Correlation Cube Attacks: From Weak-Key Distinguisher to Key Recovery

Meicheng Liu(✉), Jingchun Yang, Wenhao Wang, and Dongdai Lin

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, People's Republic of China
meicheng.liu@gmail.com

Abstract. In this paper, we describe a new variant of cube attacks called *correlation cube attack*. The new attack recovers the secret key of a cryptosystem by exploiting conditional correlation properties between the superpoly of a cube and a specific set of low-degree polynomials that we call a *basis*, which satisfies that the superpoly is a zero constant when all the polynomials in the basis are zeros. We present a detailed procedure of correlation cube attack for the general case, including how to find a basis of the superpoly of a given cube. One of the most significant advantages of this new analysis technique over other variants of cube attacks is that it converts from a weak-key distinguisher to a key recovery attack.

As an illustration, we apply the attack to round-reduced variants of the stream cipher Trivium. Based on the tool of numeric mapping introduced by Liu at CRYPTO 2017, we develop a specific technique to efficiently find a basis of the superpoly of a given cube as well as a large set of potentially good cubes used in the attack on Trivium variants, and further set up deterministic or probabilistic equations on the key bits according to the conditional correlation properties between the superpolys of the cubes and their bases. For a variant when the number of initialization rounds is reduced from 1152 to 805, we can recover about 7-bit key information on average with time complexity 2^{44}, using 2^{45} keystream bits and preprocessing time 2^{51}. For a variant of Trivium reduced to 835 rounds, we can recover about 5-bit key information on average with the same complexity. All the attacks are practical and fully verified by experiments. To the best of our knowledge, they are thus far the best known key recovery attacks for these variants of Trivium, and this is the first time that a weak-key distinguisher on Trivium stream cipher can be converted to a key recovery attack.

Keywords: Cryptanalysis · Cube attack · Numeric mapping
Stream cipher · Trivium

This work was supported by the National Natural Science Foundation of China (Grant No. 61672516), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010701), and the Fundamental Theory and Cutting Edge Technology Research Program of Institute of Information Engineering, CAS (Grant No. Y7Z0331102).

J. B. Nielsen and V. Rijmen (Eds.): EUROCRYPT 2018, LNCS 10821, pp. 715–744, 2018.
https://doi.org/10.1007/978-3-319-78375-8_23

1 Introduction

In recent years, cube attacks [11] and their variants [2,12,18] have been proven powerful in the security analysis of symmetric cryptosystems, such as Trivium [2,8,11,15], Grain-128 [9,12,16] and Keccak sponge function [3,10,18], producing the best cryptanalytic results for these primitives up to the present. Cube attacks were introduced by Dinur and Shamir at EUROCRYPT 2009 [11]. They are a generalization of chosen IV statistical attacks on stream ciphers [13,14,27], as well as an extension of higher order differential cryptanalysis [21] and AIDA [32]. The attacks treat a cryptosystem as a black-box polynomial. An attacker evaluates the sum of the output of polynomials system with a fixed private key over a subset of public variables, called a *cube*, in the hope of finding a linear coefficient of the term with maximum degree over the cube, referred to as a *superpoly*. The basic idea of cube attacks is that the symbolic sum of all the derived polynomials obtained from the black-box polynomial by assigning all the possible values to the cube variables is exactly the superpoly of the cube. The target of cube attacks is to find a number of linear superpolys on the secret variables and recover the secret information by solving a system of linear equations. In [11], the techniques was applied to a practical full key recovery on a variant of Trivium reduced to 767 rounds.

Since the seminal work of Dinur and Shamir, several variants of cube attacks have been proposed, including cube testers [2], dynamic cube attacks [12] and conditional cube attacks [18]. A cube tester [2] can detect the nonrandomness in cryptographic primitives by extracting the testable properties of the superpoly, such as unbalance, constantness and low degree, with the help of property testers. However a cube tester does not directly lead to key recovery attacks. Dynamic cube attacks [12] improve upon cube testers by introducing *dynamic variables*. When a set of conditions involving both the key bits and the dynamic variables are satisfied, the intermediate polynomials can be simplified, and cube testers (with assigned values to satisfy the conditions) are used to extract the nonrandomness of the cipher output. In this respect, a system of equations in the key bits and the dynamic variables are established. The discovery of the conditions mostly attributes to the manual work of analyzing the targeted cipher structure. Conditional cube attacks [18] work by introducing *conditional cube variables* and imposing conditions to restrain the propagation of conditional cube variables. Similar to dynamic cube attacks, the conditions used in conditional cube attacks are required to be dependent on both public bits and secret bits.

A key step to a successful cube-like attack is the search of good cubes and the corresponding superpolys during a precomputation phase. When such cubes are found, the attacker simply establishes and solves a polynomial system regarding the private key during the online phase. When cube attacks were first introduced in [11], the cryptosystems were regarded as black-box, and the authors used random walk to search for cubes experimentally. As the sum over a cube of size d involves 2^d evaluations under the fixed key, the search of cubes is time-consuming and the size of the cube is typically around 30, which restricts the capability of the attacker for better cubes. In [1] Aumasson *et al.* proposed an

evolutionary algorithm to search for good cubes. Greedy bit set algorithm was applied by Stankovski in [28] to finding cubes in distinguishers of stream ciphers. The authors of [15] and [23] both used the union of two subcubes to generate larger cube candidates. With the improved cubes of size between 34 and 37, a key recovery attack on Trivium reduced to 799 rounds [15] and a distinguisher on Trivium reduced to 839 rounds [23] are proposed.

Recently two works on cube attacks using large cubes of size greater than 50 were presented at CRYPTO 2017. Both of them treat the cryptosystems as non-blackbox polynomials. The one by Todo *et al.* [30] uses the propagation of the bit-based division property (see also [29,31]) of stream ciphers, and presents possible key recovery attacks on 832-round Trivium, 183-round `Grain`-128a and 704-round ACORN with the cubes of sizes 72, 92 and 64 respectively. The other one by Liu [22] uses numeric mapping to iteratively obtain the upper bound on the algebraic degree of an NFSR-based cryptosystem. Based on the tool of numeric mapping, cube testers are found for 842-round Trivium, 872-round Kreyvium, 1035-round TriviA-SC (v1) and 1047-round TriviA-SC (v2) with the cubes of sizes 37, 61, 63 and 61 resepectively [22].

Our Contributions. In this paper, we propose a new variant of cube attacks, named *correlation cube attack*. The general idea of this new attack is to exploit conditional correlation properties between the superpoly of a cube and a specific set of low-degree polynomials that we call a *basis*. The basis satisfies that the superpoly is a zero constant when all the polynomials in the basis are zeros. If the basis involves secret bits and has non-zero correlation with the superpoly, we can recover the secret information by solving probabilistic equations.

The attack consists of two phases: the preprocessing phase and online phase. The preprocessing phase tries to find a basis of a superpoly and its conditional correlation properties with the superpoly. The online phase targets at recovering the key by setting up and solving systems of probabilistic equations. We give a detailed procedure of both phases for the general case, including how to find a basis of the superpoly of a given cube.

As an illustration, we apply the attack to two reduced variants of the well-known stream cipher Trivium [8], and obtain the best known key recovery results for these variants. Trivium uses an 80-bit key and an 80-bit initial value (IV). We present two attacks for a variant of Trivium when the number of initialization rounds is reduced from 1152 to 805. The first attack recovers about 7 equations on the key bits by 2^4 trials on average, *i.e.*, 3-bit key information, using 2^{37}-bit operations and 2^{37}-bit data, at the expense of preprocessing time 2^{47}. In the second attack, we can recover about 14 equations on the key bits by 2^7 trials on average, *i.e.*, 7-bit key information, with time complexity 2^{44} and 2^{45} keystream bits, at the expense of preprocessing time 2^{51}. For a variant of Trivium reduced to 835 rounds, we can recover about 11 equations by 2^6 trials on average with the same complexity, that is, we can recover about 5-bit key information on average. The equations we recovered are linear or quadratic, and the quadratic ones can be easily linearized after guessing a few bits of the key.

All the attacks are directly valid for more than 30% of the keys in our experiments, and it also works for most of the other keys at the cost of recovering less key information.

Our results are summarized in Table 1 with the comparisons of the previous key recovery attacks on Trivium. In this table, by "Time" we mean the time complexity of a full key recovery. The attack time $2^{99.5}$ [24] of the full cipher is measured by bit operations, while the others are measured by cipher operations. The previous best known practical partial key recovery is applicable to a variant of Trivium reduced to 799 rounds, proposed by Fouque and Vannet [15]. The previous best known impractical (and possible) partial key recovery that is faster than an exhaustive search is applicable to a variant of Trivium reduced to 832 rounds, presented by Todo *et al.* [30]. This was shown by recovering the superpoly of a cube of size 72 with preprocessing time 2^{77}. It is possible to extract at most one key bit expression (if the superpoly depends on the key). At the same time, it is also possible that it is a distinguisher rather than a key recovery (when the superpoly does not depend on the key). In this paper, we convert from a practical weak-key distinguisher to a practical partial key recovery attack, which is applicable to a variant of Trivium reduced to 835 rounds.

Table 1. Key recovery attacks on round-reduced Trivium

#Rounds	Preproc	Data	Time	Ref
576	-	2^{12}	2^{33}	[32]
672	-	2^{15}	2^{55}	[14]
735	-	2^{29}	2^{30}	[11]
767	-	2^{34}	2^{36}	[11]
784	-	2^{39}	2^{38}	[15]
799	-	2^{40}	2^{62}	[15]
805	2^{47}	2^{37}	2^{77}	Section 4.3
805	2^{51}	2^{44}	2^{73}	Section 4.5
832	2^{77}	2^{72}	N.A	[30]
835	2^{51}	2^{44}	2^{75}	Section 4.4
Full	-	$2^{61.5}$	$2^{99.5}$	[24]
Full	-	-	2^{80}	Brute Force

The first and most critical steps in our attack are how to find good cubes and their bases. Benefited from the tool of numeric mapping [22], one can evaluate an upper bound on the algebraic degree of internal state of Trivium in linear running time. Based on this tool, we specialize the techniques to efficiently find a basis of the superpoly of a given cube as well as a large set of potentially good cubes. After this, we evaluate the conditional probability $\Pr(g = 0 | f_c(key, \cdot) \equiv 0)$ and $\Pr(g = 1 | f_c(key, \cdot) \not\equiv 0)$ for a random fixed key, where g is a function

depending on key bits in the basis of the superpoly f_c of a cube c and $f_c(key, \cdot)$ denotes the function f_c restricted at a fixed key. Finally, we record all the equations with high probability, and use them to recover the key. In the attacks, we use up to 54 cubes of sizes 28, 36 or 37. While we have found a thousand potentially favorite cubes with sizes 36 and 37 for TRIVIUM reduced to from 833 to 841 rounds, we can only make use of a small number of them in our attacks due to a limited computation resource.

Besides, we also partially apply our techniques to the stream ciphers TRIVIA-SC [5,6] and KREYVIUM [4]. We have found some cubes whose superpolys after 1047 and 852 rounds have a low-degree basis with a few elements for TRIVIA-SC and KREYVIUM respectively. The cubes for TRIVIA-SC have size larger than 60, and for KREYVIUM the size is at least 54. Though we are unable to fully verify the validity of the attack on TRIVIA-SC and KREYVIUM, we believe that there is a high chance of validness due to their similar structures with TRIVIUM.

Related Work. Similar to dynamic cube attacks and conditional cube attacks, correlation cube attacks recover the key by exploiting cube testers with constraints. Dynamic cube attacks [9,12] was applied to the full `Grain`-128 [16], while conditional cube attacks [18] was applied to round-reduced variants of KECCAK sponge function [3]. Unlike these attacks, however, the new attacks do not require the conditions to be dependent on public bits. The conditions imposed on conditional cube variables in conditional cube attacks also form a basis of the superpoly of a cube. Therefore, correlation cube attacks can be considered as a generalization of conditional cube attacks.

