The MILP-Aided Conditional Differential Attack and Its Application to Trivium

Chen-Dong Ye, Tian Tian, Fan-Yang Zeng

PLA Strategic Supporting Force Information Engineering University
ye_chendong@126.com,tiantian_d@126.com,fanyang_zeng@126.com

Abstract. Conditional differential attacks were proposed by Knellwolf et al. at ASIACRYPT 2010 which targeted at cryptographic primitives based on non-linear feedback shift registers. The main idea of conditional differential attacks lies in controlling the propagation of a difference through imposing some conditions on public/key variables. In this paper, we improve the conditional differential attack by introducing the mixed integer linear programming (MILP) method to it. Let $J = \{f_i(\boldsymbol{x}, \boldsymbol{v}) = \gamma_i | 1 \leq i \leq N\}$ be a set of conditions that we want to impose, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ (resp. $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$) represents key (resp. public) variables and $\gamma_i \in \{0, 1\}$ needs evaluating. Previous automatic conditional differential attacks evaluate $\gamma_1, \gamma_2, \ldots, \gamma_N$ just in order with the preference to zero. Based on the MILP method, conditions in J could be automatically analysed together. In particular, to enhance the effect of conditional differential attacks, in our MILP models, we are concerned with minimizing the number of 1's in $\{\gamma_1, \gamma_2, \ldots, \gamma_N\}$ and maximizing the number of weak keys.

We apply our method to analyse the security of Trivium. As a result, key-recovery attacks are preformed up to the 978-round Trivium and non-randomness is detected up to the 1108-round Trivium of its 1152 rounds both in the weak-key setting. All the results are the best known so far considering the number of rounds and could be experimentally verified. Hopefully, the new method would provide insights on conditional differential attacks and the security evaluation of Trivium.

Keywords: Conditional Differential Attacks · MILP · Trivium

1 Introduction

Recently non-linear feedback shift registers (NLFSR) are widely used in lightweight cryptographic primitives, for example, the eSTREAM finalist Trivium [3], another eSTREAM finalist Grain-v1 [11] and the block cipher KATAN [2]. Observing that an input difference propagates slowly in an NLFSR, Knellwolf et al. in [12] proposed conditional differential attacks against NLFSR-Based cryptosystems. The main idea of conditional differential attacks is imposing conditions on public/key variables to control the propagation of differences as many rounds as possible. If the propagation of differences is effectively prevented, then key-recovery attacks or distinguishing attacks could be established depending on the conditions. There are three types of conditions. Type 0 conditions only involve public variables. Type 1 conditions involve both public and key variables. Type 2 conditions only involve key variables. Among the three types of conditions, Type 1 conditions are favorable for key-recovery attacks. If there is no type 1 condition, only distinguishing attacks could be mounted. Type 2 conditions imply a class of weak keys.

Conditional differential attacks using the first order difference are mainly applied to Grain-v1. Targeting at Grain-v1, many techniques were proposed to improve conditional differential attacks, including tools to track the difference trails and strategies to choose a good difference to attack more rounds, see [1, 16, 14]. Thus far, utilizing conditional differential attacks based on the first order difference, attackers could do key-recovery attacks for Grain-v1 reduced to 120-round of its 160 rounds [14].

The conditional differential attack was first extended to higher order derivatives and applied to Grain-128 in [12], where the authors established key-recovery and distinguishing attacks for the 213- and 215-round Grain-128 respectively. In [13], the authors proposed automatic tools to find and analyze conditions. As a result, they established some cryptanalytic results on the round reduced Trivium and KATAN. In particular, for Trivium, they obtained a class of weak keys that could be practically distinguished up to 961 rounds. In [26], the authors introduced a method of arrangement of differences and conditions to obtain good higher order conditional differential characteristics. They applied their method to Kreyvium [4], which is a variant of Trivium with 128-bit security. As a result, they obtained a zero-sum distinguisher for the 730-round Kreyvium and detected bias on the 899-round Kreyvium.

In this paper, we focus on conditional differential attacks against Trivium. The most related cryptanalytic results on Trivium are obtained by cube attacks [5]. After the cube attack was proposed, many variants of cube attacks including experimental cube attacks, dynamic cube attacks, correlation cube attacks, and division property based cube attacks were proposed.

In experimental cube attacks [5, 7, 18, 29], superpolies are recovered by experimental tests, i.e., linearity/quadraticity tests. Hence, in experimental cube attacks, the sizes of cubes are typically confined to 40. Thus far, with experimental cube attacks, attackers could do key-recovery attacks for the 802-round Trivium based on cubes with size of $33-36$ [29]. The authors in [28] proposed an algebraic method to recover superpolies in cube attacks. Based on the recovered superpolies, they could recover at least 5 key variables for the 838-round Trivium under a set of about $2^{71.75}$ weak keys. In division property based cube attacks [22–25], due to the power of MILP solvers, large cubes could be explored. As a result, attackers could do key-recovery attacks and distinguishing attacks on the 832- and 839-round Trivium respectively.

Rather than recovering the superpolies to retrieve key variables, in dynamic cube attacks [6] and correlation cube attacks [15], attackers do key-recovery attacks in some other ways. Dynamic cube attacks recover key variables by exploiting distinguishers on superpolies such as unbalanceness and constantness.

Although in [8], the authors proposed dynamic cube attacks against the 721- and 855-round Trivium, quickly the attack against 721-Trivium was experimentally verified to fail and some complexity analysis also indicated that the 855-round attack was questionable in [10]. Correlation cube attacks recover key variables by solving a system of probabilistic equations in key variables derived from conditional correlation properties between superpolies and a set of simple key expressions which is a basis of the superpoly. In [15], a correlation cube attack was applied to the 835-round Trivium which could recover about 5-bit key information.

1.1 Motivation

In this paper, we focus on conditional differential attacks against NLFSR-Based stream ciphers. Let $F(\boldsymbol{x}, \boldsymbol{v})$ be the polynomial representation of the first output bit of a stream cipher, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ represents n key variables and $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ represents m public variables. Choose a set of differences $\boldsymbol{a} = \{a_1, a_2, \ldots, a_d\}$ in $\mathbb{F}_2^{n+m}$. In a single-key attack, differences are only applied to public variables $\boldsymbol{v}$. The derivative of F with respect to $\boldsymbol{a}$ is defined as

$$\Delta_{\boldsymbol{a}} F(\boldsymbol{x}, \boldsymbol{v}) = \oplus_{\boldsymbol{c} \in L(\boldsymbol{a})} F(\boldsymbol{x}, \boldsymbol{v} \oplus \boldsymbol{c}),$$

where $L(\boldsymbol{a})$ denotes the set of 2^d combinations of differences in $\boldsymbol{a}$. Since NLFSR-Based stream ciphers are iterative ciphers and very few state bits are updated in one round, the propagation of the differences in $\boldsymbol{a}$ could be traced. Then in a conditional differential attack, a set of conditions will be obtained to control the propagation of the differences so that the derivative $\Delta_{\boldsymbol{a}} F(\boldsymbol{x}, \boldsymbol{v})$ could be analysed. Let $J = \{f_i(\boldsymbol{x}, \boldsymbol{v}) = \gamma_i | 1 \leq i \leq N\}$, be a set of N conditions in a conditional differential attack where $\gamma_i \in \{0, 1\}$. The values of $(\boldsymbol{x}, \boldsymbol{v})$ satisfying all the conditions in J are called valid inputs of J. The space of valid inputs is closely related to the number of free public variables and the number of weak keys, which are expected to be as large as possible. Generally, $f_i(\boldsymbol{x}, \boldsymbol{v})$ is determined by the internal state update function of a stream cipher, which is relatively fixed. To prevent the propagation of the differences in $\boldsymbol{a}$, it is expected that $\gamma_1, \gamma_2, \ldots, \gamma_N$ to be 0's, but in practice, when $\gamma_1, \gamma_2, \ldots, \gamma_N$ are 0's, maybe there is no input that could satisfy all the conditions in J. Thus, how to choose the values of $\gamma_1, \gamma_2, \ldots, \gamma_N$ is important for successfully mounting a conditional differential attack. Besides, the choices of $\gamma_1, \gamma_2, \ldots, \gamma_N$ also affect the algebraic normal form (ANF) of the derivative $\Delta_{\boldsymbol{a}} F(\boldsymbol{x})$ which is clearly also important in a conditional differential attack.

Previous automatic conditional differential attacks evaluated $\gamma_1, \gamma_2, \ldots, \gamma_N$ in order with the preference to zero [13]. Assume that we have a set of conditions, say $J_0 = \{f_i(\boldsymbol{x}, \boldsymbol{v}) = \gamma_i | 1 \leq i \leq l\}$, and we want to add another condition $f_{l+1}(\boldsymbol{x}, \boldsymbol{v})$ to prevent the propagation of the difference for some internal state bit. Then $f_{l+1}(\boldsymbol{x}, \boldsymbol{v}) = 0$ is added to J_0 if the set of the valid inputs $J_0 \cup \{f_{l+1}(\boldsymbol{x}, \boldsymbol{v}) = 0\}$ is not empty and $f_{l+1}(\boldsymbol{x}, \boldsymbol{v}) = 1$ otherwise. It can be seen that evaluating $\gamma_1, \gamma_2, \ldots, \gamma_N$ in order could not guarantee that the number of 1's attains the minimum. There may be better choice left. Besides, we do not think that the number of 1's is the unique standard for evaluating $\gamma_1, \gamma_2, \ldots, \gamma_N$. The number of rounds, free IV bits and the number of weak keys also should be taken into consideration when we mount an attack. Hence, in the following of this paper, we propose to combine traditional conditional differential attacks with MILP methods, by which we could evaluate conditions more reasonably.

