A New Framework for Finding Nonlinear Superpolies in Cube Attacks against Trivium-Like Ciphers

Chen-Dong Ye, Tian Tian

National Digital Switching System Engineering & Technological Research Center,
P.O. Box 407, 62 Kexue Road, Zhengzhou, 450001, China.

Abstract. In this paper, we study experimental cube attacks against Trivium-like ciphers and we focus on improving nonlinear superpolies recovery. We first present a general framework in cube attacks to test nonlinear superpolies, by exploiting a kind of linearization technique. It worth noting that, in the new framework, the complexities of testing and recovering nonlinear superpolies are almost the same as those of testing and recovering linear superpolies. To demonstrate the effectiveness of our new attack framework, we do extensive experiments on Trivium, Kreyvium, and TriviA-SC-v2 respectively. We obtain several linear and quadratic superpolies for the 802-round Trivium, which is the best experimental results against Trivium regarding the number of initialization rounds. For Kreyvium, it is shown that the probability of finding a quadratic superpoly using the new framework is twice as large as finding a linear superpoly for Kreyvium. Hopefully, this new framework would provide some new insights on cube attacks against NFSR-based ciphers, and in particular make nonlinear superpolies potentially useful in the future cube attacks.

Keywords: Cube attacks, Linearity tests, Quadracity tests, Trivium-like ciphers.

1 Introduction

Trivium is a bit oriented synchronous stream cipher designed by Cannière and Preneel, which targets hardware environments with highly restricted resources, see [1]. It is one of the eSTREAM hardware-oriented finalists and an International Standard under ISO/IEC 29192-3:2012.

Since proposed, Trivium has attracted a lot of attention for its simplicity. As a result, there are many cryptanalytic results on Trivium such as key recovery attacks based on cube attacks [2,3,4,5,6], distinguishing attacks based on cube attacks [7,8,9,10,11], conditional differential attacks [12] and internal state recovery attacks [13]. Among these various cryptanalytic techniques, cube attacks are one of the most powerful tool against Trivium. It was proposed in EUROCRYPT 2009 by Dinur and

Sharmir [2]. In [2], the authors recovered 35 linear superpolies, i.e., relations between key bits and keystream bits, based on which they gave an effective key recovery attack on the 767-round Trivium with attack complexity 2^{45}. In [3], Mroczkowski and Szmidt applied cube attacks to the 709-round Trivium, and firstly reported quadratic superpolies. In specific, they found 41 linear superpolies and 22 quadratic superpolies. In [4], the authors presented a recursive method to find cubes with linear superpolies as well as took advantage of the Meobius transformation. They found 12 linear superpolies and 6 quadratic superpolies for the 799-round Trivium. In [5], Todo et al. applied the division property to cube attacks. Based on the division property, attackers could identify the key variables involved in the superpoly of a given cube by solving corresponding MILP models instead of performing linearity/quadraticity tests. Consequently, with some relatively large cubes, partial information of the secret key could be recovered for up to the 832-round Trivium. In [6], the authors proposed a technique to reduce the complexity of superpoly recovery based on the work of [5].

Due to the simplicity and the established security of Trivium, some recently proposed crypto primitives adopt similar designs, such as Kreyvium [14] and TriviA-SC [15,16].

Kreyvium is designed for the efficient homomorphic-ciphertext compression in homomorphic encryptions and aims to achieve 128-bit security. In [10], based on a cube of size 61, Liu presented a distinguisher on the 872-round Kreyvium. In [6], for the 888-round Kreyvium, the authors provided a key recovery attack based on a cube of size 102. In [17], based on 24-th and 25-th order conditional characteristic, the authors proposed distinguishers on 899-round Kreyvium.

TriviA-SC is the base component of the authenticated encryption algorithm TriviA which was a second-round candidate of CAESAR competition. In order to resist slide attacks, the original version (TriviA-SC-v1) was improved to TriviA-SC-v2 by changing the constants loaded to the initial internal state. Hereinafter, TriviA-SC means its both versions, if not specified. In [8], the authors proposed distinguishers for the 930-round TriviA-SC-v1 and the 950-round TriviA-SC-v2 respectively. Furthermore, the authors provided a slide attack on the full TriviA-SC-v2. In [10], based on cubes of sizes around 63, the author proposed distinguishers of the 1035-round TriviA-SC-v1, the 1046-round TriviA-SC-v2, and the full 1152-round of simplified TriviA-SC where the nonlinear term in the output bit was removed. In [18], for the full 1152 rounds simplified TriviA-SC, the authors found a linear distinguisher with a complexity 2^{120}.

Before the work of [5], cube attacks were based on experimental tests, which are called experimental cube attacks in this paper. In experimental cube attacks, it is important to find cubes with low-degree superpolies, and attackers usually attempt to find linear or quadratic superpolies in practice. The conventional method of recovering quadratic superpolies is doing quadraticity tests, which regards a cipher as a blockbox polynomial. Compared with linearity tests, quadraticity tests have a higher complexity, which makes experimentally implement quadraticity tests much more difficult than implementing linearity tests when the sizes of cubes reach 40. In this paper, we are concerned with finding nonlinear superpolies for Trivium-like ciphers.

1.1 Our Contribution

The inspiration of this paper comes from an interesting observation of cube attacks against Trivium. The algebraic normal form (ANF) of quadratic superpolies recovered in cube attacks against Trivium have fixed forms. They are very probable of the form $k_i k_{i+1} \oplus k_{i+2} \oplus k_{i-25(\bmod 69)}$ for $0 \leq i \leq 77$ together with $k_{78}k_{79} \oplus k_{53}$. Besides, this observation is also true for other Trivium-like ciphers. Hence, we propose to treat some nonlinear key expressions as a whole, and regard the first output bit as a tweakable function on these nonlinear key expressions not key variables themselves. Consequently nonlinear superpolies could be recovered by testing linearity on nonlinear key expressions. According to this basic idea, we propose a generic framework to recover nonlinear superpolies using linearity test principles for Trivium-like ciphers.