Actually, the idea of assigning (dynamic) constraints to public variables and using them to recover key bits was earlier appeared in conditional differential attacks, which was introduced by Knellwolf, Meier and Naya-Plasencia at ASIACRYPT 2010 [19]. The authors classified the conditions into three types:

- Type 0 conditions only involve public bits;
- Type 1 conditions involve both public bits and secret bits;
- Type 2 conditions only involve secret bits.

They exploited type 2 conditions to derive key recovery attacks based on hypothesis tests, as well as type 1 conditions to recover the key in another different way. This technique was applied to reduced variants of a few ciphers, including `Grain`-v1 [17], `Grain`-128 [16] and the block cipher family KATAN/KTANTAN [7]. Correlation cube attacks also exploit type 1 and type 2 conditions to derive key recovery attacks, while the underlying idea is very different from the work of [19]. Our techniques for finding a basis of the superpoly of a cube are more related to the automatic strategies for analyzing the conditions of higher order derivatives [20], which were exploited to derive weak-key distinguishing attacks on reduced variants of TRIVIUM. Nevertheless, the ideas are still different, and our strategies are more customized and suitable for key recovery attacks.

Organization. The rest of this paper is structured as follows. In Sect. 2, the basic definitions, notations, and background are provided. Section 3 shows the general framework of correlation cube attack, while its applications to TRIVIUM are given in Sect. 4. Section 5 concludes the paper.

2 Preliminaries

Boolean Functions and Algebraic Degree. Let $\mathbb{F}_2$ denote the binary field and $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. An n-variable Boolean function is a mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. Denote by $\mathbb{B}_n$ the set of all n-variable Boolean functions. An n-variable Boolean function f can be uniquely represented as a multivariate polynomial over $\mathbb{F}_2$,

$$f(x_1, x_2, \cdots, x_n) = \bigoplus_{c=(c_1,\cdots,c_n)\in\mathbb{F}_2^n} a_c \prod_{i=1}^{n} x_i^{c_i}, \ a_c \in \mathbb{F}_2,$$

called the algebraic normal form (ANF). The algebraic degree of f, denoted by $\deg(f)$, is defined as $\max\{wt(c) \mid a_c \neq 0\}$, where $wt(c)$ is the Hamming weight of c.

Decomposition and Basis of Boolean Functions. Given a Boolean function f, we call $f = \bigoplus_{i=1}^{u} g_i \cdot f_i$ a decomposition of f, and $G = \{g_1, g_2, \cdots, g_u\}$ a basis of f. It is clear that $g = \prod_{i=1}^{u}(g_i + 1)$ is an annihilator of f, that is, $g \cdot f = 0$.

Cube Attacks and Cube Testers. Given a Boolean function f and a term t_I containing variables from an index subset I that are multiplied together, the function can be written as the sum of terms which are supersets of I and terms that miss at least one variable from I,

$$f(x_1, x_2, \cdots, x_n) = f_S(I) \cdot t_I \oplus q(x_1, x_2, \cdots, x_n),$$

where $f_S(I)$ is called the superpoly of I in f. The basic idea of cube attacks [11] and cube testers [2] is that the symbolic sum of all the derived polynomials obtained from the function f by assigning all the possible values to the subset of variables in the term t_I is exactly $f_S(I)$. The target of cube attacks is finding a set of linear (or low-degree) functions f_S's on the secret key and recovering the key by solving this linear (or low-degree) system. Cube testers work by evaluating superpolys of carefully selected terms t_I's which are products of public variables (*e.g.*, IV bits), and trying to distinguish them from a random function. Especially, the superpoly $f_S(I)$ is equal to a zero constant, if the algebraic degree of f in the variables from I is smaller than the size of I.

Numeric Mapping. Let $f(x) = \bigoplus_{c=(c_1,\cdots,c_n)\in\mathbb{F}_2^n} a_c \prod_{i=1}^{n} x_i^{c_i}$ be a Boolean function on n variables. The *numeric mapping* [22], denoted by `DEG`, is defined as

$$\begin{aligned} \texttt{DEG}: \ & \mathbb{B}_n \times \mathbb{Z}^n \to \mathbb{Z}, \\ & (f, D) \mapsto \max_{a_c \neq 0} \{\sum_{i=1}^{n} c_i d_i\}, \end{aligned}$$

where $D = (d_1, d_2, \cdots, d_n)$ and a_c's are coefficients of the ANF of f. Let $g_i (1 \leq i \leq m)$ be Boolean functions on m variables, and denote $\deg(G) = (\deg(g_1), \deg(g_2), \cdots, \deg(g_n))$ for $G = (g_1, g_2, \cdots, g_n)$. The numeric degree of the composite function $h = f \circ G$ is defined as $\texttt{DEG}(f, \deg(G))$, denoted by $\texttt{DEG}(h)$ for short. The algebraic degree of h is always less than or equal to the numeric degree of h. The algebraic degrees of the output bits and the internal states can be estimated iteratively for NFSR-based cryptosystems by using numeric mapping [22].

3 Correlation Cube Attacks

In this section, we propose a new model for cube attacks, called *correlation cube attack*. It is a hybrid of correlation attacks [25] and cube attacks [11]. The attacked cryptosystem is supposed to be a modern symmetric-key cryptosystem. The general idea is to find a low-degree decomposition of the superpoly over a given cube, evaluate the correlation relations between the low-degree basis and the superpoly, and recover the key by solving systems of probabilistic equations. The low-degree decomposition is based on an upper bound on the algebraic degree and determines whether the superpoly is a zero constant when imposing some conditions.

The attack consists of two phases, the preprocessing phase and online phase. The preprocessing phase tries to find a basis of a superpoly and its correlation properties with the superpoly. The online phase targets at recovering the key by setting up and solving systems of probabilistic equations. In the following, we will give the details of the attack.

3.1 Preprocessing Phase

The procedure of preprocessing phase of the attack is depicted as Algorithm 1. In this phase, we can choose the input to the cipher, including the secret and public bits. First we generate a set of cubes which are potentially good in the attacks. Then for each cube c, we use a procedure `Decomposition` to find a low-degree basis of the superpoly f_c of c in the output bits of the cipher. The details of this procedure will be discussed later. If a basis of f_c is found, we calculate the conditional probability $\Pr(g = b|f_c)$ for each function g in the basis. More exactly, we compute the values of the superpoly f_c by choosing random keys and random values of free non-cube public bits, and evaluate the conditional probability $\Pr(g = 0|f_c(key, \cdot) \equiv 0)$ and $\Pr(g = 1|f_c(key, \cdot) \not\equiv 0)$ for a random key, where $f_c(key, \cdot)$ denotes the function f_c restricted at a fixed key. Finally, we record the set of (c, g, b) that satisfies $\Pr(g = b|f_c) > p$, *i.e.*,

$$\Omega = \{(c, g, b) | \Pr(g = b|f_c) > p\}.$$

Note that if g depends on both key bits and public bits, the attack will become more efficient, at least not worse, than the case that g depends only

on key bits (which can naturally be used to mount weak-key distinguishing attacks). A distinguishing attack is a much weaker attack compared to a key recovery attack, while a weak-key distinguishing attack is even weaker than a normal distinguishing attack. To illustrate how to convert from a weak-key distinguishing attack to a key recovery attack, we assume the weak case: g only depends on key bits (if not, we can set the public bits in g to constants).

Algorithm 1. Correlation Cube Attacks (Preprocessing Phase)

1: Generate a cube set C;
2: For each cube c in C do:
3: $Q_c \leftarrow$ `Decomposition`(c), and goto next c if Q_c is empty; `/* try to find a basis of the superpoly` f_c `of` c `in the output bits of the cipher */`
4: Estimate the conditional probability $\Pr(g = b|f_c)$ for each function g in the basis Q_c of the superpoly f_c, and select (c, g, b) that satisfies $\Pr(g = b|f_c) > p$.

Example 1. Given a Boolean polynomial f on five public variables $v = (v_1, v_2, v_3, v_4, v_5)$ and five secret variables $x = (x_1, x_2, x_3, x_4, x_5)$,

$$\begin{aligned} f(v, x) = {} & f_7(v_5, x)v_1v_2v_3v_4 + f_6(v_5, x)v_1v_2v_4 \\ & + f_5(v_5, x)v_2v_3v_4 + f_4(v_5, x)v_1v_4 \\ & + f_3(v_5, x)v_2v_4 + f_2(v_5, x)v_3 \\ & + f_1(v_5, x)v_4 + f_0(v_5, x) \end{aligned}$$

and

$$f_7(v_5, x) = h_1(v_5, x_2, x_3, x_4, x_5)x_1 + h_2(v_5, x_1, x_2, x_3, x_4)x_5,$$

where h_1, h_2 and $f_i (0 \leq i \leq 6)$ are arbitrary Boolean functions. We can build a weak-key cube tester for the polynomial f, by using the cube $\{v_1, v_2, v_3, v_4\}$ under the conditions $x_1 = x_5 = 0$, while it seems to be immune to cube or dynamic cube attacks. To convert from a weak-key cube tester to a key recovery, we test the correlation properties between the superpoly f_7 and its basis $\{x_1, x_5\}$. We observe the values of $f_7(v_5, x)$ for $v_5 = 0, 1$, and estimate the conditional probability

$$\Pr(x_i = 0|f_7(0, x) = f_7(1, x) = 0)$$

and

$$\Pr(x_i = 1|f_7(0, x) \neq 0 \text{ or } f_7(1, x) \neq 0)$$

for $i = 1, 5$. Noting that $(x_1 + 1)(x_5 + 1)f_7 = 0$, we also have

$$(x_1 + 1)(x_5 + 1) = 0 \text{ if } f_7(0, x) \neq 0 \text{ or } f_7(1, x) \neq 0.$$

This allows us to derive information regarding the secret key.

Now we explain how to find a basis of the superpoly f_c for a given cube c. The procedure `Decomposition` is described in Algorithm 2. The main idea is to make use of the coefficients Q_t of the terms with maximum degree on cube variables in the bits s_t of the internal state at the first rounds of the attacked cipher ($t \leq N_0$). Note that it is highly possible that f_c depends on these coefficients. To find a set Q such that $f_c = \bigoplus_{g \in Q} g \cdot f_g$, we first annihilate all the coefficients in Q_t with $1 \notin Q_t$ for all $t \leq N_0$, and then determine whether f_c is a zero constant. Once we detect that the algebraic degree of an output bit on cube variables is less than the size of c, which implies $f_c = 0$, we obtain a basis $Q = \bigcup_{t \leq N_0 | 1 \notin Q_t} Q_t$ for f_c. Then we minimize the number of elements in Q by removing redundant equations one by one. Finally, the procedure returns the minimum basis Q.

Algorithm 2. `Decomposition`

Require: a cube c of size n

1: Set Q to the empty set and X to the variable set $\{x_i | i \in c\}$;
/* find a basis Q */
2: For t from 0 to N_0 do:
3: Compute the ANF of s_t and set $d_t = \deg(s_t, X)$;
4: Set Q_t to the set of the coefficients of all the terms with degree d_t in the ANF of s_t;
5: If $d_t \geq 1$ and $1 \notin Q_t$, then set $Q = Q \cup Q_t$ and $d_t = \deg(s'_t, X)$, where s'_t is the function formed by removing all the terms with degree d_t from s_t;
6: Given $\{d_t\}$ and under the conditions that $g = 0$ for each $g \in Q$, find an upper bound $d(Q)$ on the degree of the N-round output bit;
7: If $d(Q) \geq n$, then **return** $\emptyset$;
/* minimize the basis Q */
8: Minimize N_0 such that $d(Q) < n$, and generate a new Q;
9: For each g in Q do:
10: Set $Q' = Q \setminus \{g\}$;
11: For $t \leq N_0$, if $\text{zero}(Q') \subseteq \text{zero}(Q_t)$ then set $d_t = \deg(s'_t, X)$, otherwise set $d_t = \deg(s_t, X)$, where $\text{zero}(Q)$ is the solution set of $\{g = 0 | g \in Q\}$;
12: If $d(Q') < n$, then set $Q = Q'$;
13: **return** Q.

For explanation of Algorithm 2, we give an example on a nonlinear feedback shift register (NFSR) in the following.