1.2 Our Contribution

MILP solvers are useful tools to solve many optimization problems, and they have been widely used in cryptanalysis such as differential cryptanalysis, linear cryptanalysis, integral cryptanalysis. Inspired by this, we introduce the MILP method into conditional differential attacks to optimize the choices of $\gamma_1, \gamma_2, \ldots,$ γ_N mentioned in Subsection 1.1.

Let $J = \{f_i(\boldsymbol{x}, \boldsymbol{v}) = \gamma_i | 1 \leq i \leq N\}$ be as in Subsection 1.1, where $\gamma_1, \gamma_2, \ldots,$ γ_N need to be determined. We represent every key/IV condition $f_i(\boldsymbol{x}, \boldsymbol{v})$ by linear constraints in an MILP model. In particular, in our MILP model, for each input variable x_i (resp. v_i), we introduce an auxiliary variable $S_{x_i} \in \{0_c, 1_c, \delta\}$ to indicate whether the value of x_i is fixed in the space of valid inputs of J, where $S_{x_i} = 0_c$ or 1_c means that x_i is fixed to be 0 or 1 and $S_{x_i} = \delta$ means x_i is not fixed. Auxiliary variables in our MILP model are useful for optimizing the number of weak keys vulnerable to our attack. We first set the objective function of our model to be the minimal number of 1's in $\gamma_1, \gamma_2, \ldots, \gamma_N$. This aims to prevent the propagation of differences as many rounds as possible. After we obtain the minimum number $\mathcal{B}$ of 1's in $\gamma_1, \gamma_2, \ldots, \gamma_N$, we change the objective function by minimizing the number of fixed input variables but add an upper bound B on the number of 1's with B slightly larger than $\mathcal{B}$ to the MILP model. This aims to enlarge the number of weak keys when the number of 1's in $\gamma_1, \gamma_2, \ldots, \gamma_N$ is small enough. By this way, we could obtain a better tradeoff between controlling the propagation of difference and the space of weak keys.

We apply the new technique to Trivium. As a result, we could do key-recovery attacks up to the **978**-round Trivium and detect non-randomness up to the **1108**-round Trivium both in the weak-key setting. We compare our results with the previous results on Trivium in Table 1. It can be seen that our results are currently the best as far as the number of rounds is concerned.

1.3 Organization

The rest of this paper is organized as follows. Sect. 2 briefly reviews necessary backgrounds. In Sect. 3, we describe the MILP-Aided conditional differential attack in detail. In Sect. 4, we apply our new method to Trivium. Finally, conclusions are drawn in Sect. 5.

Table 1. Summary of results on Trivium

# of rounds	complexity	type of attacks	# of key bits	key size	ref.
968/969/977/978	$2^{26.6}$	key recovery	1	$2^{28.5}$	Sect. 4.3
1108	–	non-randomness	–	$2^{28.5}$	Sect. 4.4
961	2^{25}	distinguishing	–	2^{26}	[13]
802	2^{38}	key recovery	8	all	[29]
832	2^{70}	key recovery	1	all	[22, 23, 25]
838	2^{37}	key recovery	5	$2^{71.75}$	[28]
839	2^{78}	distinguishing	–	all	[25]
855	2^{74}	key recovery	3	all	[8]

2 Preliminaries

2.1 Mixed Integer Linear Programming

The mixed integer linear programming (MILP) is a kind of mathematical optimization with linear constraints and a linear objective, whose all or some of the variables are constrained to be integers. Generally, there are variables, constraints, and an objective function in an MILP model $\mathcal{M}$. In this paper, the variables in $\mathcal{M}$ are denoted by $\mathcal{M}.var$, the constraints in $\mathcal{M}$ are denoted by $\mathcal{M}.con$, and the objective function in $\mathcal{M}$ is denoted by $\mathcal{M}.obj$. If there is no objective function in $\mathcal{M}$, then MILP solvers like Gurobi [9] will return whether $\mathcal{M}$ is feasible. The following is a small example.

Example 1.

$$\begin{aligned}
&\mathcal{M}.var \leftarrow a, b, c \text{ as binary}\\
&\mathcal{M}.con \leftarrow a + c \geq 2\\
&\mathcal{M}.con \leftarrow b + c \geq 1\\
&\mathcal{M}.con \leftarrow a + 2b + c \leq 3\\
&\mathcal{M}.obj \leftarrow \text{minimize } a + b + 2c
\end{aligned}$$

The minimum value of $a + b + 2c$ is 3, where $(a, b, c) = (1, 0, 1)$ is an optimal solution.

The MILP method was first applied to differential and linear cryptanalysis by N. Mouha et al. in [17]. Since then, it has been applied to search characteristics in many cryptanalysis techniques against block ciphers such as differential cryptanalysis [21, 20], impossible differential cryptanalysis [19] and integral cryptanalysis based on the division property [27].

2.2 Conditional Differential Attacks

Let $\mathbb{E}$ be an NFSR-based cipher whose internal state is of size l. Assume that $\mathbb{E}$ is initialized with key variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and public variables $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$. For the sake of convenience, we denote the internal state of $\mathbb{E}$ after t rounds by $s^{(t)} = (s_{t+1}, s_{t+2}, \ldots, s_{t+l})$. Let h be the output function, namely, the output bit z after r rounds is defined as follows:

$$z = h(s_{r+1}, s_{r+2}, \ldots, s_{r+l}).$$

Since $\mathbb{E}$ is initialized with $\boldsymbol{x}$ and $\boldsymbol{v}$, the output bit z can be rewritten as a polynomial on $\boldsymbol{x}, \boldsymbol{v}$, i.e.,

$$z = F(\boldsymbol{x}, \boldsymbol{v}).$$

Let a be difference in $\boldsymbol{v}$. The main idea of conditional differential attacks is to control the propagation of the difference a such that the derived polynomial

$$\Delta_a F(\boldsymbol{x}, \boldsymbol{v}) = F(\boldsymbol{x}, \boldsymbol{v}) \oplus F(\boldsymbol{x}, \boldsymbol{v} \oplus a)$$

can be distinguished from an ideal random polynomial. In order to control the propagation of the difference a, it needs to impose a set of conditions on key/IV variables. For example, when we want to control the propagation of a for the first r rounds, it would lead to a set of conditions

$$\{\Delta_a s_{i+l}(\boldsymbol{x}, \boldsymbol{v}) = \gamma_i | 1 \leq i \leq r\},$$

where $\gamma_i \in \{0, 1\}$. The values of $x_1, \ldots, x_n, v_1, \ldots, v_m$ satisfying all the conditions are called valid inputs. If a bias of $\Delta_a F(\boldsymbol{x}, \boldsymbol{v})$ can be detected, then it would lead to a key-recovery attack or a distinguishing attack depending on whether there is a condition involving both key and IV variables.

2.3 Cube Attacks and Dynamic Cube Attacks

Cube attacks were first proposed by Dinur and Shamir in [5]. Let F be a stream cipher, which is initialized with key variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and public IV variables $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$. Note that the output bit z of F could be represented as a polynomial on $\boldsymbol{x}$ and $\boldsymbol{v}$, i.e. $z = f(\boldsymbol{x}, \boldsymbol{v})$. Let $I = \{i_1, i_2, \ldots, i_d\}$ be a subset of IV indices. Then f can be rewritten as

$$f(\boldsymbol{x}, \boldsymbol{v}) = t_I \cdot p_I(\boldsymbol{x}, \boldsymbol{v}) \oplus q(\boldsymbol{x}, \boldsymbol{v}),$$

where $t_I = \prod_{i \in I} v_i$, p_I does not contain any variable in $\{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$, and each term in q is not divisible by t_I. For each value of the d variables indexed by I, there is a corresponding function being derived from f. Then, the summation of all the 2^d derived functions is equal to p_I, that is,

$$\bigoplus_{(v_{i_1}, v_{i_2}, \ldots, v_{i_d}) \in \mathbb{F}_2^d} f(\boldsymbol{x}, \boldsymbol{v}) = p_I(\boldsymbol{x}, \boldsymbol{v}).$$

In cube attacks, the public variables indexed by I are called *cube variables*, while the remaining public variables are called non-cube variables. The set C_I of all 2^d possible assignments of the cube variables is called a *d-dimensional cube*, and the polynomial p_I is called the *superpoly* of C_I in f. For the sake of convenience, we also call p_I the superpoly of I in f.

A cube attack consists of the preprocessing phase and the online phase. In the preprocessing phase, attackers try to find cubes with low-degree superpolies to obtain low-degree polynomials on key variables. In the online phase, the previously found superpolies are evaluated under the real key. Then, by solving a system of low-degree equations, some key variables could be recovered.