As illustrations, we perform extensive experiments on Trivium, Kreyvium, and TriviA-SC-v2 with our new framework. To show the correctness and effectiveness of our framework, for each of the variants with from 600 to 700 initialization rounds of these three ciphers, we search for linear and nonlinear superpolies based on 100 randomly chosen cubes. Table 1 shows the total number of nonlinear superpolies and linear superpolies we find. Note that, for all the three ciphers, the number of the nonlinear superpolies is non-ignorable compared with that of linear superpolies. In particular, in the case of Trivium and Kreyvium, the number of nonlinear superpolies is close to or even twice as large as that of linear superpolies.

Furthermore, with our framework we find several new superpoies for variants with higher initialization rounds. First, we reveal some new quadratic supeprolies of the 784- and the 799-round Trivium. Besides, we recover 5 linear superpolies and 2 quadratic superploies of the 802-round Trivium. Second, with a cube of size 38, 8 different quadratic superpolies but no linear superpolies are found for the 776-round Kreyvium. Third, we gain linear superpolies and quadratic superpolies for the 864-round TriviA-SC-v2 and the 992-round simplified TriviA-SC-v2, respectively. Table 2 summaries our results.

Table 1. The total number of linear superpolies and nonlinear superpolies

ciphers	# of linear superpolies	# of nonlinear superpolies
Trivium	8155985	7517944
Kreyvium	1194480	2538591
TriviA-SC-v2	4074914	491551

Table 2. Results on round-reduced Trivium-like stream ciphers

ciphers	# of rounds	# of superpolies
Trivium	802	5 linear, 2 quadratic
Kreyvium	776	0 linear, 8 quadratic
TriviA-SC-v2	864	12 linear, 3 quadratic
TriviA-SC-v2 simplified	992	14 linear, 2 quadratic

1.2 Organization

The rest of this paper is structured as follows. In Section 2, we introduce some basic definitions and facts. In Section 3, we propose a new framework to find nonlinear superpolies with a low complexity. In Section 4, our new framework is applied to Trivium-like stream ciphers. In Section 5, we summarize our work.

2 Preliminaries

In this section, we briefly review the general structure of Trivium-like ciphers, the basic procedures of cube attacks, and linearity/quadracity tests of black-box Boolean functions.

2.1 Trivium-Like Stream Ciphers

The main building block of a Trivium-like cipher is a Galois nonlinear feedback shift register, such that for every clock cycle there are three bits of the internal state updated by quadratic feedback functions and all the other bits of the internal sate are updated by shifting. In specific, let A, B and C be three shift registers of length L_A, L_B, and L_C respectively. For $t \geq 0$, let $A_t = (x_t, \ldots, x_{t+L_A-1})$, $B_t = (y_t, \ldots, y_{t+L_B-1})$, and $C_t = (z_t, \ldots, z_{t+L_C-1})$ denote the tth state of A, B and C respectively. Then the internal state of a Trivium-like cipher at time instance t is given by $s_t = (A_t, B_t, C_t)$, and the state update function could be described as

$$x_t = z_{t-r_c-1} \cdot z_{t-r_c} + l_A(s_{t-1}),$$

$$y_t = x_{t-r_a-1} \cdot x_{t-r_a} + l_B(s_{t-1}),$$

$$z_t = y_{t-r_b-1} \cdot y_{t-r_b} + l_C(s_{t-1}),$$

where l_A, l_B and l_C are three linear functions and $1 \leq r_\lambda \leq L_\lambda$ for $\lambda \in \{A, B, C\}$. After N initialization rounds, a filtering function f is used to compute a keystream bit from the current internal state, i.e., $z_t = f(s_t)$ for $t \geq N$.

There are three well-known Trivium-like ciphers, say Trivium [1], Kreyvium [14] , and TriviA-SC [15,16] . The first two algorithms well fulfill the description above, while the last algorithm uses two extra registers K^* and V^*, which are padded with key bits and IV bits respectively, to XOR the key bits and IV bits to the feedback function. Besides, the filtering functions of Trivium and Keryvium are linear, while that of TriviA-SC is quadratic.

2.2 Cube Attacks

The idea of cube attack was first proposed by Dinur and Shamir in [2]. In the cube attack against stream ciphers, an output bit z is described as a tweakable Boolean function f on secret key variables $Key = (k_0, k_1, \ldots, k_{n-1})$ and public IV variables $IV = (iv_0, iv_1, \ldots, iv_{m-1})$, where n and m are positive integers, i.e.,

$$z = f(Key, IV).$$

Let I be a subset of d public variables, where $1 \leq d \leq m$. Without loss of generality, we assume that $I = \{iv_0, iv_1, \ldots, iv_{d-1}\}$. Then the function f can be rewritten

$$f(Key, IV) = t_I \cdot p_I(Key, iv_d, iv_{d+1}, \ldots, iv_{m-1}) \oplus q(Key, IV),$$

where $t_I = iv_0 iv_1 \cdots iv_{d-1}$ is the product of variables in I, p_I does not contain any variable in I, and each term in q is not divisible by t_I. It can be seen that the summation of the 2^d functions derived from f by assigning all the possible values to the d variables in I is equal to p_I, which eliminates q completely, that is,

$$\bigoplus_{(iv_0, iv_1, \ldots, iv_{d-1}) \in \mathbb{F}_2^d} f(Key, IV) = p_I(Key, iv_d, iv_{d+1}, \ldots, iv_{m-1}). \qquad (1)$$

The variables in the set I are called *cube variables*, the set C_I of all 2^d possible assignments of the cube variables in I is called a *d-dimensional cube*, and the polynomial p_I is called the *superpoly* of I. That is to say $iv_0, iv_1, \ldots, iv_{d-1}$ are cube variables and $iv_d, iv_{d+1}, \ldots, iv_{m-1}$ are non-cube variables. Furthermore, fixing each non-cube variable to be a constant, the superpoly p_I becomes a polynomial with secrete key variables only. **In this paper, we always fix all non-cube variables to be 0's**, and so (1) can be rewritten

$$\bigoplus_{(iv_{i_1}, iv_{i_2}, \ldots, iv_{i_d}) \in \{0,1\}^d} f(Key, iv_0, \ldots, iv_{d-1}, \mathbf{0}) = p_{s(I)}(Key, \mathbf{0}). \qquad (2)$$

Since p_I comes from f, it follows that $\deg(p_I) \leq \deg(f)$. In particular, if $\deg(f) = d+1$, then the superpoly p_I must be affine or constant. The key idea of cube attacks is trying to recover simple or low-degree superpoly p_I.