Example 2. Let $s_t = s_{t-6}s_{t-7} + s_{t-8}$ be the update function of an NFSR with size 8. Let $(s_0, s_1, \cdots, s_7) = (x_1, x_2, x_3, x_4, v_1, v_2, v_3, 0)$, and $X = \{v_1, v_2, v_3\}$ be the cube variables. Taking $t = 10$ for example, we compute $s_{10} = s_4 s_3 + s_2 = v_1 x_4 + x_3$, then have $d_{10} = 1$, $Q_{10} = \{x_4\}$ and $s'_{10} = x_3$. Since $1 \notin Q_{10}$, we set $Q = Q \cup Q_{10}$ and $d_{10} = 0$. After computations for $t \leq N_0 = 17$, we obtain

$$Q = Q_{10} \cup Q_{16} \cup Q_{17} = \{x_4, x_2x_4 + x_3x_4, x_3 + x_4\},$$
$$(d_0, d_1, \cdots, d_{17}) = (0, 0, 0, 0, 1, 1, 1, -\infty, 0, 0, 0, 2, 2, 1, 1, 0, 0, 1).$$

For $N = 29$, we find an upper bound $d(Q) = 2$ on the algebraic degree of s_N by applying the numeric mapping. We can check that 17 is the minimum N_0 such that $d(Q) < n = 3$. After minimizing the basis Q, we obtain $Q = \{x_4, x_3 + x_4\}$.

Actually, the ANF of s_{29} is $v_1v_2v_3(x_2x_3x_4 + x_1x_3 + x_1x_4 + x_2x_4) + v_1v_3(x_2x_3x_4 + x_1x_4) + v_3(x_2x_3x_4 + x_1x_2) + v_2$, and the coefficient of the maximum term $v_1v_2v_3$ is $f_c = x_2x_3x_4 + x_1x_3 + x_1x_4 + x_2x_4$, which will be annihilated when $x_4 = x_3 + x_4 = 0$. We can see that Q is a basis of the superpoly f_c.

Complexity. It is hard to evaluate the complexity of the step for generating good cubes. How to find favorite cubes is still an intractable problem in cube attacks. The time complexity of `Decomposition` is $T_{N_0} + n_Q \cdot T_N$, where n_Q is the size of the primary basis $\bigcup_{t \le N_0 | 1 \notin Q_t} Q_t$, T_{N_0} is the time for computing this basis (Line 2–5 in Algorithm 2), and T_N is the time complexity of finding an upper bound on the algebraic degree of the N-round output bits. The estimation of the conditional probability $\Pr(g = b|f_c)$ for a cube c of size n needs about $\alpha \cdot 2^n$ cipher encryption operations, when using α values of f_c in the estimation. The total time complexity of preprocessing phase is thus about

$$n_C(T_{N_0} + n_Q \cdot T_N + \alpha \cdot 2^n),$$

where n_C is the number of cubes in C, not taking into account the time for generating the cube set C.

3.2 Online Phase

The procedure of online phase of the attack is depicted in Algorithm 3. In the online phase, the key is unknown, and we can only control the public bits. We first derive two sets of probabilistic equations according to the ciphertexts (or keystream bits), and then repeatedly solve a system consisting of a part of these equations until the correct key is found. In preprocessing phase, we have obtained a set Ω of (c, g, b) that satisfies $\Pr(g = b|f_c) > p$. In online phase, for each cube c, we test whether its superpoly f_c is a zero constant by computing α values of f_c over the cube with different non-cube public bits. If f_c is not a zero constant, we derive new equations $g = 1$ with $(c, g, 1) \in \Omega$; otherwise, we record the equations $g = 0$ with $(c, g, 0) \in \Omega$. After all the cubes are handled, we derive two sets of equations, $G_0 = \{g = 0|(c, g, 0) \in \Omega, f_c = 0\}$ and $G_1 = \{g = 1|(c, g, 1) \in \Omega, f_c \neq 0\}$. For the case that $\{g|g = 0 \in G_0 \text{ and } g = 1 \in G_1\}$ is not empty, we can use the one with higher probability between $g = 0$ and $g = 1$ or neither of them. We then randomly choose r_0 equations from G_0 and r_1 equations from G_1, solve these $r_0 + r_1$ equations and check whether the solutions are correct. Repeat this step until the correct key is found.

Complexity. The loop for deriving the equation sets G_0 and G_1 requires at most $n_C\alpha 2^n$ bit operations, where n_C is the number of cubes in C. Step 7 runs in time $2^{\ell_{key}-(r_0+r_1)}$, where ℓ_{key} is the size of the key, when the equations are

Algorithm 3. Correlation Cube Attacks (Online Phase)

Require: a cube set C and $\Omega = \{(c, g, b) | \Pr(g = b|f_c) > p\}$

1: Set G_0 and G_1 to empty sets;
2: For each cube c in C do:
3: Randomly generate α values from free non-cube public bits, and request $\alpha 2^n$ keystream bits (or ciphertexts) corresponding to the cube c of size n and these non-cube public values;
4: Compute the α values of the superpoly f_c over the cube c;
5: If all the values of f_c equal 0, then set $G_0 = G_0 \cup \{g = 0|(c, g, 0) \in \Omega\}$, otherwise set $G_1 = G_1 \cup \{g = 1|(c, g, 1) \in \Omega\}$;
6: Deal with the case that $\{g|g = 0 \in G_0$ and $g = 1 \in G_1\}$ is not empty;
7: Randomly choose r_0 equations from G_0 and r_1 equations from G_1, solve these $r_0 + r_1$ equations and check whether the solutions are correct;
8: Repeat Step 7 if none of the solutions is correct.

balanced and easy to be solved. We can estimate the probability $q > p^{(r_0+r_1)}$ that a trial successes, so the expected number of trials is $q^{-1} < p^{-(r_0+r_1)}$. Here we require that the total number of equations in G_0 and G_1 is greater than $p^{-(r_0+r_1)}$. The expected time of online phase is thus less than

$$n_C \alpha 2^n + p^{-(r_0+r_1)} 2^{\ell_{key}-(r_0+r_1)}.$$

3.3 Discussion

The crux point of the attack is finding a low-degree basis of the superpoly over a given cube, that is, finding Q with low degree such that

$$f_c = \bigoplus_{g \in Q} g \cdot f_g.$$

Theoretical Bound on the Probability. We first discuss the conditional probability $\Pr(g = 1|f_c \neq 0)$, *i.e.*, for the case that f_c is not a zero constant for a fixed key. For any x such that $f_c(x) \neq 0$, there is at least one g such that $g(x) = 1$, that is, $\prod_{g \in Q}(g(x) + 1) = 0$. Specially, if Q contains only one function g, then $g(x) = 1$ holds with probability 1. If Q contains two functions g_1 and g_2, then we have $g_1(x) = 1$ or $g_2(x) = 1$, and thus at least one of them holds with probability $\geq \frac{2}{3}$. Generally, if Q contains n_Q functions, then there is at least one $g(x) = 1$ that holds with probability $\geq \frac{2^{n_Q-1}}{2^{n_Q}-1}$, under the condition $f_c \neq 0$.

The conditional probability $p_0 = \Pr(g = 0|f_c = 0)$ can be computed according to $p_1 = \Pr(g = 1|f_c \neq 0)$ and the probability γ that $f_c \neq 0$, *i.e.*, $p_0 = \frac{\frac{1}{2}-(1-p_1)\gamma}{1-\gamma}$.

When the upper bound $d(Q)$ on algebraic degree of f_c restricted to $\{g = 0|g \in Q\}$ is tight in Algorithm 2, we expect that f_g is not a zero constant for $g \in Q$. If all the functions f_g's depend on the free non-cube bits, then f_c is

a zero constant for a fixed key if and only if $g = 0$ holds with probability 1 (or close to 1) for all $g \in Q$.

Assuming that the event of (g, f_1) and the event of (g, f_2) are statistically independent, we have

$$\begin{aligned}\Pr(g = b|f_1, f_2) &= \frac{\Pr(g = b, f_1, f_2)}{\Pr(g = b, f_1, f_2) + \Pr(g = b + 1, f_1, f_2)} \\ &= \frac{\Pr(g = b, f_1)\Pr(g = b, f_2)}{\Pr(g = b, f_1)\Pr(g = b, f_2) + \Pr(g = b + 1, f_1)\Pr(g = b + 1, f_2)} \\ &= \frac{\Pr(g = b|f_1)\Pr(g = b|f_2)}{\Pr(g = b|f_1)\Pr(g = b|f_2) + \Pr(g = b + 1|f_1)\Pr(g = b + 1|f_2)}.\end{aligned}$$

Denote by ε_1 and ε_2 the correlation coefficients of $g = b$ given f_1 and f_2 respectively, *i.e.*, $\Pr(g = b|f_i) = \frac{1}{2}(1 + \varepsilon_i)$ for $i = 1, 2$. Then

$$\varepsilon = \frac{\varepsilon_1 + \varepsilon_2}{1 + \varepsilon_1\varepsilon_2}$$

is the correlation coefficient of the event that $g = b$ given both f_1 and f_2. Specially, if ε_1 and ε_2 have the same sign, then

$$|\varepsilon| = |\frac{\varepsilon_1 + \varepsilon_2}{1 + \varepsilon_1\varepsilon_2}| \geq \max\{|\varepsilon_1|, |\varepsilon_2|\}.$$

Our experiments on Trivium show that the assumption is reasonable. In fact, we do not expect that the assumption is perfectly true. The independence assumption is used to guarantee a bound on correlation coefficient. We believe the bound is sound, at least for the case that the correlations have the same sign, even if the assumption is not true in general.

Modifications of the Attack. We may slightly modify the online phase of the attack if necessary. As mentioned above, for the case that f_c is not a zero constant for a fixed key, we have $\prod_{g \in Q}(g(x) + 1) = 0$. We may make use of this kind of equations at Step 7 in Algorithm 3. Another modification is to use two different threshold probabilities separately for the equation sets G_0 and G_1 rather than the same one p.

4 Applications to Trivium Stream Cipher

In this section, we first give a brief description of the stream cipher Trivium [8], as well as recall the technique for estimating the degree of Trivium based on numeric mapping, and then apply the correlation cube attack to two variants of Trivium when the number of initialization rounds is reduced from 1152 to 805 and 835. At the end of this section, we will discuss the possible improvements, and partially apply our analysis techniques to the stream ciphers TriviA-SC [5,6] and Kreyvium [4].

4.1 Description of Trivium

A Brief Description of Trivium-Like Ciphers. Let A, B and C be three registers with sizes of n_A, n_B and n_C, denoted by A_t, B_t and C_t their corresponding states at clock t,

$$A_t = (x_t, x_{t-1}, \cdots, x_{t-n_A+1}), \tag{1}$$
$$B_t = (y_t, y_{t-1}, \cdots, y_{t-n_B+1}), \tag{2}$$
$$C_t = (z_t, z_{t-1}, \cdots, z_{t-n_C+1}), \tag{3}$$

and respectively updated by the following three quadratic functions,

$$x_t = z_{t-r_C} \cdot z_{t-r_C+1} + \ell_A(s^{(t-1)}), \tag{4}$$
$$y_t = x_{t-r_A} \cdot x_{t-r_A+1} + \ell_B(s^{(t-1)}), \tag{5}$$
$$z_t = y_{t-r_B} \cdot y_{t-r_B+1} + \ell_C(s^{(t-1)}), \tag{6}$$

where $1 \le r_\lambda < n_\lambda$ for $\lambda \in \{A, B, C\}$ and ℓ_A, ℓ_B and ℓ_C are linear functions. We denote $A_t[i] = x_i$, $B_t[i] = y_i$ and $C_t[i] = z_i$, and define $g_A^{(t)} = z_{t-r_C} \cdot z_{t-r_C+1}$, $g_B^{(t)} = x_{t-r_A} \cdot x_{t-r_A+1}$ and $g_C^{(t)} = y_{t-r_B} \cdot y_{t-r_B+1}$. The internal state, denoted by $s^{(t)}$ at clock t, consists of the three registers A, B, C, that is, $s^{(t)} = (A_t,\ B_t,\ C_t)$. Let f be the output function. After an initialization of N rounds, in which the internal state is updated for N times, the cipher generates a keystream bit by $f(s^{(t)})$ for each $t \ge N$.

