmetric Cryptol. **2017**(1) (2017) 175–202
15. Dong, X., Li, Z., Wang, X., Qin, L.: Cube-like attack on round-reduced initialization of ketje sr. IACR Trans. Symmetric Cryptol. **2017**(1) (2017) 259–280
16. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In Katz, J., Shacham, H., eds.: CRYPTO Part III. Volume 10403 of LNCS., Springer (2017) 250–279
17. Todo, Y.: Structural evaluation by generalized integral property. In Oswald, E., Fischlin, M., eds.: EUROCRYPT Part I. Volume 9056 of LNCS., Springer (2015) 287–314
18. Todo, Y.: Integral cryptanalysis on full MISTY1. In Gennaro, R., Robshaw, M., eds.: CRYPTO Part I. Volume 9215 of LNCS., Springer (2015) 413–432
19. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. In Peyrin, T., ed.: FSE. Volume 9783 of LNCS., Springer (2016) 357–377

20. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Cheon, J.H., Takagi, T., eds.: ASIACRYPT Part I. Volume 10031 of LNCS., Springer (2016) 648–678
21. Gu, Z., Rothberg, E., Bixby, R.: Gurobi optimizer. `http://www.gurobi.com/`
22. Sun, L., Wang, W., Wang, M.: 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`.
23. Sun, L., Wang, W., Wang, M.: 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`.
24. Funabiki, Y., Todo, Y., Isobe, T., Morii, M.: Improved integral attack on HIGHT. In Pieprzyk, J., Suriadi, S., eds.: ACISP 2017, Part I. Volume 10342 of LNCS., Springer (2017) 363–383
25. Wang, Q., Grassi, L., Rechberger, C.: Zero-sum partitions of PHOTON permutations. In Smart, N., ed.: CT-RSA 2018. Volume 10808 of LNCS., Springer (2018)
26. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. IACR Cryptology ePrint Archive, Report 2017/306 (2017) `http://eprint.iacr.org/2017/306`.
27. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. Cryptology ePrint Archive, Report 2017/1063 (2017) `https://eprint.iacr.org/2017/1063`.
28. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware. In Lee, D.H., Wang, X., eds.: ASIACRYPT. Volume 7073 of LNCS., Springer (2011) 327–343
29. Fu, X., Wang, X., Dong, X., Meier, W.: A Key-Recovery Attack on 855-round Trivium. In Shacham, H., Boldyreva, A., eds.: CRYPTO 2018, Part II. Volume 10992 of LNCS., Springer (2018) 160–184
30. Hao, Y.: Predicting the number of different dimensional cubes: theoretically evaluate the secure bound of cryptographic primitives against the balance testers. IET Information Security **10**(3) (2016) 142–151
31. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In Wu, C., Yung, M., Lin, D., eds.: Inscrypt. Volume 7537 of LNCS., Springer (2011) 57–76
32. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Sarkar, P., Iwata, T., eds.: ASIACRYPT Part I. Volume 8873 of LNCS., Springer (2014) 158–178
33. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L.: 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`.
34. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations (2016) `http://eprint.iacr.org/2016/689`.
35. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In Coron, J., Nielsen, J.B., eds.: EUROCRYPT Part III. Volume 10212 of LNCS., Springer (2017) 185–215

36. Developers, S.: SageMath, The Sage Mathematics Software System. (01 2016)
37. Fu, X., Wang, X., Chen, J., Stevens, M., Dong, X.: Improved attack on full-round grain-128. Cryptology ePrint Archive, Report 2017/412 (2017) `https://eprint.iacr.org/2017/412`.