A cube attack consists of two phases: a preprocessing phase which is independent of the secret key and a online phase which should be carried out for every secret key. In the preprocessing phase, attackers should find some useful superpolies which could recover the information of the secret key. In the online phase, the previously found superpolies were evaluated under the real key. By solving a system of equations or looking up the truth tables of superpolies, a part of or even the whole information of the secret key could be revealed.

In experimental cube attacks, the aim of the preprocessing phase is to find cubes with low-degree superpolies. In fact, only linear and quadratic superpolies were implemented in the previous experimental cube attacks. So far there is no deterministic method for finding a set of cube variables whose superpoly is linear/quadratic, and all proposed methods are more or less based on random search, see the random walk method in [2] and the recursive method in [4]. It means that an attacker needs to do linearity/quadraticity tests on superpolies for many sets of chosen cube variables to find sufficient many desirable linear/quadratic ones.

2.3 Linearity and Quadraticity Tests for Black-box Boolean Functions

In this subsection we briefly recall how to test linearity/quadraticity of a black-box Boolean function. As for linearity tests one can refer to [19] and as for quadraticity or low-degree tests one can refer to [20].

Let n be a positive integer and let $f(x_1, x_2, \ldots, x_n)$ be an n-variable Boolean function. Suppose that the explicit representation of $f(x_1, x_2, \ldots, x_n)$ is extremely big and it is unknown to us, but the function f can be queried, that is, we could enquire about the value $f(\boldsymbol{a})$ for any input vector $\boldsymbol{a} \in \mathbb{F}_2^n$.

Linearity test. Choose $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_2^n$ uniformly and independently, and verify

$$f(\boldsymbol{a} \oplus \boldsymbol{b}) \oplus f(\boldsymbol{a}) \oplus f(\boldsymbol{b}) \oplus f(\boldsymbol{0}) = 0. \tag{3}$$

If f is linear, then the test will succeed, whereas if $\deg(f) \geq 2$, then the test may fail with a certain probability. Thus the test should be repeated sufficiently many times to make sure that f is very close to being linear.

If f passes through the linearity test, then its coefficients in the algebraic normal form can be calculated by $n+1$ more queries. One query could determine one coefficient. The constant term of f is given by $f(\boldsymbol{0})$. Let

$$\boldsymbol{e}_1 = (1, 0, \ldots, 0), \boldsymbol{e}_2 = (0, 1, 0, \ldots, 0), \ldots, \boldsymbol{e}_n = (0, 0, \ldots, 1) \in \mathbb{F}_2^n.$$

Then the coefficient of the variable x_i in f for $1 \leq i \leq n$ is given by

$$c_i = f(\boldsymbol{e}_i) \oplus f(\boldsymbol{0}).$$

Quadraticity test. Choose $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}_2^n$ uniformly and independently, and verify

$$f(\boldsymbol{a}\oplus\boldsymbol{b}\oplus\boldsymbol{c})\oplus f(\boldsymbol{a}\oplus\boldsymbol{b})\oplus f(\boldsymbol{a}\oplus\boldsymbol{c})\oplus f(\boldsymbol{b}\oplus\boldsymbol{c})\oplus f(\boldsymbol{a})\oplus f(\boldsymbol{b})\oplus f(\boldsymbol{c})\oplus f(\boldsymbol{0}) = 0. \tag{4}$$

Similarly if f is quadratic, then the test succeeds, whereas if $\deg(f) > 2$, then the test may fail. Thus the test should be repeated sufficiently many times to make sure that f is very close to being quadratic.

If f passes through the quadraticity test, then the coefficient of a quadratic term $x_i x_j$ in f for $1 \leq i < j \leq n$ is given by

$$f(\boldsymbol{e}_i \oplus \boldsymbol{e}_j) \oplus f(\boldsymbol{e}_i) \oplus f(\boldsymbol{e}_j) \oplus f(\boldsymbol{0}).$$

Note that a constant function also satisfies the linearity test, and a linear function always satisfies the quadraticity test. Thus, sometimes in the attacks, we also need to test whether a linear function f is a constant function.

Constantness test. Choose $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_2^n$ uniformly and independently, and verify

$$f(\boldsymbol{a}) \oplus f(\boldsymbol{b}) = 0.$$

If f is constant, then the test will succeed, whereas if f is not constant, then the test may fail with a certain probability. Thus the test should be repeated sufficiently many times.

3 A New Framework to Find Nonlinear Superpolies

In this section, we introduce some observations from which we get our basic idea. Then based on this basic idea, we propose a new framework to find nonlinear superpolies.

3.1 Motivations

The motivations of this paper come from the following observations on the extensive superpolies recovered by the previous experimental cube attacks against Trivium. Please refer to [2], [3], and [4] for a large number of instances of superpolies for Trivium variants.

Our first observation is **the sparsity of nonlinear superpolies**. It can be easily observed that the algebraic normal forms of all recovered superpolies are very sparse, most of which have less than five terms. Accordingly the systems of nonlinear equations in key variables defined by these superpolies are easy to solve during the online phase, see [3] for an example.

Our second observation is that **some key variables are missing in linear superpolies**. It can be observed that none of the linear superpolies were found so far involving the key variables between k_{69} and k_{79}. This phenomenon is also mentioned in [4, Page 511]. Hence, to recover the information of the key variables between k_{69} and k_{79}, linear superpolies are not sufficient.

According to the above two observations, nonlinear superpolies are as useful as linear superpolies in cube attacks against Trivium-like ciphers, and exploiting nonlinear superpolies could definitely bring some merits to mounting cube attacks. However, comparing quadraticity tests with linearity tests, it can be seen that to do one verification in a quadraticity test, eight queries are needed (see (4)), while to do one verification in a linear test, only four queries are needed (see (3)). For instance, in experimental cube attacks against Trivium, as the number of initialization rounds increases, the dimension of a cube becomes very large, say 40, which results in that just one query will be very time consuming. In this case, to test a quadratic superpoly using present method is much more difficult than to test a linear superpoly. Thus, although quadratic superpolies are useful, they do not paly a role as important as linear superpolies.