The stream ciphers Trivium (designed by De Cannière and Preneel [8]) and TriviA-SC (designed by Chakraborti *et al.* [5,6]) exactly fall into this kind of ciphers. Kreyvium [4] is a variant of Trivium with 128-bit security, designed by Canteaut *et al.* at FSE 2016 for efficient homomorphic-ciphertext compression. Compared with Trivium, Kreyvium uses two extra registers (K^*, V^*) without updating but shifting, *i.e.*, $s^{(t)} = (A_t,\ B_t,\ C_t, K^*, V^*)$, and add a single bit of (K^*, V^*) to each of ℓ_A and ℓ_B, where K^* and V^* only involve the key bits and IV bits respectively. Trivium uses an 80-bit key and an 80-bit initial value (IV), while Kreyvium and TriviA-SC both use a 128-bit key and a 128-bit IV. All these ciphers have 1152 rounds.

A Brief Description of Trivium. Trivium contains a 288-bit internal state with three NFSRs of different lengths. The key stream generation consists of an iterative process which extracts the values of 15 specific state bits and uses them both to update 3 bits of the state and to compute 1 bit of key stream. The algorithm is initialized by loading an 80-bit key and an 80-bit IV into the 288-bit initial state, and setting all remaining bits to 0, except for three bits. Then, the state is updated for $4 \times 288 = 1152$ rounds, in the same way as explained above, but without generating key stream bits. This is summarized in the pseudo-code below.

$(x_0, x_{-1}, \cdots, x_{-92}) \leftarrow (k_0, k_1, \cdots, k_{79}, 0, \cdots, 0)$
$(y_0, y_{-1}, \cdots, y_{-83}) \leftarrow (iv_0, iv_1, \cdots, iv_{79}, 0, \cdots, 0)$
$(z_0, z_{-1}, \cdots, z_{-110}) \leftarrow (0, \cdots, 0, 1, 1, 1)$
for i **from** 1 **to** N **do**
$x_i = z_{i-66} + z_{i-111} + z_{i-110} \cdot z_{i-109} + x_{i-69}$
$y_i = x_{i-66} + x_{i-93} + x_{i-92} \cdot x_{i-91} + y_{i-78}$
$z_i = y_{i-69} + y_{i-84} + y_{i-83} \cdot y_{i-82} + z_{i-87}$
if $N > 1152$ **then**
$ks_{i-1152} = z_{i-66} + z_{i-111} + x_{i-66} + x_{i-93} + y_{i-69} + y_{i-84}$
end if
end for

4.2 Degree Estimation of Trivium

In this section, we recall the algorithm proposed by Liu [22] for estimating algebraic degree of the output of f after N rounds for a Trivium-like cipher, as described in Algorithm 4.

This algorithm first computes the exact algebraic degrees of the internal states for the first N_0 rounds, where the degrees of the functions $g_A^{(t)}$, $g_B^{(t)}$ and $g_C^{(t)}$ are also recorded, then iteratively compute $D^{(t)}$ for $t = N_0+1, N_0+2, \cdots, N$, and finally apply the numeric mapping to calculate an estimated degree for the first bit of the keystream. In Algorithm 4, three sequences, denoted by d_A, d_B and d_C, are used to record the estimated degrees of the three registers A, B, C. In each step of a Trivium-like cipher, three bits are updated. Accordingly, the estimated degrees for these three bits in each step t are calculated, denoted by $d_A^{(t)}, d_B^{(t)}$ and $d_C^{(t)}$. Then update $D^{(t)}$ from $D^{(t-1)}$. For estimating the algebraic degrees of x_t, y_t, z_t, the two procedures `DegMul`* and `DEG` deal with their "quadratic" and "linear" parts separately. The procedure `DegMul`* is used to compute an upper bound on the algebraic degree of $g_A^{(t)} = z_{t-r_C} \cdot z_{t-r_C+1}$, $g_B^{(t)} = x_{t-r_A} \cdot x_{t-r_A+1}$ and $g_C^{(t)} = y_{t-r_B} \cdot y_{t-r_B+1}$. It has been demonstrated in [22] that for all t with $1 \leq t \leq N$ the estimated degrees $d_A^{(t)}, d_B^{(t)}, d_C^{(t)}$ for x_t, y_t, z_t are greater than or equal to their corresponding algebraic degrees, and therefore the output `DEG`$(f, D^{(N)})$ of Algorithm 4 gives an upper bound on algebraic degree of the N-round output bit of a Trivium-like cipher.

The algorithm has a linear time and space complexity on N, if we do not take into account the time and memory used for computing the exact algebraic degrees of the internal states for the first N_0 rounds.

4.3 The Attack on 805-Round Trivium

Generating a Candidate Set of Favorite Cubes. A favorite cube of size 37 was found in [22] for distinguishing attacks on Trivium. We exhaustively search the subcubes with size 28 of this cube, and pick up the subcubes such that the corresponding superpolys after 790 rounds are zero constants (*i.e.*, the output bits after 790 rounds do not achieve maximum algebraic degree over the subcube variables), by using Algorithm 4 with $N_0 = 0$. Then we find 5444 such subcubes.

Algorithm 4. Estimation of Degree of Trivium-Like Ciphers [22]

Require: Given the ANFs of all internal states (A_t, B_t, C_t) with $t \leq N_0$, and the set of variables X.

For λ in $\{A, B, C\}$ do:
  For t from $1 - n_\lambda$ to 0 do:
    $d_\lambda^{(t)} \leftarrow \deg(\lambda_0[t], X)$; // $A_i[t] = x_t$, $B_i[t] = y_t$ and $C_i[t] = z_t$

$D^{(0)} \leftarrow (d_A^{(1-n_A)}, \cdots, d_A^{(0)}, d_B^{(1-n_B)}, \cdots, d_B^{(0)}, d_C^{(1-n_C)}, \cdots, d_C^{(0)})$;

/* Compute the exact algebraic degrees of the internal states for the first N_0 rounds */

For t from 1 to N_0 do:
  For λ in $\{A, B, C\}$ do:
    $dm_\lambda^{(t)} \leftarrow \deg(g_\lambda^{(t)}, X)$;
    $d_\lambda^{(t)} \leftarrow \deg(\lambda_t[t], X)$;
  $D^{(t)} \leftarrow (d_A^{(t-n_A+1)}, \cdots, d_A^{(t)}, d_B^{(t-n_B+1)}, \cdots, d_B^{(t)}, d_C^{(t-n_C+1)}, \cdots, d_C^{(t)})$;

/* Iteratively compute the upper bounds on algebraic degrees of the internal states for the remaining rounds */

For t from $N_0 + 1$ to N do:
  For λ in $\{A, B, C\}$ do:
    $dm_\lambda^{(t)} \leftarrow \texttt{DegMul}^*(g_\lambda^{(t)})$;
    $d_\lambda^{(t)} \leftarrow \max\{dm_\lambda^{(t)}, \texttt{DEG}(\ell_\lambda, D^{(t-1)})\}$;
  $D^{(t)} \leftarrow (d_A^{(t-n_A+1)}, \cdots, d_A^{(t)}, d_B^{(t-n_B+1)}, \cdots, d_B^{(t)}, d_C^{(t-n_C+1)}, \cdots, d_C^{(t)})$;

return $\texttt{DEG}(f, D^{(N)})$.

/* Description of the procedure $\texttt{DegMul}^*(g_\lambda^{(t)})$ for $\lambda \in \{A, B, C\}$ */

procedure $\texttt{DegMul}^*(g_\lambda^{(t)})$
  $t_1 \leftarrow t - r_{\rho(\lambda)}$; // $\rho(A) = C, \rho(C) = B, \rho(B) = A$
  If $t_1 \leq 0$ then:
return $d_{\rho(\lambda)}^{(t_1)} + d_{\rho(\lambda)}^{(t_1+1)}$.
  $t_2 \leftarrow t_1 - r_{\rho^2(\lambda)}$;
  $d_1 \leftarrow \min\{d_{\rho^2(\lambda)}^{(t_2)} + dm_{\rho(\lambda)}^{(t_1+1)}, d_{\rho^2(\lambda)}^{(t_2+2)} + dm_{\rho(\lambda)}^{(t_1)}, d_{\rho^2(\lambda)}^{(t_2)} + d_{\rho^2(\lambda)}^{(t_2+1)} + d_{\rho^2(\lambda)}^{(t_2+2)}\}$;
  $d_2 \leftarrow \texttt{DEG}(\ell_{\rho(\lambda)}, D^{(t_1)}) + dm_{\rho(\lambda)}^{(t_1)}$;
  $d_3 \leftarrow \texttt{DEG}(\ell_{\rho(\lambda)}, D^{(t_1-1)}) + d_{\rho(\lambda)}^{(t_1+1)}$;
  $d \leftarrow \max\{d_1, d_2, d_3\}$;
  return d.
end procedure

Finding the Basis and Free Non-cube IV Bits. We apply the procedure `Decomposition` to each cube c from the 5444 candidates, setting all the non-cube IV bits to zeros. Note here that we use Algorithm 4 in the procedure `Decomposition` to find an upper bound $d(Q)$ on the algebraic degree. Once a non-trivial basis of f_c is found, we set one of the non-cube IV bits to a parameter variable, and apply `Decomposition` again. This bit is considered as a free IV bit if it does not affect the basis, and otherwise we set it to 0. Then we add another non-cube IV bit to be a parameter variable, and do this again. By this way, we obtain a set of free non-cube IV bits.

Using this method, we get 47 cubes of size 28 satisfying that a basis of the superpoly after 805 rounds can be found. To make the attack more efficient, we further search the cubes whose superpolys after 805 rounds have a basis containing at most two elements. Once such cubes are found, we modify them by randomly shifting and changing some indexes, and test them by the same method. After computations within a dozen hours on a desktop computer, we are able to find more than 100 cubes whose superpolys after 805 rounds have a basis with one or two elements.

Computing the Probability. We test 32 out of these cubes, each of which has a different basis after 805 rounds, according to Step 4 of Algorithm 1. In each test, we compute the values of the superpoly f_c for 1000 random keys and at most $\alpha = 16$ non-cube IVs for each key, and evaluate the conditional probability $\Pr(g = 0|f_c(key, \cdot) \equiv 0)$ and $\Pr(g = 1|f_c(key, \cdot) \not\equiv 0)$ for a random fixed key, where g is a function depending on key bits in the basis of f_c and $f_c(key, \cdot)$ denotes the function f_c restricted at a fixed key. Our experiment shows that all the computations need about $13.5 \cdot 1000 \cdot 32 \cdot 2^{28} \approx 2^{47}$ cipher operations. We remind the readers that, in our experiment, we take all the possible values of the first $\log_2(\alpha) = 4$ free non-cube IV bits and set random values for the other free non-cube IV bits. Once we observe a non-zero value of the superpoly f_c, we skip the remaining IVs and continue to compute for the next key. On average, we need to compute 13.5 IVs for each key.