Dynamic Cube Attacks Dynamic cube attacks were first proposed in [6]. The main idea of dynamic cube attacks is to simplify the ANFs of some intermediate state bits by assigning dynamic constraints to public/key variables. As a result, the ANF of the output bit could be simplified and so distinguishers might be obtained. With distinguishers, attackers could do key-recovery attacks or distinguishing attacks depending on whether there is a condition involving both key and IV variables.

2.4 Relationship between Dynamic Cube Attacks and Conditional High Order Differential Attacks

Conditional differential attacks could be extended to higher order derivatives. Let $a_{i_1}, a_{i_2}, \ldots, a_{i_d}$ be d differences. Assume $F(\boldsymbol{x}, \boldsymbol{v})$ is the output function of a stream cipher, where $\boldsymbol{x}$ and $\boldsymbol{v}$ are key variables and IV variables respectively. The d-th order derivative of F with respect to $a_{i_1}, a_{i_2}, \ldots, a_{i_d}$ is defined as

$$\Delta_{a_{i_1},\ldots,a_{i_d}} F(\boldsymbol{x}, \boldsymbol{v}) = \oplus_{\boldsymbol{c} \in L(a_{i_1},\ldots,a_{i_d})} F(\boldsymbol{x}, \boldsymbol{v} \oplus \boldsymbol{c}),$$

where $L(a_{i_1}, \ldots, a_{i_d})$ is the set of all 2^d linear combinations of $a_{i_1}, \ldots, a_{i_d}$. When the Hamming weight of all a_{i_j} is one, it can be seen that $\Delta_{a_{i_1},\ldots,a_{i_d}} F(\boldsymbol{x}, \boldsymbol{v})$ is the superpoly of C_I in F, where $I = \{i_1, i_2, \ldots, i_d\}$.

Originally, in conditional differential attacks, attackers utilize the bias of derivative $\Delta_{a_{i_1},\ldots,a_{i_d}} F(\boldsymbol{x}, \boldsymbol{v})$ to do key-recovery attacks or distinguishing attacks. From the point of view of dynamic cube attacks, the conditions imposed to control the propagation of the chosen differences actually simplify the corresponding superpoly. Therefore, by recovering the ANF of the derivative $\Delta_{a_{i_1},\ldots,a_{i_d}} F(\boldsymbol{x}, \boldsymbol{v})$, i.e. the superpoly, we could do key-recovery attacks even if there is no condition involving both key and IV variables.

3 The MILP-Aided Conditional Differential Attack

Let $F(\boldsymbol{x}, \boldsymbol{v})$ be the polynomial representation of the first output bit of a stream cipher, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ represents n key variables and $\boldsymbol{v} = (v_1, v_2, \ldots,$

v_m) represents m IV variables. Let $\boldsymbol{a} = \{a_1, a_2, \ldots, a_d\}(d \geq 1)$ be a set of differences. Assume

$$J = \{f_i(\boldsymbol{x}, \boldsymbol{v}) = \gamma_i | 1 \leq i \leq N\}$$

is a set of N conditions which we want to impose to control the propagation of $\boldsymbol{a}$, where $(\gamma_1, \gamma_2, \ldots, \gamma_N) \in \mathbb{F}_2^N$ describes the differential characteristic. In this section, we give an MILP-Aided method to determine the values of $\gamma_1, \gamma_2, \ldots, \gamma_N$ in conditional differential attacks.

3.1 The Technique of Auxiliary Variables

In this subsection, we introduce the concept of auxiliary variables and define their operation rules.

Recall that the condition $f_i(\boldsymbol{x}, \boldsymbol{v})$ for $(1 \leq i \leq N)$ is a polynomial in $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$. For each variable $x \in \{x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_m\}$, we add to it an auxiliary variable $S_x \in \{0_c, 1_c, \delta\}$ to indicate the state of x, where 0_c means that x is fixed to 0, 1_c means that x is fixed to 1 and δ means that the value of x is not fixed*. Next, we define operation rules of auxiliary variables for XOR and AND. The rules for XOR are given by

$$\begin{cases} 1_c \oplus 1_c = 0_c, \\ 0_c \oplus y = y, \text{ where } y \in \{0_c, 1_c, \delta\}, \\ \delta \oplus y = \delta, \text{ where } y \in \{0_c, 1_c\}, \\ \delta \oplus \delta \oplus y = 0_c, \text{ where } y \in \{0_c, 1_c, \delta\}. \end{cases}$$

The rules for AND are given by

$$\begin{cases} 1_c \ \& \ y = y, \text{ where } y \in \{0_c, 1_c, \delta\}, \\ 0_c \ \& \ y = 0_c, \text{ where } y \in \{0_c, 1_c, \delta\}, \\ \delta \ \& \ \delta = \delta. \end{cases}$$

Then for a condition $f_i(\boldsymbol{x}, \boldsymbol{v})(1 \leq i \leq N)$, we could determine the state of f_i with the above two rules according to the states of $x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_m$. We offer an illustrative example in the following.

Example 2. Let $f = v_1 \oplus x_2 x_3 \oplus x_4$ be a polynomial on v_1, x_2, x_3, x_4. The auxiliary variables of v_1, x_2, x_3, x_4 are $S_{v_1} = 0_c, S_{x_2} = 1_c, S_{x_3} = \delta, S_{x_4} = 1_c$. Then, $S_f = 0_c \oplus 1_c \ \& \ \delta \oplus 1_c = \delta$, namely, the value of f is not fixed.

Remark 1. For the XOR operation, the result of $\delta \oplus \delta \oplus y$ is defined as 0_c. This rule is rational. For $f(\boldsymbol{x}, \boldsymbol{v})$, it is not necessary to fix each variable to a constant to make $f = 0$. For example, to make $f = v_1 \oplus x_2 = 0$, we do not need to fix the values of v_1 and x_2 respectively but only to require that $v_1 = v_2$ instead. In this case, $S_{v_1} = S_{x_2} = \delta$ and $S_f = 0_c$.

*Naturally, the state of the constant 0/1 is defined as $0_c/1_c$.

MILP Models of Basic Operations In this subsection, we show the MILP models of the above two rules. In our models, for each variable x, we use two binary variables, i.e. A_x and F_x, to describe the state of x. For simplicity, A_x and F_x are called the assignment variable and the flag variable, respectively. With A_x and F_x, we could represent the state of x properly as follows[†]:

– When $F_x = 0$ and $A_x = 0$, it means that x is fixed to 0, i.e., $S_x = 0_c$.
– When $F_x = 0$ and $A_x = 1$, it means that x is fixed to 1, i.e., $S_x = 1_c$.
– When $F_x = 1$ and $A_x = 0$, it means that the value of x is not fixed, i.e., $S_x = \delta$.

In Propositions 1 and 2, we show MILP models for the basic operations AND and XOR.

Proposition 1 (MILP Model for AND). *Assume that $d = a_1 \& a_2 \& \cdots \& a_m$. Let A_{a_i} and F_{a_i} be the assignment variable and flag variable of a_i $(1 \le i \le m)$, respectively. Then, A_d and F_d, the assignment variable and flag variable of d, can be calculated as following*

$$\begin{cases} \mathcal{M}.var \leftarrow A_{a_1}, A_{a_2}, \dots, A_{a_m} \text{ as binary}, \\ \mathcal{M}.var \leftarrow F_{a_1}, F_{a_2}, \dots, F_{a_m} \text{ as binary}, \\ \mathcal{M}.var \leftarrow A_d, F_d, b_{a_1}, b_{a_2}, \dots, b_{a_m}, t_d \text{ as binary}, \\ \mathcal{M}.con \leftarrow b_{a_i} = \max(A_{a_i}, F_{a_i}) \text{ for } i \in \{1, 2, \dots, m\}, \\ \mathcal{M}.con \leftarrow t_d = \max(F_{a_1}, F_{a_2}, \dots, F_{a_m}), \\ \mathcal{M}.con \leftarrow A_d = \min(A_{a_1}, A_{a_2}, \dots, A_{a_m}), \\ \mathcal{M}.con \leftarrow F_d = \min(b_{a_1}, b_{a_2}, \dots, b_{a_m}, t_d). \end{cases}$$

We explain the rationale of the **MILP Model for AND** from the following three cases.

– Case 1: There exists some $j \in \{1, 2, \dots, m\}$ such that $a_j = 0$.
  - In this case, according to the operation rules, we have that $d = 0$, i.e. $S_d = 0_c$. Since $a_j = 0$, we have that $F_{a_j} = 0$ and $A_{a_j} = 0$. Hence, $b_{a_j} = 0$ and so $A_d = 0$ and $F_d = 0$. Namely, the constraints added to the model guarantee that $d = 0$.
– Case 2: $a_i = 1$ for each $1 \le i \le m$.
  - In this case, according to the operation rules, we have that $d = 1$. Since $a_i = 1$ for $1 \le i \le m$, we have that $A_{a_i} = 1$ and $F_{a_i} = 0$ for $1 \le i \le m$. It can be deduced that $b_{a_i} = 1$ for $1 \le i \le m$ and $t_d = 0$. Hence, we have that $F_d = 0$ and $A_d = 1$. Namely, the constraints added to the model guarantee that d is equal to 1.
– Case 3: $a_i \neq 0$ for $1 \le i \le m$ and there exists some $j \in \{1, 2, \dots, m\}$ such that a_j is not fixed.