Our third observation is **the fixed forms of nonlinear superpolies**. It is interesting to find that the ANFs of all nonlinear superpolies recovered in cube attacks against Trivium have very specific forms. That is to say they are far from random polynomials. It can be observed that most of the published quadratic superpolies only have one quadratic monomial of the form $x_i x_{i+1}$ accompanied by two degree 1 monomials. This observation was also mentioned in [4, Section 4.2], but was not utilized in their attacks.

Inspired by the third observation, we propose a new framework to find and recover nonlinear superpolies with low complexities. In the new framework, we fix some nonlinear key expressions, and find superpolies which are linear about these fixed nonlinear key expressions. It can be seen that linear superplies in this sense are *nonlinear* on key variables. There are two key points involved in the new framework. One is how to do linearity tests on superpolies about the fixed nonlinear key expressions. The other is how to choose useful nonlinear key expressions. We shall explain these two points in detail in the following two subsections respectively.

3.2 A Generic Technique for Linearity Tests of Composite Functions

Let $g(y_0, y_1, \ldots, y_{m-1})$ be a Boolean function on the variables $y_0, y_1, \ldots, y_{m-1}$. For $0 \le i \le m-1$, let $h_i(x_0, x_1, \ldots, x_{n-1})$ be a Boolean function on the variables $x_0, x_1, \ldots, x_{n-1}$. Then

$$f(x_0, x_1, \ldots, x_{n-1}) = g(h_0(x_0, x_1, \ldots, x_{n-1}), \ldots, h_{m-1}(x_0, x_1, \ldots, x_{n-1}))$$

is a composite function of $g(y_0, y_1, \ldots, y_{m-1})$ and $h_i(x_0, x_1, \ldots, x_{n-1})$. It is easy to see that when the composite function $f(x_0, x_1, \ldots, x_{n-1})$ is nonlinear, it is not necessary that g is nonlinear. In other words, when f is nonlinear on the variables $x_0, x_1, \ldots, x_{n-1}$, it is not necessary that f is nonlinear on the expressions $h_0, h_1, \ldots, h_{m-1}$. Let us take a small example.

Example 1. Let $f = x_0 \cdot x_1 \oplus x_2 \cdot x_3$ be a Boolean function. Let $h_0 = x_0 \cdot x_1$ and $h_1 = x_2 \cdot x_3$. It is clear that $f = h_0 \oplus h_1$. Hence f is linear on the expressions h_0 and h_1, but nonlinear on the variables x_0, x_1, x_2, x_3.

In the above small example, the ANF of $f(x_0, x_1, x_2, x_3)$ is known, and so it is easy to see whether f is linear on h_0 and h_1. Now the problem is when $f(x_0, x_1, \dots, x_{n-1})$ is a black-box Boolean function, how to test whether f is a linear Boolean function on $h_0, h_1, \dots, h_{m-1}$. Note that f could be queried only by assigning values to the variables $x_0, x_1, \dots, x_{n-1}$. We formally present this problem in the following.

Problem 1. Let $f(x_0, x_1, \dots, x_{n-1})$ be a black-box Boolean function. Assume that $h_0, h_1, \dots, h_{m-1}$ are m Boolean functions on the variables $x_0, x_1, \dots, x_{n-1}$ such that there is a Boolean function $g(y_0, y_1, \dots, y_{m-1})$ satisfying $f = g(h_0, h_1, \dots, h_{m-1})$. How to test whether f is linear about $h_0, h_1, \dots, h_{m-1}$ by querying $f(x_0, x_1, \dots, x_{n-1})$?

The difference between Problem 1 and the traditional linearity test of black-box Boolean functions lies in that we ask the linearity of a set of nonlinear expressions of inputting variables not simply inputting variables themselves. This general problem is open. In the following we give a simple technique to tackle some instances of the problem which is useful in the following attacks. Our basic idea is still the BLR linearity test introduced in Subsection 2.3.

Theorem 1. *Let $f, h_0, \dots, h_{m-1}$ be as described in Problem 1. If the mapping*

$$H : \boldsymbol{a} = (a_0, a_1, \dots, a_{n-1}) \mapsto (h_0(\boldsymbol{a}), h_1(\boldsymbol{a}), \dots, h_{m-1}(\boldsymbol{a})), \boldsymbol{a} \in \mathbb{F}_2^n,$$

is surjective with $H(\mathbf{0}) = \mathbf{0}$, then Algorithm 1 is a one-sided tester for f being linear on the expressions $h_0, h_1, \dots, h_{m-1}$. In particular, if Algorithm 1 returns reject, then f is not linear on the expressions $h_0, h_1, \dots, h_{m-1}$ with probability 1.

Proof. Since f is a composite function of the form

$$f = g(h_0, h_1, \dots, h_{m-1}),$$

if follows that f being linear on the given expressions $h_0, h_1, \dots, h_{m-1}$ is equivalent to $g(y_0, y_1, \dots, y_{m-1})$ is linear. Thus it suffices to show Algorithm 1 is actually a BLR linearity test on $g(y_0, y_1, \dots, y_{m-1})$.

Algorithm 1 Linearity test of composite functions

```
Require: a black-box function $f$ on $X = (x_0, x_1, \ldots, x_{n-1})$ and a vectorial Boolean
    function $H = (h_0(X), h_1(X), \ldots, h_{m-1}(X))$.
1: choose $\boldsymbol{a}$ and $\boldsymbol{b}$ randomly and uniformly in $\mathbb{F}_2^m$;
2: compute $X_1, X_2, X_3$ satisfying $H(X_1) = \boldsymbol{a}$, $H(X_2) = \boldsymbol{b}$, and $H(X_3) = \boldsymbol{a} \oplus \boldsymbol{b}$,
   respectively;
3: compute $v = f(X_1) \oplus f(X_2) \oplus f(X_3) \oplus f(\mathbf{0})$
4: if $v \neq 0$ then
5:     return reject;
6: else
7:     return accept;
8: end if
```

Let $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, X_1, X_2, and X_3 be as described in Algorithm 1, where the existence of X_1, X_2, X_3 can be deduced from the hypothesis that h is surjective. Then we have

$$f(X_1) = g(\boldsymbol{a}), f(X_2) = g(\boldsymbol{b}) \text{ and } f(X_3) = \boldsymbol{a} \oplus \boldsymbol{b}.$$

It follows that

$$f(X_1) \oplus f(X_2) \oplus f(X_3) \oplus f(\mathbf{0}) = g(\boldsymbol{a}) \oplus g(\boldsymbol{b}) \oplus g(\boldsymbol{a} \oplus \boldsymbol{b}) \oplus g(\mathbf{0}).$$

Hence it can be seen that line 3 in Algorithm 1 is a BLR linearity test for $g(y_0, y_1, \ldots, y_{m-1})$.