The results are listed in Table 3 in Appendix, together with the cubes, free non-cube IV bits and the equations. Note that 4 out of the 32 cubes are excluded from the table due to their little impact in our attack. In Table 3, by $p(0|0)$ (resp., $p(1|1)$) we mean the conditional probability of $g = 0$ (resp., $g = 1$) when the superpoly f_c is a zero constant (resp., not a zero constant) for a fixed key, by $p_{f_c \neq 0}$ we denote the probability that the superpoly f_c is not a zero constant for a fixed key, and #Rds is the number of rounds. We set the estimate threshold value of the probability to $\sigma = \frac{1+\sqrt{10/N_s}}{2}$, where N_s is the number of the samples, and set the attack threshold value of the probability, *i.e.*, the minimum probability used in the attack, to $p_0 = 0.6$ and $p_1 = 0.7$ for $\Pr(g = 0|f_c = 0)$ and $\Pr(g = 1|f_c \neq 0)$ respectively. The probability below the estimate threshold value σ is marked with slash throughs, *e.g.*, ~~0.574~~, and will never be used in the attack. The probability with a strikethrough, *e.g.*, ~~0.568~~,

is below the attack threshold value p_0 or p_1. From the experimental results, we derive two sets

$$\Omega_0 = \{(c, g, 0) | \Pr(g = 0 | f_c = 0) > p_0\}$$

and

$$\Omega_1 = \{(c, g, 1) | \Pr(g = 1 | f_c \neq 0) > p_1\}.$$

All the functions g's are either linear or quadratic. We also record all the equations with probability 1,

$$\Lambda = \{(c, g, b) | \Pr(g = b | f_c = 0) = 1 \text{ or } \Pr(g = b | f_c \neq 0) = 1\}.$$

As shown in Table 3, there are 11 cubes with a basis that contains only one linear or quadratic function. As discussed in Sect. 3.3, if their superpolys are not zero constants for a fixed key, then the sole function in the basis is always equal to one. In addition, for the 19th cube in the table, we observe that one of the functions in its basis is always equal to one given $f_c \neq 0$, *i.e.*, the conditional probability $\Pr(g_4 = 1 | f_c \neq 0) = 1$. The remaining 16 cubes have a basis that contains two linear or quadratic functions. The number of rounds ranges from 805 to 808.

We have also verified for 100 random keys, each with 16 IVs, that the superpolys of the cubes listed in the table are zero constants when imposing all the functions in their bases to zeros.

Recovering the Key in Online Phase. In this phase, we set C to the set of the 28 cubes listed in Table 3, $\Omega = \Omega_0 \cup \Omega_1$, and $\alpha = 16$. Then execute Algorithm 3 as described in Sect. 3.2. For avoidance of repetition, here we only show some necessary details that are not included in Algorithm 3. Remind that, in Step 3 of the algorithm, we take all the possible values of the first $\log_2(\alpha) = 4$ free non-cube IV bits, and set the other free non-cube IV bits to random values. The non-free non-cube IV bits are set to zeros. In Step 5, we update the equation sets G_0 and G_1 according to the values of f_c, and use an extra set E to collect the equations with probability 1 according to Λ. In Step 6, for the case that $\{g | g = 0 \in G_0 \text{ and } g = 1 \in G_1\}$ is not empty, we retain the one with higher probability between $g = 0$ and $g = 1$, and remove the other one. Meanwhile, we remove the equations in E from G_0 and G_1. In Step 7, we set r_i to the maximum r_i such that $p_i^{-r_i} < \binom{|G_i|}{r_i}$, where $|G_i|$ is the cardinality of G_i, $i = 0, 1$. Then randomly choose r_0 equations from G_0 and r_1 equations from G_1, solve these $r_0 + r_1$ equations together with E and check whether the solutions are correct.

Note that all the equations are linearly independent and can be linearized after guessing the values of some key bits. The expected time complexity of the online phase is less than

$$28 \times 13.5 \times 2^{28} + p_0^{-r_0} p_1^{-r_1} 2^{80-(r_0+r_1+|E|)} \approx 2^{37} + 2^{80-(\frac{1}{4}r_0+\frac{1}{2}r_1+|E|)}.$$

As shown in Table 3, the probability $p_{f_c \neq 0}$ of non-zero superpoly ranges from 0.036 to 0.113 for the 12 cubes that can generate deterministic equations.

Our experiments show that, for about 45% keys, there is at least one cube c out of these 12 cubes such that $f_c \neq 0$. For such 45% keys, the average number of such non-zero superpolys is around 2, and the maximum number is 7. In our experiments for 1000 keys, the average values of r_0 and r_1 are respectively 3.8 and 2.4, and the average value of $\frac{1}{4}r_0 + \frac{1}{2}r_1 + |E|$ is about 3. In other words, we can recover 7 equations on key bits by 2^4 trials on average. The average attack time is thus around 2^{77}, using 2^{37} keystream bits and at the expense of preprocessing time 2^{47}. The attack time on 805-round TRIVIUM can be cut down by using more cubes and at the expense of more preprocessing time and higher data complexity. Our attack is valid for more than half of 1000 random keys in our experiments. The attack fails when none of the systems of $r_0+r_1+|E|$ equations derived from G_0 and G_1 are correct. We stress here that the success probability of the attack can be increased by using smaller systems of equations (smaller r_0 and r_1) or larger probability thresholds (larger p_0 and p_1), at the cost of more attack time.

Next we give an example of the attack procedure. Note here that in the example the time complexity is better than the average case.

Example 3. Given that the 80-bit secret key is 71 DB 8B B3 21 CD AE F9 97 84 in hexadecimal, where the most significant bit is k_0 and the least significant bit is k_{79}. For each of the 28 cubes in Table 3, we generate 16 different non-cube IVs according to its free IV bits, and request 16×2^{28} keystream bits corresponding to this cube and the non-cube IVs, then compute the values of its superpoly. Taking the 8-th cube as an instance, we set the four free IV bits $0, 8, 53, 54$ to all possible values, the other free IV bits to random values, and the remaining non-cube IV bits to zeros; we then request 2^{32} keystream bits of 806 rounds accordingly, and sum these bits over the cube (module 2); finally we find a non-zero sum and get a deterministic equation $g_6 = k_{63} = 1$. We request $28 \times 16 \times 2^{28} \approx 2^{37}$ keystream bits in total, and find that there are 9 cubes having zero superpolys,

$$1, 2, 4, 10, 11, 16, 19, 24, 25,$$

and 19 cubes whose superpolys are not zero constants,

$$3, 5, 6, 7, 8, 9, 12, 13, 14, 15, 17, 18, 20, 21, 22, 23, 26, 27, 28.$$

From Table 3, we obtain 6 deterministic equations by the cubes $3, 5, 6, 7, 8, 9$.

$$E = \{g_i = 1 | i \in \{2, 6, 7, 11, 12, 13\}\},$$

where

$$\begin{aligned}
g_2 &= k_{59},\\
g_6 &= k_{63},\\
g_7 &= k_{64},\\
g_{11} &= k_{66} \cdot k_{67} + k_{41} + k_{68},\\
g_{12} &= k_{67} \cdot k_{68} + k_{42} + k_{69},\\
g_{13} &= k_{68} \cdot k_{69} + k_{43} + k_{70}.
\end{aligned}$$

Further, we derive two equation sets G_0 and G_1

$$G_0 = \{g_5 = k_{63} = 0\},\ G_1 = \{g_i = 1 | i \in \{3, 4, 8, 14\}\},$$

where

$$\begin{aligned} g_3 &= k_{60}, \\ g_4 &= k_{61}, \\ g_8 &= k_{34} \cdot k_{35} + k_9 + k_{36}, \\ g_{14} &= k_{69} \cdot k_{70} + k_{44} + k_{71}. \end{aligned}$$

We can see that all the equations are linear after guessing the values of three bits k_{35}, k_{67} and k_{69}. The equation $g_5 = 0$ in G_0 holds with probability 0.643, and the equations $g_i = 1$ in G_1 hold with probability $0.888, 0.735, 0.907, 0.799$ respectively for $i = 3, 4, 8, 14$. Accordingly, we have $r_0 = 0$ and $r_1 = 3$. Then we randomly choose 3 equations from G_1, solve a system of $3 + 6$ equations (together with the 6 equations in E), and repeat this step until the correct solution is found. In theory, the expected number of trials for finding the correct solution is less than 3. As a matter of fact, all the equations in G_1 but $g_4 = 1$ are true for the secret key, which means that we could find the correct key by at most 4 trials of solving a system of 9 equations. Therefore we can recover the key with time complexity of $2^{37} + 4 \times 2^{71} \approx 2^{73}$. The time complexity can be cut down to 2^{72} if we set $r_0 + r_1 = 4$ and exploit the equations in G_0 and G_1 together.

4.4 The Attack on 835-Round Trivium

Generating a Candidate Set of Favorite Cubes. In [22], an exhaustive search was done on the cubes of size $37 \leq n \leq 40$ that contain no adjacent indexes, by using a simplified version of Algorithm 4. Similarly, we exhaustively search the cubes of size $36 \leq n \leq 40$ that contain no adjacent indexes, and pick up the cubes such that the corresponding superpolys after 815 rounds are zero constants. Then we find 37595 and 3902 cubes of sizes 36 and 37 respectively that satisfy the requirement. There are also a number of such cubes of size higher than 37. This step is done in a few hours on a desktop computer.

Finding the Basis and Free Non-cube IV Bits. As done before, we apply the procedure `Decomposition` to each cube c from the candidate set, and also obtain a set of free non-cube IV bits. We then get 1085 and 99 cubes of sizes 36 and 37 such that a basis of the superpoly after 833 rounds can be found. The maximum number of rounds after which we can still find a basis is 841. No basis is found for the superpoly after 833 rounds of the cubes with size higher than 37 in the candidate set. The results are found in several hours on a desktop computer.

Computing the Probability. Computing the value of the superpoly f_c over a big cube is time consuming. We test 13 cubes of size 37 and 28 cubes of size 36, each of which has a different basis with less than 8 elements after 835 rounds. In each test, we compute the values of the superpoly f_c for 128 random keys with at most $\alpha = 8$ non-cube IVs, and evaluate the conditional probability $\Pr(g = 0|f_c(key, \cdot) \equiv 0)$ and $\Pr(g = 1|f_c(key, \cdot) \not\equiv 0)$ for a random fixed key. The values of non-cube IVs are taken in the same manner as done in Sect. 4.3. Our experiment shows that all the computations need about $6{\cdot}128{\cdot}(13{\cdot}2^{37}+28{\cdot}2^{36}) \approx 2^{51}$ cipher operations. On average, we need to compute 6 IVs for each key.

The results are listed in Tables 4 and 5 in Appendix, together with the cubes, free non-cube IV bits and the equations. We set the attack threshold value of the probability to $p = \frac{2}{3}$ for both $\Pr(g = 0|f_c = 0)$ and $\Pr(g = 1|f_c \neq 0)$. The probability below the estimate threshold value σ is marked with slash throughs, *e.g.*, ~~0.571~~, and will never be used in the attack. The probability with a strikethrough, *e.g.*, ~~0.654~~, is below the attack threshold value p. From the experimental results, we derive one set

$$\Omega = \{(c, g, b)|\Pr(g = b|f_c = 0) > p \text{ or } \Pr(g = b|f_c \neq 0) > p\}.$$

All the functions g's are either linear or quadratic. We also record all the equations with probability 1,

$$\Lambda = \{(c, g, b)|\Pr(g = b|f_c = 0) = 1 \text{ or } \Pr(g = b|f_c \neq 0) = 1\}.$$

As shown in Table 4 for the cubes of size 37, there are 2 cubes having a basis that contains only one function, while there are 5 cubes from which it is possible to set up an equation with probability 1. The third and 11-th cubes have no qualified equations, and will be discarded in online phase. The 13-th and 14-th cubes are the same, while the keystream bits of two different numbers of rounds, 835 and 840, are used.

The results for the cubes of size 36 are listed in Table 5, and there are 7 cubes that have no qualified equations and will be discarded in online phase.

We have also verified for 32 random keys, each with 4 IVs, that the superpolys of the cubes listed in the table sum to zeros when imposing all the functions in their bases to zeros.

Recovering the Key in Online Phase. In this phase, we set $\alpha = 8$, and then execute Algorithm 3. Remind that, in Step 3 of the algorithm, we take all the possible values of the first $\log_2(\alpha) = 3$ free non-cube IV bits, and set the other free non-cube IV bits to random values. The non-free non-cube IV bits are set to zeros. In Step 5, we update the equation sets $G = G_0 \cup G_1$ according to the values of f_c, and use an extra set E to collect the equations with probability 1 according to Λ. In Step 6, if G has two incompatible equations $g = 0$ and $g = 1$, we remove them both from G. Meanwhile, we remove the equations in E from G. In Step 7, we set r to the maximum r such that $p^{-r} < \binom{|G|}{r}$. Then randomly

choose r equations from G, solve these r equations together with E and check whether the solutions are correct.