[†]Since there are only three states of the variable x, we add a constraint $A_x \le 1 - F_x$ to discard the case of $F_x = 1$ and $A_x = 1$.

  - In this case, according to the operation rules, we have that d is not fixed. Since $a_i \neq 0$ for $1 \leq i \leq m$, we have that $A_{a_i} + F_{a_i} = 1$. Furthermore, it can be deduced that $A_{a_j} = 0$ and $F_{a_j} = 1$ for a_j is not fixed. Then, we have that $b_{a_i} = 1$ for each $1 \leq i \leq m$ and $t_d = 1$. Hence, we know that $A_d = 0$ and $F_d = 1$. Namely, the constraints added to the model guarantee that d is not fixed.

According to the above illustrations, we know that the MILP model for AND built above is in accordance with the rules of auxiliary variables for AND. In the remainder of this paper, the above procedure is denoted by

$$(\mathcal{M}, A_d, F_d) \leftarrow AFAND(\mathcal{M}, A_{a_1}, A_{a_2}, \ldots, A_{a_m}, F_{a_1}, F_{a_2}, \ldots, F_{a_m})$$

for simplicity.

Proposition 2 (MILP Model for XOR). *Assume that $d = a_1 \oplus a_2 \oplus \cdots \oplus a_m \oplus \alpha$, where α is a constant belonging to $\{0,1\}$. Let A_{a_i} and F_{a_i} be the assignment variable and flag variable of a_i ($1 \leq i \leq m$), respectively. Then, A_d and F_d, the assignment variable and flag variable of d, can be calculated as following*

$$\begin{cases} \mathcal{M}.var \leftarrow A_{a_1}, A_{a_2}, \ldots, A_{a_m}, T_1, \ldots, T_m, b_d \text{ as binary }, \\ \mathcal{M}.var \leftarrow F_{a_1}, F_{a_2}, \ldots, F_{a_m}, \beta_d \text{ as binary }, \\ \mathcal{M}.con \leftarrow \beta_d = \bigoplus_{i=1}^{m} A_{a_i} \oplus \alpha^{\ddagger}, \\ \mathcal{M}.con \leftarrow b_d = \max(F_{a_1}, F_{a_2}, \ldots, F_{a_m}), \\ \mathcal{M}.con \leftarrow T_j = \min(\widetilde{F}_{a_1}, \ldots, \widetilde{F}_{a_{j-1}}, F_{a_j}, \widetilde{F}_{a_{j+1}}, \ldots, \widetilde{F}_{a_m}) \text{ for } 1 \leq j \leq m, \\ \mathcal{M}.con \leftarrow F_d = T_1 + T_2 + \cdots + T_m, \\ \mathcal{M}.con \leftarrow A_d = \min(1 - F_d, \beta_d, 1 - b_d), \end{cases}$$

where $\widetilde{F}_{a_j} = 1 - F_{a_j}$ for $1 \leq j \leq m$. We explain the rationale of the **MILP Model for XOR** from the following three cases.

- Case 1: For $1 \leq i \leq m$, x_i is fixed to 0 or 1.
  - In this case, according to the operation rules, we have that $d = \bigoplus_{i=1}^{m} a_i \oplus \alpha$. Since a_i is fixed to 0 or 1, we have $F_{a_i} = 0$ for $1 \leq i \leq m$. Hence, $b_d = 0$ and $T_j = 0$ for $1 \leq j \leq m$. Then, it can be deduced that $F_d = 0$ and so $A_d = \bigoplus_{i=1}^{m} A_{a_i} \oplus \alpha$. Namely, the constraints added to the model guarantee that $d = \bigoplus_{i=1}^{m} a_i \oplus \alpha$.
- Case 2: There is exactly one $j \in \{1, 2, \ldots, m\}$ such that a_j is not fixed, while the remaining variables are fixed to 0 or 1.
  - In this case, according to the operation rules, we have that d is not fixed. Since a_j is not fixed, $F_{a_j} = 1$. Then, we have $T_j = 1$ and $T_i = 0$ for $i \neq j$. Hence, $F_d = 1$ and so $A_d = 0$. Namely, the constraints added to the model guarantee that d is not fixed.

[‡]In MILP models, we could not add the constraint $\beta_d = \bigoplus_{i=1}^{m} A_{a_i} \oplus \alpha$ directly. We show how to describe $\beta_d = \bigoplus_{i=1}^{m} A_{a_i} \oplus \alpha$ with linear constraints in Appendix.

– Case 3: There are at least two variables are not fixed.
  - In this case, according to the operation rules, we have that d could be fixed to 0, i.e. $S_d = 0_c$. For simplicity, we assume that a_1 and a_2 are not fixed. Since a_1 and a_2 are not fixed, we have that $F_{a_1} = F_{a_2} = 1$. Then, $T_j = 0$ for $1 \leq j \leq m$, and so $F_d = 0$. Since $b_d = \max(F_{a_1}, F_{a_2}, \ldots, F_{a_m}) = 1$, we have that $A_d = 0$. Namely, the constraints added to the model guarantee that d could be fixed to 0.

According to the above illustrations, we know that the MILP model for XOR built above is in accordance with the rules of auxiliary variables for XOR. In the remainder of this paper, the above process is denoted by

$$(\mathcal{M}, A_d, F_d) \leftarrow AFXOR(\mathcal{M}, A_{a_1}, A_{a_2}, \ldots, A_{a_m}, F_{a_1}, F_{a_2}, \ldots, F_{a_m}, \alpha)$$

for simplicity.

3.2 MILP Models to Determine Proper Differential Characteristics

In this subsection, we show how to build MILP models to determine a proper differential characteristic, i.e. the values of $\gamma_1, \gamma_2, \ldots, \gamma_N$. First, we introduce how to build MILP models to determine the state of a polynomial $f(\boldsymbol{x}, \boldsymbol{v})$ according to the states of $x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_m$. Then, we show how to build MILP models which could determine a proper value of $\gamma_1, \gamma_2, \ldots, \gamma_N$ with the consideration of the space of valid inputs.

MILP Models to Determine the State of A Condition Let $f(\boldsymbol{x}, \boldsymbol{v}) = \bigoplus_{1 \leq j \leq L} (\boldsymbol{x}||\boldsymbol{v})^{\boldsymbol{w}_j}$ be a polynomial of size L, where $\boldsymbol{w}_j = (w_j^1, w_j^2, \ldots, w_j^{m+n}) \in \mathbb{F}_2^{m+n}$ and $(\boldsymbol{x}||\boldsymbol{v})^{\boldsymbol{w}_j} = \prod_{i=1}^{n} x_i^{w_j^i} \cdot \prod_{i=1}^{m} v_i^{w_j^{n+i}}$. Based on Propositions 1 and 2, we could determine the states of f according to the state of $x_1, x_2, \ldots, x_n, v_1, v_2 \ldots, v_m$. For an MILP model $\mathcal{M}$ which contains A_x and F_x for each variable $x \in \{x_1, x_2, \ldots, x_n, v_1, v_2 \ldots, v_m\}$, Algorithm 1 shows how to build a new MILP model which determines the state of f. In Algorithm 1, we first linearize the polynomial f as $f = \bigoplus_{1 \leq j \leq L} y_j$, where $y_j = (\boldsymbol{x}||\boldsymbol{v})^{\boldsymbol{w}_j}$. Then, in Line 6 of Algorithm 1, we determine the state of y_j, i.e. A_{y_j} and F_{y_j}, by applying the procedure $AFAND$. Finally, by applying the procedure $AFXOR$ directly, the state of f can be determined according to the states of $y_1, y_2, \ldots, y_L$.

MILP Models to Determine A Proper Differential Characteristic In this subsection, we show how to build an MILP model to determine a proper differential characteristic. Algorithm 2 describes the detailed procedure. In Algorithm 2, for each condition $f_i (1 \leq i \leq N)$, we determine the state of f_i using Algorithm 1. Then, we add constraints to make requirements on the number of conditions which are fixed to 1. Finally, maximizing $\sum_{x \in \mathbb{X}} F_x$ is set as the objective of our model, where $\mathbb{X} = \{x_1, \ldots, x_n, v_1, \ldots, v_m\}$. In the following, we make some explanations of Algorithm 2.