Remark 1. The probability that Algorithm 1 rejects a function f which is nonlinear on $h_0, h_1, \ldots, h_{m-1}$ is equal to the probability that the algorithm in [20] rejecting the corresponding function g which is nonlinear.

Algorithm 1 needs repeating sufficient times to make sure that f is very close to being linear on $h_0, h_1, \ldots, h_{m-1}$. When we make sure that f is linear on $h_0, h_1, \ldots, h_{m-1}$, we could recover the ANF of f using only $m+1$ queries like recovering a linear Boolean function, which is much less compared with $\frac{n(n-1)}{2} + n + 1$ queries to recover the ANF of a quadratic Boolean function on $x_0, x_1, \ldots, x_{n-1}$. It can be seen that the complexities of doing linearity test on f and the ANF recovery of f are almost the same as that of linearity tests and linear Boolean functions recovery except the time spent on finding a preimage of the mapping H. A preimage of the mapping H could be found by solving a system of equations defined by $h_0, h_1, \ldots, h_{m-1}$. When this system of equations is sparse and simple, it can be solved efficiently. That is the case in our attacks.

3.3 A Generic Method of Choosing Useful Nonlinear Key Expressions

When it comes to cube attacks, the composite function f discussed in the last subsection is a superpoly p_I of some chosen cube C_I. Traditionally, p_I is seen as a black-box Boolean function on key variables, say $k_0, k_1, \ldots, k_{n-1}$, and attackers try to recover linear superpolies on $k_0, k_1, \ldots, k_{n-1}$. If there exists a set of nonlinear expressions $h_0, h_1, \ldots,$ h_{m-1} in key variables such that p_I could be represented as a composite function $p_I = g(h_0, h_1, \ldots, h_{m-1})$ for some function g, then our new technique could efficiently test whether p_I is linear on the expressions $h_0, h_1, \ldots, h_{m-1}$ resulting in a desirable nonlinear superpoly in key variables. In the following, we shall show a generic method to find such useful nonlinear expressions in key variables.

During the initialization process of stream ciphers, key variables are gradually mixed with IV variables, and so in some early rounds, when the mixture is not sufficient, they may not be multiplied together. Namely, at some time instance t, each internal state bit s_i^t could be written as

$$s_i^t = g_{i,1}(IV) \oplus g_{i,2}(Key)(0 \leq i \leq l-1),$$

where l is the size of the internal state and $g_{i,1}$ and $g_{i,2}$ may be equal to 0. Consequently, since the internal state is updated iteratively, the output keystream bit could be described by a Boolean function on the following expressions:

$$\{g_{i,1}(IV) \mid 0 \leq i \leq l-1\} \bigcup \{g_{i,2}(Key) \mid 0 \leq i \leq l-1\}.$$

Accordingly, in cube attacks, when all the non-cube variables are set to constant values, the superpoly p_I of a given cube C_I could be naturally seen as a Boolean function on the expressions in the set

$$G = \{g_{i,2}(Key) \mid 0 \leq i \leq l-1\}.$$

As a result, p_I may be nonlinear on key variables but linear on the expressions in G which is the case we desire. By reasonably classifying the set G, attackers could choose several subsets of G satisfying the surjective condition in Theorem 1.

Finally, recall that the third observation given in Subsection 3.1 points out that Trivium's nonlinear superpolies have fixed forms. In fact, such fixed forms are in accordance with our choosing method, which will be clearly seen in Subsection 2.1. Hence, this method for choosing useful nonlinear expressions to find nonlinear superpolies in cube attacks is very reasonable.

4 Application to Trivium-like Stream Ciphers

In this section, we discuss specific applications of our new framework to cube attacks against Trivium-like ciphers including Trivium, Keryvium, and TriviA-SC-v2. First, we make some notes on implementation details. Then, we list useful nonlinear expressions for the three stream ciphers. Finally, with these useful nonlinear expressions in key variables, we perform several experiments on the three stream ciphers and obtain various results.

4.1 Some Notes

We give some remarks on implementation details about our new technique being used in experimental cube attacks to recover nonlinear superpolies.

First, we suggest to solve the involved systems of nonlinear equations by SAT solvers such as CryptoMiniSat-2.9.5 developed by M. Soos [21]. There are two main reasons for using CryptoMiniSat not Gröbner basis algorithms or other algebraic methods. The first one is that we only need one solution not all solutions for each system of equations. The second one is that CryptoMiniSat is experimentally fast for sparse equations.

Second, recall that in [4], the Moebius transformation was used to search all the subcubes of a large cube to find linear and quadratic superpolies. Our new framework for recover nonlinear superpolies could be combined with the Meobius transformation technique if one has enough memory.

Third, for a stream cipher, useful nonlinear expressions are classified into several groups according to the hypothesis of Theorem 1. Reusing $f(X_1)$ and $f(X_2)$ described in Algorithm 1 for each group test could reduce lots of queries. Besides, when there exists only one set of useful nonlinear expressions, $f(X_1)$ and $f(X_2)$ can be reused to find linear superpolies.

Forth, all our programs were implemented with CUDA and executed on a PC with an Intel(R) Core i7-4790k @4.00GHZ CPU, 32G memory and a GTX-1080 GPU.