Note that all the equations are linearly independent and can be linearized after guessing the values of some key bits. The expected time complexity of this phase is less than

$$6 \times (12 \times 2^{37} + 21 \times 2^{36}) + p^{-r} 2^{80-(r+|E|)} \approx 2^{44} + 2^{80-(\frac{2}{5}r+|E|)}.$$

As shown in Table 4, the probability for the 5 cubes that can generate equations with probability 1 ranges from 0.008 to 0.297. Our experiments show that, for about half keys, we can generate from one to three equations with probability 1. In our experiments for 128 random keys, the average values of r is larger than 10, and the average value of $\frac{2}{5}r + |E|$ is about 5. In other words, we can recover 11 equations on key bits by 2^6 trials on average. The average attack time is thus around 2^{75}, using $8 \times (12 \times 2^{37} + 21 \times 2^{36}) \approx 2^{45}$ keystream bits and at the expense of preprocessing time 2^{51}. The attack is valid for more than 44% out of 128 random keys in our experiments. The attack time can be cut down by using more cubes and at the expense of more preprocessing time and more data. On the other hand, the success probability of the attack can be increased at the cost of more attack time.

4.5 Discussion

Improvements of the Attack. A natural method to cut down the attack time is to use more cubes with keystream bits of different numbers of rounds. While we have found a thousand potentially favorite cubes for Trivium reduced to from 833 to 841 rounds, we can make use of a small number of them due to a limited computation resource. Increasing the number α in online phase gives a higher chance to find deterministic equations. Testing more random keys with larger α in preprocessing phase gives a more accurate estimate of the conditional probability $\Pr(g|f_c)$, as well as generates more valid probabilistic equations. One may also exploit one of the two equations $g = 0$ and $g = 1$ by carefully computing the probability $\Pr(g = b|f_1, f_2)$ as discussed in Sect. 3.3, when both of them appear in the equation set G.

For the attack on Trivium reduced to less than 835 rounds, it is possible to cut down the attack time by using cubes of size less than 36 and combining the equations retrieved in Sect. 4.4. For instance, using 54 out of the 69 cubes in Tables 3, 4 and 5 gives an improved key recovery attack on 805-round Trivium. In this improved attack, we adopt the same strategy that was used for analysis of 835-round Trivium, and find that we can recover 14 equations by 2^7 trials on average. Thus the attack on 805-round Trivium is faster than an exhaustive search by a factor of around 2^7, using 2^{45} keystream bits and at the expense of preprocessing time 2^{51}. The attack is directly valid for 31% out of 128 random keys in our experiments. The attack also works for most of the remaining keys after increasing the probability threshold p and repeating the attack again.

Table 2. Success probability of the attack

805 rounds:	#key bits	7.2	6.9	6.5	6.1	5.7
	Success rate	31%	60%	77%	86%	93%
835 rounds:	#key bits	5.0	4.6	4.2	3.8	3.4
	Success rate	44%	72%	83%	95%	98%

Success Probability of the Attack. In the above attack, we maximize the system of probabilistic equations. This is achieved by setting r to the maximum r such that $p^{-r} < \binom{|G|}{r}$ in Algorithm 3. The attack works for more keys when a smaller system with fewer equations is used, *i.e.*, a smaller r is adopted. We have verified this by supplementary experiments on round-reduced Trivium.

As shown in Table 2, for 805-round variant of Trivium, we can deduce 7.2, 6.9, 6.5, 6.1 and 5.7 key bits on average for 31%, 60%, 77%, 86% and 93% of the keys, respectively; and for 835-round variant, we can deduce 5.0, 4.6, 4.2, 3.8 and 3.4 key bits on average for 44%, 72%, 83%, 95% and 98% of the keys, respectively. Actually, our experiments for 128 random keys show that we can always set up equations. As shown in Tables 3 and 4, there are many cubes (e.g., Cube 5 in Table 3) such that we can set up probabilistic equations from both sides, which implies that the attack works for a random key.

Applications to TriviA-SC and Kreyvium. We apply our techniques to TriviA-SC and Kreyvium, and can find some cubes whose superpolys after 1047 and 852 rounds have a low-degree basis with a few elements for TriviA-SC and Kreyvium respectively. The cubes for TriviA-SC have size larger than 60, and for Kreyvium the size is at least 54. Computing the conditional probability $\Pr(g|f_c)$ for such large cubes is infeasible for us. Though we are unable to fully verify the validity of the attack on TriviA-SC and Kreyvium, we believe that there is a high chance of validness due to their similar structures with Trivium.

5 Conclusions

In this paper, we have shown a general framework of a new model of cube attacks, called *correlation cube attack*. It is a generalization of conditional cube attack, as well as a variant of conditional differential attacks. As an illustration, we applied it to Trivium stream cipher, and gained the best key recovery attacks for Trivium. To the best of our knowledge, this is the first time that a weak-key distinguisher on Trivium stream cipher can be converted to a key recovery attack. We believe that this new cryptanalytic tool is useful in both cryptanalysis and design of symmetric cryptosystems. In the future, it is worthy of working on its applications to more cryptographic primitives, such as the `Grain` family of stream ciphers, block cipher Simon and hash function Keccak.

Acknowledgments. We are grateful to the anonymous reviewers for their valuable comments.

A The Cubes, Equations and Probabilities

Table 3. The cubes, equations and probabilities in the attack on 805-round Trivium (16 IVs and 1000 keys for cube size 28)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
1	2, 4, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 57, 62, 79	0, 1, 9, 54, 55, 58, 59, 60, 64, 65, 66, 67, 68, 69, 72, 73, 74, 77	g_{14}	~~0.503~~	1	0.078	805
2	2, 4, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 62, 73, 79	0, 9, 54, 55, 56, 64, 65, 66, 67, 68, 69, 70, 71, 75, 77	g_3	~~0.534~~	1	0.036	805
3	2, 4, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 62, 77, 79	0, 1, 9, 54, 55, 56, 57, 58, 59, 60, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74	g_7	~~0.508~~	1	0.071	805
4	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 61, 74, 78	53, 54, 55, 56, 57, 63, 64, 65, 66, 67, 68, 69, 70, 71, 76	g_4	~~0.550~~	1	0.061	805
5	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 56, 61, 78	0, 8, 53, 54, 57, 58, 59, 63, 64, 65, 66, 67, 68, 71, 72, 73, 76	g_{13}	~~0.534~~	1	0.093	806
6	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 61, 78	0, 8, 55, 56, 57, 58, 59, 62, 63, 64, 65, 66, 69, 70, 71, 72, 73, 76	g_{11}	~~0.514~~	1	0.056	806
7	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 61, 72, 78	8, 53, 54, 55, 63, 64, 65, 66, 67, 68, 69, 70, 74, 76	g_2	~~0.509~~	1	0.041	806
8	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 61, 76, 78	0, 8, 53, 54, 55, 56, 57, 58, 59, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73	g_6	~~0.547~~	1	0.080	806
9	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 55, 60, 79	7, 52, 53, 56, 57, 58, 61, 62, 64, 65, 66, 67, 70, 71, 72, 75, 76	g_{12}	~~0.568~~	1	0.113	807
10	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 60, 79	7, 54, 55, 56, 57, 58, 61, 62, 64, 65, 68, 69, 70, 71, 72, 75, 76	g_{10}	~~0.520~~	1	0.061	807
11	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 60, 75, 79	7, 52, 53, 54, 55, 56, 57, 58, 61, 62, 64, 65, 66, 67, 68, 69, 70, 71, 72, 76	g_5	~~0.556~~	1	0.097	807
12	2, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 57, 62, 77, 79	0, 1, 3, 4, 9, 54, 55, 58, 59, 60, 64, 65, 66, 67, 68, 69, 72, 73, 74	g_7 g_{14}	~~0.594~~ ~~0.587~~	0.832 0.748	0.286	805
13	2, 4, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 55, 57, 62, 79	0, 1, 5, 6, 9, 58, 59, 60, 64, 65, 66, 67, 70, 72, 73, 74, 77	g_{12} g_{14}	~~0.575~~ 0.621	~~0.654~~ 0.799	0.309	805
14	2, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 55, 62, 77, 79	0, 1, 3, 4, 9, 56, 57, 58, 59, 60, 63, 64, 65, 66, 67, 70, 71, 72, 73, 74	g_7 g_{12}	~~0.573~~ ~~0.587~~	0.803 0.721	0.269	805
15	1, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 61, 72, 74, 78	2, 3, 53, 54, 55, 56, 57, 63, 64, 65, 66, 67, 68, 69, 70, 76	g_2 g_4	~~0.533~~ ~~0.573~~	0.708 0.735	0.185	805
16	2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 60, 62, 75, 79	0, 1, 9, 52, 53, 54, 55, 56, 57, 58, 64, 65, 66, 67, 68, 69, 70, 71, 72, 77	g_5 g_{16}	~~0.558~~ ~~0.512~~	0.902 ~~0.673~~	0.122	805
17	1, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 56, 61, 76, 78	0, 2, 3, 8, 53, 54, 57, 58, 59, 63, 64, 65, 66, 67, 68, 71, 72, 73	g_6 g_{13}	0.622 ~~0.580~~	0.778 0.744	0.297	806
18	1, 3, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 56, 61, 78	0, 4, 5, 8, 57, 58, 59, 63, 64, 65, 66, 69, 71, 72, 73, 76	g_{11} g_{13}	~~0.580~~ 0.635	~~0.697~~ 0.805	0.343	806

Table 3. (*continued*)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
19	1, 3, 5, 7, 9, 11, 14, 16, 18, 20, 22, 24, 26, 29, 31, 33, 35, 37, 39, 41, 44, 46, 48, 50, 61, 70, 74, 78	0, 8, 51, 52, 53, 59, 62, 63, 64, 65, 66, 67, 68, 72, 76	g_1 g_4	~~0.501~~ ~~0.568~~	~~0.511~~ 1	0.092	806
20	1, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 61, 76, 78	0, 2, 3, 8, 55, 56, 57, 58, 59, 62, 63, 64, 65, 66, 69, 70, 71, 72, 73	g_6 g_{11}	0.607 ~~0.547~~	0.792 ~~0.692~~	0.260	806
21	0, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 61, 78	1, 8, 14, 55, 56, 57, 58, 59, 62, 63, 64, 65, 66, 69, 70, 71, 72, 73, 76	g_8 g_{11}	0.694 ~~0.542~~	0.907 ~~0.629~~	0.334	806
22	2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 55, 60, 75, 79	0, 7, 52, 53, 56, 57, 58, 61, 62, 64, 65, 66, 67, 70, 71, 72, 73, 76	g_5 g_{12}	0.660 0.705	~~0.691~~ 0.742	0.450	807
23	2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 60, 79	0, 7, 56, 57, 58, 61, 62, 64, 65, 68, 70, 71, 72, 73, 74, 75, 76	g_{10} g_{12}	0.636 0.762	~~0.665~~ 0.762	0.492	807
24	2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 60, 75, 79	0, 7, 54, 55, 56, 57, 58, 61, 62, 64, 65, 66, 67, 68, 69, 70, 71, 72, 76	g_5 g_{10}	0.643 ~~0.558~~	0.815 ~~0.669~~	0.308	807
25	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 60, 64, 79	7, 52, 53, 54, 55, 56, 57, 58, 61, 62, 65, 66, 67, 68, 69, 70, 71, 72, 75, 76	g_{17} g_{18}	~~0.553~~ ~~0.496~~	0.801 ~~0.585~~	0.246	807
26	0, 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 38, 40, 43, 45, 47, 49, 58, 60, 73, 79	7, 50, 51, 52, 53, 54, 55, 56, 61, 62, 64, 65, 66, 67, 68, 69, 70, 75, 76	g_3 g_{15}	~~0.592~~ ~~0.499~~	0.888 ~~0.650~~	0.160	807
27	1, 3, 5, 7, 9, 11, 14, 16, 18, 20, 22, 24, 26, 29, 31, 33, 35, 37, 39, 41, 44, 46, 48, 50, 54, 59, 74, 78	6, 51, 52, 55, 56, 57, 60, 61, 63, 64, 65, 66, 69, 70, 71, 72, 75	g_4 g_{11}	0.652 0.696	~~0.641~~ 0.760	0.463	808
28	1, 3, 5, 7, 9, 11, 14, 16, 18, 20, 22, 24, 26, 29, 31, 33, 35, 37, 39, 41, 44, 46, 48, 50, 52, 54, 59, 78	6, 55, 56, 57, 60, 61, 63, 64, 67, 69, 70, 71, 72, 73, 74, 75	g_9 g_{11}	0.644 0.766	~~0.636~~ 0.787	0.508	808