Algorithm 1 Determine the state of a condition

```
1: procedure DetStaofCond($\mathcal{M}$,$f$)
2:     Let $T$ be the set of terms of $f$;
3:     Set $j = 1$;
4:     for each term $u \neq 1$ in $T$ do
5:         Let $x_{i_1}, x_{i_2}, \ldots, x_{i_d}, v_{l_1}, v_{l_2}, \ldots, v_{l_q}$ be the variables appearing in $u$;
6:         Set $\boldsymbol{A_u} = (A_{x_{i_1}}, \ldots, A_{x_{i_d}}, A_{v_{l_1}}, \ldots, A_{v_{l_q}})$;
7:         Set $\boldsymbol{F_u} = (F_{x_{i_1}}, \ldots, F_{x_{i_d}}, F_{x_{l_1}}, \ldots, F_{v_{l_q}})$;
8:         $(\mathcal{M}, A_y, F_y) \leftarrow AFAND(\mathcal{M}, \boldsymbol{A_u}, \boldsymbol{F_u})$
9:         Set $j = j + 1$;
10:    end for
11:    $(\mathcal{M}, A_f, F_f) \leftarrow AFXOR(\mathcal{M}, A_{y_1}, A_{y_2}, \ldots, A_{y_{j-1}}, F_{y_1}, F_{y_2}, \ldots, F_{y_{j-1}}, a)$, where
       $a$ is the constant term of $f$;
12:    return $(\mathcal{M}, F_f, A_f)$;
13: end procedure
```

- The parameter D in Line 6 of Algorithm 2 is a positive integer, which is much larger than N. If there exists some $j \in \{1, 2, \ldots, N\}$ such that $F_{f_j} = 1$, i.e. the value of f_j is not fixed, then we have that $Xsum < 0$. Namely, the constraint $Xsum \geq 0$ guarantees that the value of each condition is fixed to 0 or 1.
- The parameter B, which is set by the attackers, is used to bound the number of conditions which are fixed to 1. To determine a proper value of B, minimizing the value of $Xsum$ under the constraint $Xsum \geq 0$ is set as the objective of the model. Therefore, the attackers can obtain an lower bound of B by solving the model, and so the attackers could set a proper value of B accordingly.
- In conditional differential attacks, less variables which are fixed to constants usually means a larger space of valid inputs. Hence, for the given bound B, $\sum_{x \in \mathbb{X}} F_x$ is set as the objective of the model so that we could make the number of the variables $x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_m$ which are not fixed as large as possible.

Remark 2. In the rules of auxiliary variables for XOR, the result of $\delta \oplus \delta \oplus y$ is defined as 0_c, where $y \in \{0_c, 1_c, \delta\}$. It may lead to a contradiction. We show an example in the following.

Example 3. Let $f_1 = x_1 \oplus x_2$ and $f_2 = x_1 \oplus x_2 \oplus 1$ be two polynomials. When $S_{x_1} = S_{x_2} = \delta$, we have that $S_{f_1} = 0_c$ and $S_{f_2} = 0_c$ according to the operation rules of auxiliary variables for XOR. This implies that f_1 and f_2 could be fixed to 0 at the same time. However, since $f_1 = f_2 \oplus 1$, f_1 and f_2 would never be fixed to 0 at the same time.

Although our model can not avoid the similar case in Example 3, we could remedy it by taking an extra step. Let c_i be the value of $f_i(\boldsymbol{x}, \boldsymbol{v})$ determined by

Algorithm 2 Determine a proper differential characteristic

```
Input: The set $J' = \{f_1, f_2, \ldots, f_N\}$
1: Declare an empty MILP model $\mathcal{M}$;
2: Declare $A_{x_1}, \ldots, A_{x_n}, A_{v_1}, \ldots, A_{v_m}, F_{x_1}, \ldots, F_{x_n}, F_{v_1}, \ldots, F_{v_m}$ as the assignment
   and flag variables of $x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_m$;
3: for $f \in J'$ do
4:     $(\mathcal{M}, A_f, F_f) = DetStaofCond(\mathcal{M}, f)$
5: end for
6: $\mathcal{M}.con \leftarrow Xsum = (\sum_{f \in J'} A_f - D \times F_f)$;
7: $\mathcal{M}.con \leftarrow Xsum \geq 0$;
8: $\mathcal{M}.con \leftarrow Xsum \leq B$;
9: $\mathcal{M}.obj \leftarrow$ maximum $\sum_{x \in \mathbb{X}} F_x$, where $\mathbb{X} = \{x_1, x_2, \cdots, x_n, v_1, v_2, \cdots, v_m\}$;
```

our model for $1 \leq i \leq N$. Then, to check whether there is a contradiction, we only need to solve the following system of equations

$$\begin{cases} f_1(\boldsymbol{x}, \boldsymbol{v}) = c_1, \\ \qquad \vdots \\ f_N(\boldsymbol{x}, \boldsymbol{v}) = c_N. \end{cases}$$

If this system of equations is unsolvable, then we know that there are contradictions happening. In this case, we only need to remove this solution from the model and search for another solution. We could repeat the above procedure until we find a valid solution.

4 Application to Trivium

In this section, we apply the method to analyse the security of Trivium.

4.1 Specification of Trivium

Trivium is a bit oriented synchronous stream cipher which was selected as one of eSTREAM hardware-oriented portfolio ciphers. The main building block of Trivium is a 288-bit Galois nonlinear feedback shift register. For every clock cycle there are three bits of the internal state updated by a feedback function and all the remaining bits of the internal sate are updated by shifting. The internal state of Trivium, denoted by $(s_1, s_2, \ldots, s_{288})$, is initialized by loading an 80-bit secret key and an 80-bit IV into the registers, and setting all the remaining bits to 0 except for the last three bits of the third register. Then, after updating the internal state 1152 rounds, the algorithm would generate the output keystream bits. Algorithm 3 describes the pseudo-code of Trivium. For more details, please refer to [3].

4.2 MILP Models for Trivium

When applying our method to Trivium, similar to [13], we choose the difference of Hamming weight one which are applied to IV variables only. Let $\boldsymbol{a} = \{a_{i_1}, a_{i_2}, \ldots, a_{i_d}\}$ be the chosen differences. Denote $I = \{i_1, i_2, \ldots, i_d\}$ by the set of indices of the chosen differences. For each single difference a_{i_j} $(1 \leq j \leq d)$, we try to control the propagation of a_{i_j} for the first r rounds. In Algorithm 4, for the i-th round $(1 \leq i \leq r)$, we first collect the conditions derived from the three updated bits s^i_1, s^i_{94} and s^i_{178}, where we use the software Singular to calculate the ANFs of s^i_1, s^i_{94} and s^i_{178}. Then, for the key variables $x_1, x_2, \ldots, x_{80}$ and the IV variables $v_1, v_2, \ldots, v_{80}$, we add the corresponding assignment and flag variables, i.e. $A_{x_1}, F_{x_1}, \ldots, A_{x_{80}}, F_{x_{80}}, A_{v_1}, F_{v_1}, \ldots, A_{v_{80}}, F_{v_{80}}$, to the model. Besides, since a_i $(i \in I)$ is a chosen difference, the value of v_i could not be fixed to 0 or 1. Hence, in Line 21-24 of Algorithm 4, we add constraints to make sure that the value of v_i is not fixed.

4.3 Key-Recovery Attacks on the Round-Reduced Trivium

With respect to the structure of Trivium, similar to [13], we choose the differences at a distance of three. First, we perform experiments for $I_1 = \{0, 3, 6, \ldots, 69\}$, namely, $\{e_0, e_3, \ldots, e_{69}\}$ are the chosen differences.

In our experiments[§], we attempt to control the propagation of each difference e_i $(i \in I_1)$ for the first 200 rounds. As a result, we obtain a set $J_1 = \{f_1, f_2, \ldots, f_{652}\}$ of 652 conditions. Denote by $NC1$ the order of the set $\{f = 1 | f \in J_1\}$. Then, by solving MILP models, we obtain that $NC_1 \geq 3$ under the condition that each $f \in J_1$ is fixed to 0 or 1. Hence, we set the parameter $B = 3$ and set maximizing the number of key variables which are not fixed (denoted

[§]We use the MILP solver Gurobi to solve the generated MILP models. Besides, all our experiments are performed on a PC with an i7-7700K CPU and 32G RAM.

Algorithm 3 Pseudo-code of Trivium

```
1: $(s_1, s_2, \ldots, s_{93}) \leftarrow (x_1, x_2, \ldots, x_{80}, 0, \ldots, 0)$;
2: $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (v_1, v_2, \ldots, v_{80}, 0, \ldots, 0)$;
3: $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$;
4: for $i$ from 1 to $N$ do
5:     $t_1 \leftarrow s_{66} \oplus s_{93} \oplus s_{91}s_{92} \oplus s_{171}$;
6:     $t_2 \leftarrow s_{162} \oplus s_{177} \oplus s_{175}s_{176} \oplus s_{264}$;
7:     $t_3 \leftarrow s_{243} \oplus s_{288} \oplus s_{286}s_{287} \oplus s_{69}$;
8:     if $i > 1152$ then
9:         $z_{i-1152} \leftarrow s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288}$;
10:    end if
11:    $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$;
12:    $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$;
13:    $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$;
14: end for
```

by NA_x) as the objective of the model. By solving the corresponding MILP model, which could be solved with Gurobi in seconds under a PC, we obtain the following condition:

Algorithm 4 Determine proper differential characteristics for Trivium

```
Input: The set of indices of chosen differences: $I$, The number of difference-controlled
    rounds: $r$
 1: Set $J = \emptyset$;
 2: for $i \in I$ do
 3:     for $j$ from 1 to $r$ do
 4:         $f \leftarrow \Delta_{e_i} s_1^j(\boldsymbol{x}, \boldsymbol{v})$;
 5:         if $f \neq 1$ and $f \neq 0$ then
 6:             $J \leftarrow J \cup f$
 7:         end if
 8:         $f \leftarrow \Delta_{e_i} s_{94}^j(\boldsymbol{x}, \boldsymbol{v})$;
 9:         if $f \neq 1$ and $f \neq 0$ then
10:             $J \leftarrow J \cup f$
11:         end if
12:         $f \leftarrow \Delta_{e_i} s_{178}^j(\boldsymbol{x}, \boldsymbol{v})$;
13:         if $f \neq 1$ and $f \neq 0$ then
14:             $J \leftarrow J \cup f$
15:         end if
16:     end for
17: end for
18: Declare an empty MILP model $\mathcal{M}$;
19: Declare $A_{x_1}, \dots, A_{x_{80}}, F_{x_1}, \dots, F_{x_{80}}$ as 160 MILP variables of $\mathcal{M}$ corresponding to
    the assignment and flag variables of secret variables $x_1, x_2, \dots, x_{80}$;
20: Declare $A_{v_1}, \dots, A_{v_{80}}, F_{v_1}, \dots, F_{v_{80}}$ as 160 MILP variables of $\mathcal{M}$ corresponding to
    the assignment and flag variables of IV variables $v_1, v_2, \dots, v_{80}$;
21: for $i \in I$ do
22:     $\mathcal{M}.con \leftarrow A_{v_i} = 0$;
23:     $\mathcal{M}.con \leftarrow F_{v_i} = 1$;
24: end for
25: for $f \in J$ do
26:     $(\mathcal{M}, A_f, F_f) = DetStaofCond(\mathcal{M}, f)$
27: end for
28: $\mathcal{M}.con \leftarrow Xsum = \sum_{f \in J}(A_f - D \times F_f)$;
29: $\mathcal{M}.con \leftarrow Xsum \geq 0$;
30: $\mathcal{M}.con \leftarrow Xsum \leq B$;
31: $\mathcal{M}.obj \leftarrow$ maximum $\sum_{f \in J} V_f$;
```

$$\begin{aligned}
C_1:\ &(v_1,v_2,v_4,v_5,v_7,v_8,v_{10},v_{11},v_{13},v_{14},v_{16},v_{17},v_{19},v_{20},\\
&v_{22},v_{23},v_{25},v_{26},v_{28},v_{29},v_{31},v_{32},v_{34},v_{35},v_{37},v_{38},v_{40},\\
&v_{41},v_{43},v_{44},v_{46},v_{47},v_{49},v_{50},v_{52},v_{53},v_{55},v_{56},v_{58},v_{59},\\
&v_{61},v_{62},v_{64},v_{65},v_{67},v_{68},v_{70},v_{71},v_{73},v_{74},v_{76},v_{77},v_{79})=\mathbf{0}_{53};\\
&(k_1,k_2,k_4,k_5,k_7,k_8,k_{10},k_{11},k_{13},k_{14},k_{16},k_{17},k_{19},k_{20},\\
&k_{22},k_{23},k_{25},k_{26},k_{28},k_{29},k_{31},k_{32},k_{34},k_{35},k_{37},k_{38},k_{40},\\
&k_{41},k_{43},k_{44},k_{46},k_{47},k_{49},k_{50},k_{52},k_{53},k_{55},k_{56},k_{58},k_{59},\\
&k_{61},k_{62},k_{64},k_{65},k_{67},k_{68},k_{70},k_{71},k_{73},k_{74},k_{76},k_{77},k_{79})=\mathbf{0}_{53};\\
&k_{66}=1;
\end{aligned}$$

Furthermore, we try to increase B to 4 and set maximizing NA_x as the objective of the model. As a result, we obtain several conditions, and we list two of them as follows:

$$\begin{aligned}
C_2:\ &(v_1,v_2,v_4,v_5,v_7,v_8,v_{10},v_{11},v_{13},v_{14},v_{16},v_{17},v_{19},v_{20},\\
&v_{22},v_{23},v_{25},v_{26},v_{28},v_{29},v_{31},v_{32},v_{34},v_{35},v_{37},v_{38},v_{40},\\
&v_{41},v_{43},v_{44},v_{46},v_{47},v_{49},v_{50},v_{52},v_{53},v_{55},v_{56},v_{58},v_{59},\\
&v_{61},v_{62},v_{64},v_{65},v_{67},v_{68},v_{70},v_{71},v_{73},v_{74},v_{76},v_{77},v_{79})=\mathbf{0}_{53};\\
&(k_1,k_2,k_4,k_5,k_7,k_8,k_{10},k_{11},k_{13},k_{14},k_{16},k_{17},k_{19},k_{20},\\
&k_{22},k_{23},k_{25},k_{26},k_{28},k_{29},k_{31},k_{32},k_{34},k_{35},k_{37},k_{38},k_{40},\\
&k_{41},k_{43},k_{44},k_{46},k_{47},k_{49},k_{50},k_{52},k_{53},k_{55},k_{56},k_{58},k_{59},\\
&k_{61},k_{62},k_{64},k_{65},k_{66},k_{68},k_{70},k_{71},k_{73},k_{74},k_{76},k_{77},k_{79})=\mathbf{0}_{53};\\
&k_{67}=1;\\
C_3:\ &(v_1,v_2,v_4,v_5,v_7,v_8,v_{10},v_{11},v_{13},v_{14},v_{16},v_{17},v_{19},v_{20},\\
&v_{22},v_{23},v_{25},v_{26},v_{28},v_{29},v_{31},v_{32},v_{34},v_{35},v_{37},v_{38},v_{40},\\
&v_{41},v_{43},v_{44},v_{46},v_{47},v_{49},v_{50},v_{52},v_{53},v_{55},v_{56},v_{58},v_{59},\\
&v_{61},v_{62},v_{64},v_{65},v_{67},v_{68},v_{70},v_{71},v_{73},v_{74},v_{76},v_{77},v_{79})=\mathbf{0}_{53};\\
&(k_1,k_2,k_4,k_5,k_7,k_8,k_{10},k_{11},k_{13},k_{14},k_{16},k_{17},k_{19},k_{20},\\
&k_{22},k_{23},k_{25},k_{26},k_{28},k_{29},k_{31},k_{32},k_{34},k_{35},k_{37},k_{38},k_{40},\\
&k_{41},k_{43},k_{44},k_{46},k_{47},k_{49},k_{50},k_{52},k_{53},k_{55},k_{56},k_{58},k_{59},\\
&k_{61},k_{62},k_{64},k_{65},k_{66},k_{67},k_{68},k_{70},k_{71},k_{73},k_{74},k_{76},k_{77},k_{79})=\mathbf{0}_{54};
\end{aligned}$$

To obtain further results, we slide the differences chosen above, i.e. $\{e_1, e_4, \dots, e_{70}\}$ are the chosen differences and $I_2 = \{1, 4, 7, \dots, 70\}$. Then, we perform similar experiments as above. As a result, we obtain three different conditions C_4, C_5 and C_6 given by

¶The condition C_2 is the same as the condition derived in [13].