4.2 Results for Trivium

Every internal state bit of Trivium is seen as a Boolean function of key and IV variables. By observing the internal states after 91 initialization rounds, we choose the following two sets of nonlinear expressions in Table

3. Actually, they cover all quadratic expressions appearing the internal state after 91 initialization rounds.

Table 3. The chosen nonlinear expressions for Trivium

ciphers	set	chosen nonlinear expressions
Trivium	Set A	$k_{i+25}k_{i+26} \oplus k_{i+27} \oplus k_i (0 \leq i \leq 52)$
	Set B	$k_0k_1 \oplus k_2 \oplus k_{44}$
		$k_ik_{1+i} \oplus k_{2+i} \oplus k_{44+i} \oplus k_{53+i} (1 \leq i \leq 12)$
		$k_ik_{1+i} \oplus k_{2+i} \oplus k_{44+i} (13 \leq i \leq 24)$

To show the correctness and effectiveness of finding nonlinear superpolies using our new framework, we do extensive experiments on the Trivium variants with from 600 to 700 initialization rounds. For each variant, we randomly choose 100 cubes to search linear superpolies and superpolies which are linear about expressions in Set A or B. As a result, we totally obtain 8155985 linear superpolies and 7517944 quadratic superpolies for all these 100 variants. It worth noting that the number of quadratic superpolies is very close to that of linear superpolies. It indicates that quadratic superpolies could be found as easily as linear superpolies with our new framework. Namely, our new framework would make quadratic superpolies play a more important role in cube attacks against Trivium than ever before.

Second, we try our framework for Trivium variants with up to 802 initialization rounds. Some new cubes and superpolies for the 784, 799 and 802-round Tirivum are listed in Table 6 in the Appendix. To the best of our knowledge, for Trivium variants, it is the first time that experimental cube attacks could reach 802 initialization rounds.

4.3 Results for Kreyvium

In the case of Kreyvium, we choose the following set of nonlinear expressions of key variables in Table 4 according to the internal state after 66 initialization rounds. Certainly, there may exist other sets of useful nonlinear expressions.

We first do experiments to search linear supeprolies and superpolies which are linear about the chosen nonlinear expressions of Kreyvium variants with from 600 to 700 initialization rounds. We totally find 1194480 linear superpolies and 2538591 quadratic superpolies for all these 100 variants. Note that the number of quadratic superpolies is more than twice as large as that of linear superpolies. It indicates that quadratic superpolies could be found more easily than linear superpolies with our framework in the case of experimental cube attacks against Kreyvium. Then, we apply our new framework to search linear superpolies and quadratic superpolies for Kreyvium variants with a higher number of initialization rounds. Consequently, for the 776-round Kreyvium, we gain 8 different quadratic superpolies but no linear superpolies based on a cube of size 38, see Table 7 in the Appendix. It further indicates that quadratic superpolies maybe more useful than linear superpolies in the case of round-reduced Kreyvium.

Table 4. The nonlinear expressions chosen for Kreyvium

ciphers	chosen nonlinear expressions	degree
Kreyvium	$k_i \oplus k_{25+i}k_{26+i} \oplus k_{27+i}(0 \leq i \leq 65)$	quadratic

4.4 Results for TriviA-SC-v2

Since TriviA-SC-v2 is the newer version of TriviA-SC, we perform experiments on it with our new framework. According to the internal state of TriviA-SC-v2 after 96 initialization rounds, we choose the following two sets of quadratic expressions in Table 5.

Table 5. The nonlinear expressions chosen for of TriviA-SC-v2

ciphers	set	chosen nonlinear expressions
TriviA-SC-v2	Set A	$k_i \oplus k_{64+i}k_{65+i} \oplus k_{66+i}(0 \leq i \leq 61)$ $k_{62} \oplus k_{126}k_{127}$
	Set B	$k_{35+i} \oplus k_{36+i}k_{37+i} \oplus k_{47+i}(0 \leq i \leq 29)$

First, we search linear superpolies as well as superpolies which are linear about expressions in Set A or B for the TriviA-SC-v2 variants with from 600 to 700 initialization rounds. By choosing 100 cubes randomly for each variant, we obtain various linear superpolies and quadratic superpolies. For all these 100 variants, we have found 4074914 linear superpolies and 491551 quadratic superpolies. It can be seen that the number of quadratic superpolies is non-ignorable. Namely, finding quadratic superpolies with our framework would bring non-ignorable benefits to experimental cube attacks on TriviA-SC-v2. Hence, based on the chosen nonlinear expressions, we attack TriviA-SC-v2 variants which have a higher number of initialization rounds with our new framework. As a result, we find several linear superpolies and quadratic superpolies for the 864-round TriviA-SC-v2 and the 992-round simplified TriviA-SC-v2, see Table 8 in the Appendix.

5 Conclusion

In this paper, we study experimental cube attacks against Trivium-like stream ciphers, and propose a new framework to find nonlinear superpolies. The key idea is doing linearity tests on superpolies about a set of fixed nonlinear key expressions. The complexity of finding nonlinear superpolies in our framework is almost the same as that of finding linear superpolies. Besides, experimental results show that the probability of find nonlinear superpolies using our framework is twice as large as that of finding linear superpolies for Kreyvium and as large as that of finding linear superpolies for Trivium. In fact, we further perform experiments on the 740-round Trivium by choosing 100 cubes of size 30 randomly, and the quadratic superpolies found are twice as many as linear superpolies. Hence, our new framework for finding nonlinear superpolies shall definitely be helpful for cube attacks against Kreyvium and Trivium. Although we only did experiments on quadratic superpolies for Trivium-like stream ciphers, cubic superpolies and superpolies with degree larger than three are also applicable. In such cases, more careful analysis is needed to choose useful key expressions. This will be one subject of our future work. We hope that by gradually increasing the degree of superpolies, the dimension of cubes could be reduced, which is a very important factor of experimental cube attacks. In the future, it is also worthy of working on its applications to other NFSR-based ciphers.

References

1. Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.
2. Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.
3. Piotr Mroczkowski and Janusz Szmidt. Corrigendum to: The cube attack on stream cipher trivium and quadraticity tests. *IACR Cryptology ePrint Archive*, 2011:32, 2011.

4. Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 2013.
5. Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Katz and Shacham [22], pages 250–279.
6. Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting low degree property of superpoly. Cryptology ePrint Archive, Report 2017/1063, 2017. `https://eprint.iacr.org/2017/1063`.
7. Meicheng Liu, Dongdai Lin, and Wenhao Wang. Searching cubes for testing boolean functions and its application to trivium. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 496–500. IEEE, 2015.
8. Santanu Sarkar, Subhamoy Maitra, and Anubhab Baksi. Observing biases in the state: case studies with trivium and trivia-sc. *Des. Codes Cryptography*, 82(1-2):351–375, 2017.
9. Ali Vardasbi, Mahmoud Salmasizadeh, and Javad Mohajeri. Superpoly algebraic normal form monomial test on trivium. *IET Information Security*, 7(3):230–238, 2013.
10. Meicheng Liu. Degree evaluation of nfsr-based cryptosystems. In Katz and Shacham [22], pages 227–249.
11. Yonglin Hao Willi Meier Yosuke Todo, Takanori Isobe. Cube attacks on non-blackbox polynomials based on division property (full version). Cryptology ePrint Archive, Report 2017/306, 2017. `https://eprint.iacr.org/2017/306`.
12. Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of trivium and KATAN. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 200–212. Springer, 2011.
13. Alexander Maximov and Alex Biryukov. Two trivial attacks on trivium. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *Lecture Notes in Computer Science*, pages 36–55. Springer, 2007.
14. Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 313–333. Springer, 2016.
15. Avik Chakraborti, Anupam Chattopadhyay, Muhammad Hassan, and Mridul Nandi. Trivia: A fast and secure authenticated encryption scheme. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 330–353. Springer, 2015.

16. Avik Chakraborti and Mridul Nandi. Trivia-ck-v2. http://competitions.cr.yp.to/round2/triviackv2.pdf (2015).
17. Yuhei Watanabe, Takanori Isobe, and Masakatu Morii. Conditional differential cryptanalysis for kreyvium. In Josef Pieprzyk and Suriadi Suriadi, editors, *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part I*, volume 10342 of *Lecture Notes in Computer Science*, pages 421–434. Springer, 2017.
18. Chao Xu, Bin Zhang, and Dengguo Feng. Linear cryptanalysis of FASER128/256 and trivia-ck. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 237–254. Springer, 2014.
19. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.
20. Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over GF(2). In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques, 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003, Proceedings*, volume 2764 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2003.
21. Mate Soos. Cryptominisat-2.9.5. http://www.msoos.org/cryptominisat2/.
22. Jonathan Katz and Hovav Shacham, editors. *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*. Springer, 2017.

Appendix

The detailed cubes and corresponding superpolies found in our this paper.

Table 6. New superpolies of round-reduced Trivium variants

# of rounds	superpolies	cube index
784	$k_{63}k_{64} \oplus k_{65} \oplus k_{38}$	2,4,6,8,10,12,13,15,19,22,24,28,29,32,34,37,38, 40,41,44,47,49,51,53,55,57,65,68,70,73,74,76,78
	$k_{71}k_{72} \oplus k_{73} \oplus k_{46}$	2,4,6,8,10,12,13,15,19,24,28,29,32,34,37,40,41, 44,47,49,51,53,55,57,59,62,65,70,72,73,74,76,78
	$k_{73}k_{74} \oplus k_{75} \oplus k_{48}$	2,4,6,8,10,12,13,15,19,24,28,29,32,34,37,38,40, 41,44,47,49,51,53,55,57,59,68,70,72,73,74,76,78
799	$k_{71}k_{72} \oplus k_{73} \oplus k_{46}$	0,2,4,5,6,7,9,11,13,14,15,18,20,22,24,26,32,35, 37,39,42,44,48,52,53,55,57,61,62,68,70,74,79
	$k_{27}x_{28} \oplus k_{29} \oplus k_2$	0,2,4,5,6,7,9,11,13,15,18,20,22,24,26,30,32,35, 37,39,42,44,46,52,53,57,62,68,70,72,74,79
802	k_{47}	2,3,4,6,8,10,11,12,15,17,19,21,23,25,29,30,32,34,36, 39,41,43,45,48,50,54,57,58,65,67,69,76,49,59,73,79
	k_{55}	5,7,9,11,13,16,18,20,22,24,26,28,30,31,33,35,37,40, 42,44,46,47,49,51,53,56,60,62,64,66,68,70,74,76,79
	k_{56}	2,4,6,8,10,11,15,17,19,21,23,25,29,30,32,34,36,39, 41,43,45,50,52,54,57,58,67,69,76,49,59,71,73,79
	k_{57}	5,7,9,11,13,16,18,20,22,24,26,28,30,31,33,35,37,40, 42,44,46,49,51,53,55,60,62,64,66,68,70,74,76,79
	k_{59}	5,7,9,11,13,16,18,20,22,24,26,28,30,31,33,35,37,38, 40,42,44,49,51,55,56,60,62,64,66,68,72,74,76,79
	k_{61}	5,7,9,11,13,16,18,20,22,24,26,28,30,31,33,35,37,38, 40,42,46,49,51,53,55,56,60,62,64,66,68,72,74,76,79
	$k_{38}k_{39} \oplus k_{13} \oplus k_{40}$	0,5,7,9,11,13,16,18,20,22,24,26,28,30,31,33,35,37, 40,42,44,46,47,49,51,53,60,62,64,66,72,74,76,79
	$k_{61}k_{62} + k_{36} + k_{63}$	1,2,3,4,6,8,10,12,15,17,19,21,23,25,29,30,32,34,36, 39,41,43,45,50,52,54,57,58,65,67,69,76,49,59,73,79

Table 7. Superpolies of the 776-round Kreyvium

superpolies	cube index
$k_4 \oplus k_{29}k_{30} \oplus k_{31}$	2,5,7,9,13,17,19,22,24,28,30,37,39,41,43,45,52,54,58, 64,69,71,73,77,81,83,87,92,97,103,106,109,117,121
$k_5 \oplus k_{30}k_{31} \oplus k_{32}$	0,2,5,7,9,13,19,22,24,28,30,37,39,41,43,45,52,54,58,66, 69,71,73,77,81,83,87,92,97,103,106,109,117,121,127
$k_6 \oplus k_{31}k_{32} \oplus k_{33}$	0,2,5,7,9,13,17,19,22,24,28,30,37,39,41,43,45,49,52, 54,64,66,69,71,73,77,81,83,97,103,106,117,121,127
$k_{26} \oplus k_{51}k_{52} \oplus k_{53}$	2,5,7,9,13,17,19,22,24,28,30,37,39,41,43,45,49,52,64, 66,69,71,73,77,81,83,87,92,97,103,106,109,117,121
$k_{38} \oplus k_{63}k_{64} \oplus k_{65}$	0,2,5,7,9,13,17,19,22,24,28,30,37,39,41,43,45,49,52, 54,64,66,69,71,73,77,81,83,87,92,97,103,106,117,127
$k_{39} \oplus k_{64}k_{65} \oplus k_{66}$	0,2,5,7,13,17,19,22,24,28,30,37,41,43,45,49,52,54,58, 64,66,71,73,77,81,83,87,92,97,103,106,109,117,121,127
$k_{46} \oplus k_{71}k_{72} \oplus k_{73}$	0,2,5,7,9,13,17,19,22,24,28,30,37,39,41,43,45,52,54, 64,66,69,71,73,77,81,83,87,92,97,103,106,109,117,127
$k_{58} \oplus k_{83}k_{84} \oplus k_{85}$	2,5,7,9,13,17,19,22,24,28,30,37,39,41,43,45,52,54, 58,64,66,69,71,73,77,81,83,87,97,103,109,117,121,127