$g_1 = k_{57}$
$g_2 = k_{59}$
$g_3 = k_{60}$
$g_4 = k_{61}$
$g_5 = k_{62}$
$g_6 = k_{63}$
$g_7 = k_{64}$
$g_8 = k_{34} \cdot k_{35} + k_9 + k_{36}$
$g_9 = k_{64} \cdot k_{65} + k_{39} + k_{66}$
$g_{10} = k_{65} \cdot k_{66} + k_{40} + k_{67}$
$g_{11} = k_{66} \cdot k_{67} + k_{41} + k_{68}$
$g_{12} = k_{67} \cdot k_{68} + k_{42} + k_{69}$
$g_{13} = k_{68} \cdot k_{69} + k_{43} + k_{70}$
$g_{14} = k_{69} \cdot k_{70} + k_{44} + k_{71}$
$g_{15} = k_{70} \cdot k_{71} + k_{45} + k_{72}$
$g_{16} = k_{72} \cdot k_{73} + k_{47} + k_{74}$
$g_{17} = k_{76} \cdot k_{77} + k_{51} + k_{78}$
$g_{18} = k_{67} \cdot k_{68} + k_{76} \cdot k_{77} + k_0 + k_{42} + k_{51} + k_{68} + k_{78}$

Table 4. The cubes, equations and probabilities in the attack on 835-round Trivium (8 IVs and 128 keys for cube size 37)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
1	2, 4, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 55, 57, 59, 62, 64, 66, 68, 70, 72, 74, 77, 79	0, 1, 3, 5, 7, 9	g_{16}	0.472	1	0.008	836
2	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 56, 58, 61, 63, 65, 67, 69, 71, 73, 76, 78	0, 2, 4, 6, 8, 79	g_{15}	0.524	1	0.016	837
3	0, 2, 4, 7, 9, 11, 13, 15, 17, 19, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 75, 79	1, 3, 5, 6, 8, 10, 77	g_5 g_{47}	0.554 0.653	0.714 0.857	0.055	835
4	0, 2, 4, 6, 8, 11, 13, 15, 17, 19, 21, 23, 26, 28, 30, 32, 34, 36, 38, 41, 43, 45, 47, 49, 51, 53, 56, 58, 60, 62, 64, 66, 68, 71, 73, 75, 79	1, 3, 5, 7, 9, 10, 77	g_{24} g_{26}	0.500 0.459	0.667 1	0.047	835
5	1, 3, 5, 7, 10, 12, 14, 16, 18, 20, 22, 25, 27, 29, 31, 33, 35, 37, 40, 42, 44, 46, 48, 50, 52, 55, 57, 59, 61, 63, 65, 67, 70, 72, 74, 76, 78	0, 2, 4, 6, 8, 9, 11	g_{17} g_{25} g_{27}	0.696 0.478 0.543	0.972 0.444 0.694	0.281	835
6	0, 2, 4, 6, 9, 11, 13, 15, 17, 19, 21, 24, 26, 28, 30, 32, 34, 36, 39, 41, 43, 45, 47, 49, 51, 54, 56, 58, 60, 62, 64, 66, 69, 71, 73, 75, 79	1, 3, 5, 7, 8, 10, 77	g_{16} g_{24} g_{26}	0.667 0.478 0.889	1 0.474 0.447	0.297	836
7	0, 2, 5, 7, 9, 11, 13, 15, 17, 20, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 75, 79	1, 3, 4, 6, 8, 10, 33, 76, 77	g_{31} g_{33} g_{46} g_{47} g_{49}	0.962 0.962 0.615 0.808 0.346	0.627 0.627 0.529 0.422 0.480	0.797	835
8	0, 2, 4, 7, 9, 11, 13, 15, 17, 19, 21, 24, 26, 28, 30, 32, 34, 36, 39, 41, 43, 45, 47, 49, 51, 54, 56, 58, 60, 62, 64, 66, 69, 71, 73, 75, 79	1, 3, 5, 6, 8, 10, 20, 77	g_{20} g_{21} g_{33} g_{35} g_{36}	0.968 0.968 0.419 0.452 0.419	0.619 0.691 0.485 0.485 0.515	0.758	835
9	0, 2, 4, 6, 9, 11, 13, 15, 17, 19, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 75, 79	3, 5, 7, 8, 10, 20, 76, 77	g_{31} g_{33} g_{35} g_{36} g_{46} g_{47}	0.610 0.644 0.627 0.458 0.492 0.729	0.594 0.638 0.609 0.522 0.493 0.464	0.539	835
10	0, 2, 5, 7, 9, 11, 13, 15, 17, 19, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 75, 79	3, 4, 6, 8, 10, 18, 76, 77	g_{20} g_{31} g_{33} g_{35} g_{36} g_{46} g_{47}	0.716 0.597 0.582 0.612 0.522 0.537 0.776	0.689 0.607 0.607 0.623 0.590 0.541 0.541	0.477	835
11	0, 2, 4, 7, 9, 11, 13, 15, 17, 20, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 75, 79	5, 6, 8, 10, 18, 33, 76, 77	g_{19} g_{20} g_{31} g_{33} g_{46} g_{47} g_{49}	0.508 0.703 0.703 0.703 0.486 0.730 0.486	0.484 0.549 0.582 0.593 0.495 0.418 0.516	0.711	835
12	2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 64, 66, 68, 70, 72, 75, 77, 79	0, 1, 3, 5, 7, 13	g_8 g_{10} g_{12} g_{14} g_{16} g_{18} g_{45}	0.569 0.538 0.585 0.600 0.508 0.708 0.615	0.571 0.603 0.651 0.619 0.571 0.778 0.524	0.492	839

Table 4. (*continued*)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
13	1, 3, 5, 7, 9, 11, 14, 16, 18, 20, 22, 24, 26, 29, 31, 33, 35, 37, 39, 41, 44, 46, 48, 50, 52, 54, 56, 59, 61, 63, 65, 67, 69, 71, 74, 76, 78	0, 2, 4, 6, 8, 12	g_{15}	0.509	0.438	0.125	835
			g_{17}	0.580	1		
14		0, 2, 4, 6, 12	g_7	0.561	0.606	0.555	840
			g_9	0.632	0.493		
			g_{13}	0.614	0.535		
			g_{15}	0.754	0.676		
			g_{17}	0.754	0.690		
			g_{44}	0.561	0.507		
			g_{50}	0.649	0.524		

$g_1 = k_{54}$
$g_2 = k_{55}$
$g_3 = k_{56}$
$g_4 = k_{57}$
$g_5 = k_{58}$
$g_6 = k_{59}$
$g_7 = k_{61}$
$g_8 = k_{62}$
$g_9 = k_{63}$
$g_{10} = k_{64}$
$g_{11} = k_{65}$
$g_{12} = k_{78} \cdot k_{79} + k_{53}$
$g_{13} = k_{30} \cdot k_{31} + k_5 + k_{32}$
$g_{14} = k_{31} \cdot k_{32} + k_6 + k_{33}$
$g_{15} = k_{32} \cdot k_{33} + k_7 + k_{34}$
$g_{16} = k_{33} \cdot k_{34} + k_8 + k_{35}$
$g_{17} = k_{34} \cdot k_{35} + k_9 + k_{36}$
$g_{18} = k_{35} \cdot k_{36} + k_{10} + k_{37}$
$g_{19} = k_{38} \cdot k_{39} + k_{13} + k_{40}$
$g_{20} = k_{40} \cdot k_{41} + k_{15} + k_{42}$
$g_{21} = k_{42} \cdot k_{43} + k_{17} + k_{44}$
$g_{22} = k_{44} \cdot k_{45} + k_{19} + k_{46}$
$g_{23} = k_{45} \cdot k_{46} + k_{20} + k_{47}$
$g_{24} = k_{46} \cdot k_{47} + k_{21} + k_{48}$
$g_{25} = k_{47} \cdot k_{48} + k_{22} + k_{49}$
$g_{26} = k_{48} \cdot k_{49} + k_{23} + k_{50}$
$g_{27} = k_{49} \cdot k_{50} + k_{24} + k_{51}$
$g_{28} = k_{50} \cdot k_{51} + k_{25} + k_{52}$
$g_{29} = k_{51} \cdot k_{52} + k_{26} + k_{53}$
$g_{30} = k_{52} \cdot k_{53} + k_{27} + k_{54}$
$g_{31} = k_{53} \cdot k_{54} + k_{28} + k_{55}$
$g_{32} = k_{54} \cdot k_{55} + k_{29} + k_{56}$
$g_{33} = k_{55} \cdot k_{56} + k_{30} + k_{57}$
$g_{34} = k_{56} \cdot k_{57} + k_{31} + k_{58}$
$g_{35} = k_{57} \cdot k_{58} + k_{32} + k_{59}$
$g_{36} = k_{59} \cdot k_{60} + k_{34} + k_{61}$
$g_{37} = k_{61} \cdot k_{62} + k_{36} + k_{63}$
$g_{38} = k_{62} \cdot k_{63} + k_{37} + k_{64}$
$g_{39} = k_{63} \cdot k_{64} + k_{38} + k_{65}$
$g_{40} = k_{64} \cdot k_{65} + k_{39} + k_{66}$
$g_{41} = k_{65} \cdot k_{66} + k_{40} + k_{67}$
$g_{42} = k_{66} \cdot k_{67} + k_{41} + k_{68}$
$g_{43} = k_{67} \cdot k_{68} + k_{42} + k_{69}$
$g_{44} = k_{68} \cdot k_{69} + k_{43} + k_{70}$
$g_{45} = k_{69} \cdot k_{70} + k_{44} + k_{71}$
$g_{46} = k_{72} \cdot k_{73} + k_{47} + k_{74}$
$g_{47} = k_{74} \cdot k_{75} + k_{49} + k_{76}$
$g_{48} = k_{75} \cdot k_{76} + k_{50} + k_{77}$
$g_{49} = k_{76} \cdot k_{77} + k_{51} + k_{78}$
$g_{50} = k_{77} \cdot k_{78} + k_{52} + k_{79}$

Table 5. The cubes, equations and probabilities in the attack on 835-round Trivium (8 IVs and 128 keys for cube size 36)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
1	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 56, 58, 61, 63, 65, 67, 69, 71, 76, 78	0, 2, 4, 6, 73	g_3	0.530	0.536	0.219	835
			g_5	0.630	0.786		
2	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 41, 43, 46, 48, 50, 52, 54, 56, 58, 61, 63, 65, 67, 69, 71, 73, 76, 78	0, 2, 4, 6, 8, 38, 39	g_1	0.431	0.375	0.438	836
			g_3	0.653	0.661		
			g_5	0.639	0.589		
			g_{50}	0.556	0.446		
3	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 56, 58, 61, 63, 65, 67, 69, 73, 76, 78	0, 2, 4, 6, 8, 71	g_1	0.523	0.500	0.328	836
			g_3	0.698	0.857		
			g_{40}	0.535	0.619		
			g_{50}	0.593	0.524		
4	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 54, 56, 58, 61, 63, 65, 69, 71, 73, 76, 78	0, 2, 4, 6, 8, 67	g_3	0.570	0.595	0.328	836
			g_5	0.616	0.619		
			g_{40}	0.523	0.595		
			g_{50}	0.605	0.548		
5	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 64, 66, 68, 70, 72, 75, 79	1, 3, 5, 7, 37, 38, 77	g_2	0.554	0.762	0.492	837
			g_4	0.585	0.683		
			g_{12}	0.431	0.492		
			g_{49}	0.492	0.524		