$$
\begin{aligned}
C_4:\ & (v_0, v_2, v_3, v_5, v_6, v_8, v_9, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{18}, v_{20},\\
& v_{21}, v_{23}, v_{24}, v_{26}, v_{27}, v_{29}, v_{30}, v_{32}, v_{33}, v_{35}, v_{36}, v_{38}, v_{39},\\
& v_{41}, v_{42}, v_{44}, v_{45}, v_{47}, v_{48}, v_{50}, v_{51}, v_{53}, v_{54}, v_{56}, v_{57}, v_{59},\\
& v_{60}, v_{62}, v_{63}, v_{65}, v_{66}, v_{68}, v_{69}, v_{71}, v_{72}, v_{74}, v_{75}, v_{77}, v_{78}) = \mathbf{0}_{53};\\
& (k_0, k_2, k_3, k_5, k_6, k_8, k_9, k_{11}, k_{12}, k_{14}, k_{15}, k_{17}, k_{18}, k_{20},\\
& k_{21}, k_{23}, k_{24}, k_{26}, k_{27}, k_{29}, k_{30}, k_{32}, k_{33}, k_{35}, k_{36}, k_{38}, k_{39},\\
& k_{41}, k_{42}, k_{44}, k_{45}, k_{47}, k_{48}, k_{50}, k_{51}, k_{53}, k_{54}, k_{56}, k_{57}, k_{59},\\
& k_{60}, k_{62}, k_{63}, k_{65}, k_{66}, k_{68}, k_{69}, k_{71}, k_{72}, k_{74}, k_{75}, k_{77}, k_{78}) = \mathbf{0}_{53};\\
& k_{67} = 1;\\
C_5:\ & (v_0, v_2, v_3, v_5, v_6, v_8, v_9, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{18}, v_{20},\\
& v_{21}, v_{23}, v_{24}, v_{26}, v_{27}, v_{29}, v_{30}, v_{32}, v_{33}, v_{35}, v_{36}, v_{38}, v_{39},\\
& v_{41}, v_{42}, v_{44}, v_{45}, v_{47}, v_{48}, v_{50}, v_{51}, v_{53}, v_{54}, v_{56}, v_{57}, v_{59},\\
& v_{60}, v_{62}, v_{63}, v_{65}, v_{66}, v_{68}, v_{69}, v_{71}, v_{72}, v_{74}, v_{75}, v_{77}, v_{78}) = \mathbf{0}_{53};\\
& (k_0, k_2, k_3, k_5, k_6, k_8, k_9, k_{11}, k_{12}, k_{14}, k_{15}, k_{17}, k_{18}, k_{20},\\
& k_{21}, k_{23}, k_{24}, k_{26}, k_{27}, k_{29}, k_{30}, k_{32}, k_{33}, k_{35}, k_{36}, k_{38}, k_{39},\\
& k_{41}, k_{42}, k_{44}, k_{45}, k_{47}, k_{48}, k_{50}, k_{51}, k_{53}, k_{54}, k_{56}, k_{57}, k_{59},\\
& k_{60}, k_{62}, k_{63}, k_{65}, k_{67}, k_{68}, k_{69}, k_{71}, k_{72}, k_{74}, k_{75}, k_{77}, k_{78}) = \mathbf{0}_{53};\\
& k_{66} = 1;\\
C_6:\ & (v_0, v_2, v_3, v_5, v_6, v_8, v_9, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{18}, v_{20},\\
& v_{21}, v_{23}, v_{24}, v_{26}, v_{27}, v_{29}, v_{30}, v_{32}, v_{33}, v_{35}, v_{36}, v_{38}, v_{39},\\
& v_{41}, v_{42}, v_{44}, v_{45}, v_{47}, v_{48}, v_{50}, v_{51}, v_{53}, v_{54}, v_{56}, v_{57}, v_{59},\\
& v_{60}, v_{62}, v_{63}, v_{65}, v_{66}, v_{68}, v_{69}, v_{71}, v_{72}, v_{74}, v_{75}, v_{77}, v_{78}) = \mathbf{0}_{53};\\
& (k_0, k_2, k_3, k_5, k_6, k_8, k_9, k_{11}, k_{12}, k_{14}, k_{15}, k_{17}, k_{18}, k_{20},\\
& k_{21}, k_{23}, k_{24}, k_{26}, k_{27}, k_{29}, k_{30}, k_{32}, k_{33}, k_{35}, k_{36}, k_{38}, k_{39},\\
& k_{41}, k_{42}, k_{44}, k_{45}, k_{47}, k_{48}, k_{50}, k_{51}, k_{53}, k_{54}, k_{56}, k_{57}, k_{59},\\
& k_{60}, k_{62}, k_{63}, k_{65}, k_{66}, k_{67}, k_{68}, k_{69}, k_{71}, k_{72}, k_{74}, k_{75}, k_{77}, k_{78}) = \mathbf{0}_{54};
\end{aligned}
$$

For I_1 (resp. I_2), we recover the derivatives for the output bits of some Trivium variants with no less than 960 rounds, i.e. the superpolies, of the chosen differences under conditions C_1, C_2 and C_3 (resp. C_4, C_5 and C_6). We summarise our results in Table 2[‖]. According to Table 2, we have $6 \times 2^{26} \approx 2^{28.6}$ weak keys totally. For each class of weak keys with 2^{26} elements, we could recover one key bit with a complexity of 2^{24} for the 977-/968-/978-/969-round Trivium respectively. Therefore, for a weak key in each class of weak keys, we could recover it with a complexity of $2^{25} + 2^{24}$.

[‖] We put the detailed superpolies in the supporting materials.

Table 2. The derivatives of I_1 and I_2 under different conditions

Differences	condition	B	# of weak keys	# of rounds	degree of derivatives
I_1	C_1	3	2^{26}	968	1
	C_2	4	2^{26}	960	3
				961	2
				962	2
				977	2
	C_3	4	2^{26}	968	2
I_2	C_4	3	2^{26}	969	1
	C_5	4	2^{26}	960	2
				961	1
				978	3
	C_6	4	2^{26}	969	2

4.4 Non-Randomness of the Round Reduced Trivium

In the above subsection, for $I_1 = \{0, 3, 6, \ldots, 69\}$, the IV variables v_{72}, v_{75} and v_{78} are not involved in the conditions C_1, C_2 and C_3. Hence, under the conditions C_1, C_2 and C_3, we evaluate the degrees of the superpolies of the cube C_{I_3} in the first output bit of Trivium variants with no less than 960 rounds, where $I_3 = \{0, 3, 6, \ldots, 69, 72, 75, 78\}$. When evaluating the degrees of the superpolies, we use the method proposed in [24] which could return upper bounds of degrees of the superpolies. Table 3 summarises the upper bounds of degrees of superpolies of C_{I_3}.

Table 3. Upper bounds of degrees of superpolies of C_{I_3}

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$j = 960$, C_1	3	2	3	3	3	5	5	4	0	2	3	4	5	7	5	3	2	3	3	3	2	6	8	7	5	4
$j = 960$, C_2	0	0	0	1	2	4	4	5	4	2	2	1	2	2	4	4	3	0	1	2	4	3	4	5	7	8
$j = 960$, C_3	4	4	4	4	4	6	6	4	0	4	5	7	7	7	5	4	4	4	4	4	4	7	9	9	6	5
$j = 986$, C_1	4	7	7	5	6	6	3	3	3	6	6	9	11	11	10	6	7	6	7	9	10	9	13	16	15	13
$j = 986$, C_2	4	5	5	5	4	6	6	7	4	2	3	4	6	8	8	10	10	7	5	5	6	7	7	8	10	13
$j = 986$, C_3	7	8	9	7	8	8	4	4	4	7	7	11	12	13	10	6	8	9	9	10	10	9	15	17	17	14
$j = 1012$, C_1	13	12	12	13	14	13	11	7	9	11	13	14	17	18	17	17	12	12	14	14	15	15	18	17	16	13
$j = 1012$, C_2	10	7	8	9	9	9	11	10	11	10	8	9	11	12	12	14	13	12	11	8	13	15	15	16	17	16
$j = 1012$, C_3	14	14	13	14	14	14	11	9	10	13	14	16	18	20	20	17	14	14	15	16	17	17	20	19	17	15
$j = 1038$, C_1	15	16	16	18	20	19	19	14	14	15	19	18	19	21	22	22	19	17	20	20	20	20	20	19	18	19
$j = 1038$, C_2	14	11	10	12	15	16	18	20	18	16	16	15	15	16	16	19	21	18	16	17	17	17	19	21	21	19
$j = 1038$, C_3	16	17	19	21	21	22	20	15	16	18	21	21	22	24	25	24	20	19	21	22	21	21	23	21	20	21
$j = 1064$, C_1	19	21	23	24	26	23	23	22	20	22	23	25	26	26	26	26	26	23	25	26	26	26	26	26	26	26
$j = 1064$, C_2	15	16	18	19	20	20	21	21	17	17	18	20	20	21	22	24	24	20	17	18	20	22	23	25	26	26
$j = 1064$, C_3	21	22	25	26	26	26	26	24	22	24	24	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26
$j = 1090$, C_1	26	26	26	26	26	26	26	26	26	26	26	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
$j = 1090$, C_2	24	22	22	25	24	25	24	26	26	24	**23**	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
$j = 1090$, C_3	26	26	26	26	26	26	26	26	26	26	26	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Similarly, we do the same experiments for $I_4 = I_2 \cup \{73, 76, 79\}$ under the conditions C_4, C_5 and C_6 respectively. We summarise the upper bounds of the degrees of superpolies of C_{I_4} in Table 4.

Table 4. Upper bounds of degrees of superpolies of C_{I_4}

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$j = 960$, C_4	3	2	3	2	1	4	6	4	2	0	2	3	4	5	5	2	2	3	2	1	2	3	5	7	7	5
$j = 960$, C_5	0	0	0	0	0	2	3	3	3	3	2	1	2	1	4	4	2	1	0	1	2	3	2	3	5	8
$j = 960$, C_6	3	3	3	3	2	6	6	5	3	0	3	4	5	6	6	3	2	3	2	2	3	3	5	8	8	5
$j = 986$, C_4	4	5	6	5	3	6	7	2	1	4	4	6	9	11	10	8	7	6	5	8	9	8	12	14	15	14
$j = 986$, C_5	7	2	3	4	4	4	5	5	4	4	2	2	4	6	6	9	8	7	4	3	4	5	5	6	8	11
$j = 986$, C_6	5	6	7	7	6	7	7	2	1	4	5	7	10	12	10	8	7	7	7	8	9	8	14	16	17	14
$j = 1012$, C_4	10	12	11	12	12	14	12	9	8	8	11	13	14	17	16	16	14	11	12	14	13	13	14	15	17	13
$j = 1012$, C_5	11	10	6	7	9	8	7	10	10	9	9	9	9	11	10	11	13	12	9	11	11	11	13	13	14	14
$j = 1012$, C_6	10	12	12	12	12	14	12	10	9	10	12	14	15	17	17	17	16	11	12	14	15	15	16	17	17	14
$j = 1038$, C_4	13	14	14	16	17	19	18	16	14	13	16	17	17	17	21	21	21	17	19	21	20	19	18	19	19	17
$j = 1038$, C_5	12	11	10	10	11	15	16	18	18	15	13	13	14	14	14	16	18	18	14	14	16	16	16	18	20	19
$j = 1038$, C_6	15	15	16	18	18	19	19	16	14	15	17	18	19	19	22	23	22	18	21	21	21	20	19	20	19	18
$j = 1064$, C_4	18	18	19	24	23	23	22	23	21	20	20	22	23	26	26	26	26	24	21	23	25	26	26	26	26	26
$j = 1064$, C_5	16	14	14	18	18	17	19	20	20	16	16	18	18	18	18	19	22	20	18	16	17	21	20	21	23	25
$j = 1064$, C_6	18	20	21	24	25	25	26	24	21	21	21	22	24	26	26	26	26	25	23	25	26	26	26	26	26	26
$j = 1090$, C_4	25	26	26	26	26	26	26	26	26	24	26	26	26	26	26	26	26	26	26	26	26	–	–	–	–	–
$j = 1090$, C_5	24	22	22	21	23	23	23	23	25	24	23	23	26	26	26	25	26	26	24	25	**25**	–	–	–	–	–
$j = 1090$, C_6	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	–	–	–	–	–