Table 8. Superpolies of round-reduce TriviA-SC-v2 variants

ciphers	# of rounds	superpolies	cube indexes
TriviA-v2	864	k_1	0,2,8,12,15,18,22,25,30,33,40,47,50,69,72,86,89,92,95,98, 104,111,115,120,127
		k_{20}	0,2,8,12,18,22,25,30,33,40,55,60,66,69,72,86,89,98, 100, 104,111,115,120,127
		k_{21}	0,2,8,15,18,22,25,30,33,40,47,55,66,69,72,86,92,95,98,100, 111,115,120,127
		$k_{22} \oplus 1$	0,2,8,12,18,22,25,30,33,47,50,55,66,69,72,86,89,92,95,98, 104,111,115,120,127
		k_{35}	0,2,8,15,18,22,25,27,30,33,47,55,60,66,69,72,86,89,95,100, 104,111,115,120,127
		k_{37}	0,8,12,15,18,22,27,30,33,47,50,55,66,69,72,86,89,92,98, 100,104,111,115,120,127
		k_{46}	0,2,8,12,15,18,22,30,33,40,44,47,55,60,66,69,72,86,89,92, 98,100,104,111,115,127
		k_{50}	0,2,8,12,15,18,22,25,30,33,40,47,50,60,69,86,89,92,98,100, 104,111,115,120,127
		$k_{52} \oplus 1$	0,2,8,12,18,22,25,30,33,44,47,50,66,69,72,86,89,92,98,100, 104,111,115,120,127
		k_{54}	0,2,8,12,15,18,22,25,30,33,40,50,60,66,69,72,86,89,92,98, 100,104,115,120,127
		k_{56}	0,2,8,12,15,18,22,25,30,33,40,55,66,69,72,86,89,92,100, 104,111,115,120,127
		k_{64}	0,2,8,12,15,18,22,25,30,33,40,50,55,69,72,86,92,95,98,100, 104,111,115,120,127
		$k_{32} \oplus k_{96}k_{97} \oplus k_{98}$	0,2,8,15,22,22,25,30,33,40,44,55,60,66,69,72,86,89,92,100, 111,115,120,127
		$k_{47} \oplus k_{111}k_{112} \oplus k_{113}$	0,2,12,15,18,22,25,30,33,40,47,55,60,69,72,86,89,92,95,98, 100,104,111,115,120,127
		$k_{61} \oplus k_{125}k_{126} \oplus k_{127}$	0,2,8,12,15,22,25,30,33,47,50,55,66,69,72,86,89,92,98,100, 111,115,120
TriviA-v2(simplified)	992	k_2	0,2,5,10,13,16,23,29,34,40,45,49,51,59,66,78,88,90,98,104, 108,110,114,117,119,121,123,125,127
		k_{25}	0,2,5,10,13,19,23,29,34,40,45,49,55,59,66,71,78,85,88,90, 94,98,104,110,112,114,119,121,123,125,
		k_{26}	0,2,5,10,13,16,19,23,29,34,40,45,49,55,59,66,71,78,85,88, 90,94,104,110,112,114,119,121,123,125
		k_{27}	0,2,5,10,13,16,19,23,29,34,40,45,49,55,59,62,71,78,85,90, 94,98,104,108,110,112,114,117,119,121,123,125,
		k_{41}	0,2,10,13,16,19,23,29,40,45,49,55,59,66,71,78,85,88,90,94, 98,104,108,110,112,114,117,121,123,125
		$k_{41} + k_{63}$	0,2,10,13,16,19,23,29,40,45,49,51,55,59,71,78,85,88,90,94, 98,104,108,110,112,114,119,121,123,125
		k_{48}	0,2,5,10,13,16,23,29,40,45,49,51,55,59,66,71,78,88,90,94, 98,104,110,114,117,119,121,123,125
		k_{50}	0,2,5,10,13,16,19,23,29,40,45,49,55,59,66,71,78,85,88,90, 98,104,110,112,114,119,121,123,125,127
		$k_{53} \oplus 1$	0,2,5,10,13,16,19,23,29,34,40,45,49,55,59,66,71,78,85,88, 90,94,98,104,110,112,114,119,121,123
		k_{56}	0,5,10,13,16,19,23,29,40,45,49,55,59,66,71,78,85,88,90, 94,98,104,110,117,119,121,125,127
		k_{57}	2,5,10,13,16,19,23,29,34,40,45,49,55,59,66,71,78,85,88,90, 94,98,104,110,112,114,117,119,121,123,125
		k_{59}	0,2,5,10,13,16,19,23,29,40,45,49,51,55,59,62,66,71,78,85, 94,104,110,112,114,117,119,121,123
		k_{61}	0,2,5,10,13,16,19,23,29,34,40,45,51,55,59,66,71,78,85,90, 94,98,104,108,110,112,114,117,119,121,125
		k_{72}	0,5,10,13,16,19,23,29,40,45,55,59,62,66,71,78,85,90,94, 98,104,110,112,114,117,121,123,125,127
		$k_{33} \oplus k_{97}k_{98} \oplus k_{99}$	0,2,5,10,13,19,23,29,40,45,49,51,55,59,66,71,78,85,88,90, 98,104,108,110,117,119,121,127
		$k_{61} \oplus k_{125}k_{126} \oplus k_{127}$	0,5,10,13,16,19,23,29,34,40,45,49,55,59,62,66,71,78,88, 90,94,98,104,108,110,112,119,121,123,127