Table 5. (*continued*)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
6	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 66, 68, 70, 72, 75, 79	1, 3, 5, 7, 64, 77	g_2	0.500	0.741	0.422	837
			g_4	0.554	0.685		
			g_{12}	0.473	0.537		
			g_{39}	0.486	0.648		
7	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 64, 68, 70, 72, 75, 79	1, 3, 5, 7, 66, 77	g_2	0.507	0.755	0.414	837
			g_4	0.573	0.717		
			g_{39}	0.493	0.660		
			g_{49}	0.480	0.509		
8	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 64, 66, 68, 72, 75, 79	1, 3, 5, 7, 70, 77	g_2	0.613	0.906	0.414	837
			g_{12}	0.480	0.547		
			g_{39}	0.480	0.642		
			g_{49}	0.533	0.585		
9	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 64, 66, 70, 72, 75, 79	1, 3, 5, 7, 68, 77	g_4	0.602	0.875	0.312	837
			g_{12}	0.477	0.550		
			g_{39}	0.489	0.700		
			g_{49}	0.466	0.477		
10	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 43, 46, 48, 50, 52, 56, 58, 61, 63, 65, 67, 69, 71, 73, 76, 78	0, 2, 4, 6, 8, 54	g_1	0.423	0.840	0.391	836
			g_3	0.641	0.680		
			g_5	0.641	0.620		
			g_{40}	0.538	0.600		
			g_{44}	0.564	0.540		
11	0, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 42, 45, 47, 49, 51, 53, 55, 57, 60, 62, 64, 66, 68, 70, 72, 75, 79	1, 2, 3, 5, 7, 77	g_2	0.493	0.712	0.461	837
			g_4	0.565	0.678		
			g_{12}	0.406	0.458		
			g_{39}	0.507	0.661		
			g_{49}	0.536	0.576		
12	0, 2, 4, 6, 8, 10, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 38, 40, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 70, 73, 75, 79	1, 3, 5, 11, 12, 13, 77	g_2	0.529	0.750	0.469	839
			g_{12}	0.588	0.667		
			g_{37}	0.441	0.583		
			g_{47}	0.588	0.333		
			g_{49}	0.544	0.583		
13	2, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 44, 47, 49, 51, 53, 55, 57, 59, 62, 64, 66, 68, 70, 72, 74, 77, 79	0, 1, 3, 4, 5, 7, 9	g_2	0.453	0.656	0.500	835
			g_4	0.562	0.656		
			g_6	0.562	0.703		
			g_{41}	0.578	0.703		
			g_{43}	0.656	0.562		
			g_{45}	0.641	0.547		
14	1, 4, 6, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 29, 31, 34, 36, 38, 40, 42, 44, 49, 51, 53, 55, 57, 59, 62, 64, 66, 68, 70, 72, 74, 77, 79	0, 2, 3, 5, 7, 9, 45, 46, 47	g_2	0.463	0.632	0.680	835
			g_4	0.683	0.657		
			g_6	0.780	0.736		
			g_{12}	0.488	0.540		
			g_{34}	0.732	0.678		
			g_{41}	0.561	0.621		
15	0, 2, 4, 7, 9, 11, 13, 15, 17, 19, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 67, 69, 71, 73, 75, 79	5, 6, 8, 10, 63, 64, 76	g_{11}	0.821	0.753	0.695	835
			g_{33}	0.692	0.596		
			g_{35}	0.692	0.584		
			g_{36}	0.410	0.506		
			g_{46}	0.564	0.528		
			g_{47}	0.795	0.449		
16	0, 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 36, 38, 40, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 70, 73, 75, 79	1, 3, 5, 7, 9, 11, 33, 34, 77	g_2	0.590	0.685	0.695	835
			g_4	0.590	0.607		
			g_{12}	0.718	0.640		
			g_{26}	0.615	0.640		
			g_{28}	0.487	0.584		
			g_{30}	0.487	0.506		
17	0, 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 38, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 70, 73, 75, 79	1, 3, 5, 7, 9, 11, 39, 40, 41, 77	g_2	0.533	0.643	0.766	835
			g_4	0.533	0.571		
			g_{12}	0.533	0.551		
			g_{24}	0.700	0.571		
			g_{26}	0.733	0.653		
			g_{28}	0.667	0.633		

Table 5. (*continued*)

No.	Cube Indexes	Free IV bits	Eqs.	$p(0\|0)$	$p(1\|1)$	$p_{f_c\neq 0}$	#Rds
18	0, 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 40, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 70, 73, 75, 79	1, 3, 5, 7, 9, 11, 37, 38, 77	g_2	0.429	0.616	0.672	835
			g_4	0.595	0.616		
			g_{12}	0.643	0.616		
			g_{24}	0.619	0.570		
			g_{26}	0.714	0.698		
			g_{30}	0.643	0.584		
19	0, 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 38, 40, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 68, 70, 73, 75, 79	1, 3, 5, 7, 9, 11, 66, 77	g_2	0.551	0.696	0.617	835
			g_4	0.469	0.557		
			g_{24}	0.592	0.570		
			g_{26}	0.592	0.658		
			g_{28}	0.531	0.620		
			g_{30}	0.653	0.608		
20	0, 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 38, 40, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 73, 75, 79	1, 3, 5, 7, 9, 11, 70, 77	g_2	0.564	0.674	0.695	835
			g_{12}	0.538	0.562		
			g_{24}	0.615	0.562		
			g_{26}	0.692	0.674		
			g_{28}	0.564	0.618		
			g_{30}	0.641	0.573		
21	0, 3, 5, 7, 9, 11, 13, 15, 18, 20, 22, 24, 26, 28, 30, 33, 35, 37, 39, 41, 43, 48, 50, 52, 54, 56, 58, 61, 63, 65, 67, 69, 71, 73, 76, 78	1, 2, 4, 6, 8, 44, 45, 46, 79	g_1	0.486	0.473	0.727	836
			g_3	0.629	0.527		
			g_5	0.714	0.527		
			g_{33}	0.829	0.634		
			g_{40}	0.514	0.527		
			g_{50}	0.657	0.484		
22	2, 4, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29, 32, 34, 36, 38, 40, 42, 47, 49, 51, 53, 55, 57, 60, 62, 64, 66, 68, 70, 72, 75, 77, 79	0, 1, 3, 5, 7, 43, 44, 45	g_2	0.620	0.744	0.609	837
			g_4	0.620	0.654		
			g_{12}	0.380	0.474		
			g_{32}	0.720	0.641		
			g_{39}	0.440	0.577		
			g_{49}	0.500	0.526		
23	1, 3, 5, 7, 9, 11, 13, 16, 18, 20, 22, 24, 26, 28, 31, 33, 35, 37, 39, 41, 46, 48, 50, 52, 54, 56, 59, 61, 63, 65, 67, 69, 71, 74, 76, 78	0, 2, 4, 6, 42, 43, 44	g_1	0.673	0.603	0.570	838
			g_3	0.709	0.630		
			g_{31}	0.818	0.740		
			g_{38}	0.582	0.493		
			g_{48}	0.436	0.384		
			g_{50}	0.636	0.507		
24	0, 2, 4, 6, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 30, 32, 34, 36, 38, 40, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 70, 73, 75, 79	1, 3, 5, 41, 42, 43, 76	g_2	0.590	0.685	0.695	839
			g_{12}	0.615	0.596		
			g_{30}	0.714	0.618		
			g_{37}	0.487	0.596		
			g_{47}	0.692	0.404		
			g_{49}	0.513	0.528		
25	0, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 25, 28, 30, 32, 34, 36, 38, 40, 43, 45, 47, 49, 51, 53, 55, 58, 60, 62, 64, 66, 68, 70, 73, 75, 79	1, 2, 3, 5, 7, 9, 11, 77	g_2	0.522	0.671	0.641	835
			g_4	0.522	0.585		
			g_{12}	0.543	0.573		
			g_{24}	0.609	0.573		
			g_{26}	0.565	0.634		
			g_{28}	0.543	0.622		
			g_{30}	0.652	0.598		
26	1, 5, 7, 9, 11, 14, 16, 18, 20, 22, 24, 26, 29, 31, 33, 35, 37, 39, 41, 44, 46, 48, 50, 52, 54, 56, 59, 61, 63, 65, 67, 69, 71, 74, 76, 78	0, 3, 4, 6, 8, 12	g_{23}	0.577	0.596	0.445	835
			g_{25}	0.634	0.667		
			g_{27}	0.535	0.596		
			g_{29}	0.521	0.491		
			g_{31}	0.493	0.491		
			g_{42}	0.535	0.456		
			g_{44}	0.521	0.474		
27	0, 2, 4, 7, 9, 11, 13, 15, 17, 19, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 79	3, 5, 6, 8, 75, 77	g_{20}	0.592	0.519	0.617	835
			g_{21}	0.551	0.582		
			g_{31}	0.592	0.557		
			g_{33}	0.837	0.722		
			g_{35}	0.755	0.658		
			g_{36}	0.408	0.494		
			g_{44}	0.592	0.519		
28	0, 2, 4, 6, 8, 11, 13, 15, 17, 19, 21, 23, 26, 28, 30, 32, 34, 36, 38, 41, 43, 45, 47, 49, 51, 53, 56, 58, 60, 62, 64, 66, 68, 71, 73, 79	1, 3, 5, 7, 9, 75, 77	g_2	0.581	0.660	0.758	837
			g_{12}	0.548	0.557		
			g_{22}	0.710	0.567		
			g_{24}	0.839	0.619		
			g_{26}	0.613	0.619		
			g_{28}	0.581	0.608		
			g_{49}	0.581	0.546		

References

1. Aumasson, J., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128. IACR Cryptology ePrint Archive 2009:218 (2009)
2. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference, January 2011. http://keccak.noekeon.org, Version 3.0
4. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 313–333. Springer, Heidelberg (2016)
5. Chakraborti, A., Chattopadhyay, A., Hassan, M., Nandi, M.: TriviA: a fast and secure authenticated encryption scheme. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 330–353. Springer, Heidelberg (2015)
6. Chakraborti, A., Nandi, M.: TriviA-ck-v2. CAESAR Submission (2015). http://competitions.cr.yp.to/round2/triviackv2.pdf
7. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_20
8. De Cannière, C., Preneel, B.: Trivium. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_18
9. 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 2011. LNCS, vol. 7073, pp. 327–343. Springer, Heidelberg (2011)
10. 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 and Fischlin [26], pp. 733–761
11. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
12. Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011)
13. Englund, H., Johansson, T., Sönmez Turan, M.: A framework for chosen IV statistical analysis of stream ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
14. Fischer, S., Khazaei, S., Meier, W.: Chosen IV statistical analysis for key recovery attacks on stream ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
15. Fouque, P.-A., Vannet, T.: Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2014)
16. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: grain-128. In: 2006 IEEE International Symposium on Information Theory, pp. 1614–1618. IEEE (2006)

17. Hell, M., Johansson, T., Maximov, A., Meier, W.: The grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)
18. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round Keccak sponge function. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 259–288. Springer, Cham (2017)
19. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of NLFSR-based cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)
20. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of Trivium and KATAN. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 200–212. Springer, Heidelberg (2012)
21. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Proceedings Symposium in Communications, Coding Cryptography, pp. 227–233. Kluwer Academic Publishers (1994)
22. Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 227–249. Springer, Cham (2017)
23. Liu, M., Lin, D., Wang, W.: Searching cubes for testing Boolean functions and its application to Trivium. In: IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, 14–19 June 2015, pp. 496–500. IEEE (2015)
24. Maximov, A., Biryukov, A.: Two trivial attacks on Trivium. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 36–55. Springer, Heidelberg (2007)
25. Meier, W., Staffelbach, O.: Fast correlation attacks on certain stream ciphers. J. Cryptol. **1**(3), 159–176 (1989)
26. Oswald, E., Fischlin, M. (eds.): EUROCRYPT 2015. LNCS, vol. 9056. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5
27. 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, 7–10 August 2006, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications, pp. 260–266. INSTICC Press (2006)
28. Stankovski, P.: Greedy distinguishers and nonrandomness detectors. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 210–226. Springer, Heidelberg (2010)
29. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald and Fischlin [26], pp. 287–314
30. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017)
31. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016)
32. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. IACR Cryptology ePrint Archive, 2007:413 (2007)