According to Tables 3 and 4, it can be seen that there are many rounds such that the degrees of the superpolies of C_{I_3}/C_{I_4} are less than 10. For example, for I_3 under the condition C_2, the degree of its superpoly in the first output bit of the 1031-round Trivium is $8 < 10$. Furthermore, for the I_4 (resp. I_3) under the condition C_5 (resp. C_2), the degree of superpoly is not larger than 23 (resp. 24) for Trivium reduced to 1108 (resp. 1100) rounds. The probability of a random polynomial in 26 variables of degree not larger than 24 is 2^{-27}, since each term occurs in a random polynomial with the probability 1/2 and there are 27 terms of degree larger than 24 not appearing in the polynomial. Similarly, a random polynomial in 26 variables of degree not larger than 23 is 2^{-327}. Note that our superpoly contains 26 variables. Thus, our observation that the degree of the superpoly is not larger than 24 or 23 shows a kind of non-randomness for Trivium in terms of algebraic property.

Remark 3. As comparisons, we estimate the degree of the superpolies of C_{I_1} under another two conditions C_7 and C_8. In C_7, a natural condition which could be obtained by observing the structure of Trivium, we set $k_{3\cdot i+1} = 0$ for $0 \le i \le 26$ and $k_{3\cdot i+2} = 0$ for $0 \le i \le 25$. Under the condition C_7, we could observe the similar nonrandomness for Trivium reduced to 1072 rounds. Furthermore, we do similar experiments on C_8 which is obtained by randomly add one condition on key variables, i.e. $k_0 = 0$, to C_7. Under the condition C_8, we could observe the similar nonrandomness for Trivium reduced to 1072 rounds. It can be seen that there is a gap between 1072 and 1108. It indicates that the conditions found by our method are superior to those natural conditions.

5 Conclusion

In this paper, we focus on conditional differential attacks. Deriving and analysing the conditions imposed to control the propagation of the chosen differences is a key step of conditional differential attacks. To determine a proper differential characteristic of a set of derived conditions automatically, we apply the MILP method to conditional differential attacks. Our new method enables us to consider all the derived conditions together as well as take the space of valid input into account. As illustrations, we apply our method to Trivium. Combining with the idea of cube attacks, we do key-recovery attacks for up to the 978-round Trivium in the weak-key setting. Furthermore, we could detect non-randomness up to the 1108-round Trivium in the weak-key setting.

Appendix

In [17], the authors introduce how to describe the $b = a_1 \oplus a_2$ with linear constraints, which is shown in the following.

$$\begin{cases} \mathcal{M}.con \leftarrow b + a_1 + a_2 \geq 2d \\ \mathcal{M}.con \leftarrow d \geq b \\ \mathcal{M}.con \leftarrow d \geq a_1 \\ \mathcal{M}.con \leftarrow d \geq a_2 \end{cases}$$

The above procedure is denoted by $(\mathcal{M}, b) \leftarrow Xor2(\mathcal{M}, a_1, a_2)$. Based on this method, Algorithm 5 shows how to describe $y = x_1 \oplus x_2 \oplus \cdots \oplus x_m$ for $m \geq 2$ with linear constraints. In Algorithm 5, we first convert $y = x_1 \oplus x_2 \oplus \cdots \oplus x_m$ to $y = y_1 \oplus y_2 \oplus \cdots \oplus y_{\lceil m/2 \rceil}$, where $y_i = x_i \oplus x_{i+1}$ ($y_{\lceil m/2 \rceil} = x_m$ is m is an odd integer). Then, we do this operation repeatedly until there are only two variables being involved in the XOR operation. Finally, by applying the procedure $Xor2$ to the final two variables, we could represent $y = x_1 \oplus x_2 \oplus \cdots \oplus x_m$ with linear constraints. With the above procedure, we could build an MILP model which describes $y = x_1 \oplus x_2 \oplus \cdots \oplus x_m$ equivalently.

References

1. Subhadeep Banik. Conditional differential cryptanalysis of 105 round Grain v1. *Cryptography and Communications*, 8(1):113–137, 2016.
2. Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 272–288, 2009.
3. Christophe De Cannière and Bart Preneel. Trivium. In *New Stream Cipher Designs - The eSTREAM Finalists*, pages 244–266. 2008.
4. 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. *J. Cryptology*, 31(3):885–916, 2018.

Algorithm 5 MILP model of the XOR operation

```
1: Set $i = 1$;
2: Initialize $L$ as $L = (x_1, x_2, \cdots, x_m)$;
3: while $m > 2$ do
4:     $j = 0$;
5:     for $j \leq m$ do
6:         $\mathcal{M}.var \leftarrow y_i$ as binary;
7:         if $j \neq m - 1$ then
8:             $(\mathcal{M}, y_i) \leftarrow Xor2(\mathcal{M}, L[j], L[j+1])$, where $L[j]$ is the $j$-th element of $L$;
9:         else
10:            $\mathcal{M}.con \leftarrow y_i = L[j]$;
11:        end if
12:        Set $j = j + 2$;
13:        Set $i = i + 1$;
14:    end for
15:    Set $m = \lceil m/2 \rceil$;
16:    Delete all the elements in $L$;
17:    Set $L$ as $L = (y_{i-m}, y_{i-m+1}, \ldots, y_{i-1})$;
18: end while
19: $(\mathcal{M}, y) \leftarrow Xor2(\mathcal{M}, L[0], L[1])$;
```

5. Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 278–299, 2009.
6. Itai Dinur and Adi Shamir. Breaking Grain-128 with dynamic cube attacks. In *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, pages 167–187, 2011.
7. Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 502–517, 2013.
8. Ximing Fu, Xiaoyun Wang, Xiaoyang Dong, and Willi Meier. A key-recovery attack on 855-round Trivium. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 160–184, 2018.
9. Zonghao Gu, Edward Rothberg, and Robert Bixby. Gurobi optimizer. http://www.gurobi.com/.
10. Yonglin Hao, Lin Jiao, Chaoyun Li, Willi Meier, Yosuke Todo, and Qingju Wang. Observations on the dynamic cube attack of 855-round TRIVIUM from crypto'18. Cryptology ePrint Archive, Report 2018/972, 2018. https://eprint.iacr.org/2018/972.
11. Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *IJWMC*, 2(1):86–93, 2007.
12. Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of NLFSR-Based cryptosystems. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of*

Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings, pages 130–145, 2010.

13. Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of Trivium and KATAN. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 200–212, 2011.
14. Jun-Zhi Li and Jie Guan. Advanced conditional differential attack on Grain-like stream cipher and application on Grain v1. *IET Information Security*, 13(2):141–148, 2019.
15. Meicheng Liu, Jingchun Yang, Wenhao Wang, and Dongdai Lin. Correlation cube attacks: From weak-key distinguisher to key recovery. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 715–744, 2018.
16. Zhen Ma, Tian Tian, and Wen-Feng Qi. Improved conditional differential attacks on Grain v1. *IET Information Security*, 11(1):46–53, 2017.
17. Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, pages 57–76, 2011.
18. Piotr Mroczkowski and Janusz Szmidt. Corrigendum to: The cube attack on stream cipher Trivium and quadraticity tests. *IACR Cryptology ePrint Archive*, 2011:32, 2011.
19. Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 185–215, 2017.
20. Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747, 2014. https://eprint.iacr.org/2014/747.
21. Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 158–178, 2014.
22. Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, pages 250–279, 2017.
23. Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on Non-Blackbox polynomials based on division property. *IEEE Trans. Computers*, 67(12):1720–1736, 2018.
24. Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 275–305, 2018.

25. SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang, and TaiRong Shi. A practical method to recover exact superpoly in cube attack. Cryptology ePrint Archive, Report 2019/259, 2019. https://eprint.iacr.org/2019/259.
26. Yuhei Watanabe, Takanori Isobe, and Masakatu Morii. Conditional differential cryptanalysis for Kreyvium. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part I*, pages 421–434, 2017.
27. Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 648–678, 2016.
28. Chen-Dong Ye and Tian Tian. An algebraic method to recover superpolies in cube attacks. Cryptology ePrint Archive, Report 2018/1082, 2018. https://eprint.iacr.org/2018/1082.
29. Chen-Dong Ye and Tian Tian. A new framework for finding nonlinear superpolies in cube attacks against Trivium-Like ciphers. In *Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings*, pages 172–187, 2018.