IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Cube distinguisher extraction using division property in block ciphers

ISSN 1751-8709
Received on 25th May 2018
Revised 28th May 2019
Accepted on 19th August 2019
doi: 10.1049/iet-ifs.2018.5252
www.ietdl.org

Zahra Eskandari[1], Abbas Ghaemi Bafghi[1] ✉

[1]*Data and Communication Security Laboratory, Computer Department, Ferdowsi University of Mashhad, Iran*

✉ *E-mail: ghaemib@um.ac.ir*

Abstract: Cube attack, a simplified type of algebraic attack, is widely utilised to cryptanalyse ciphers. However, since the cube attack works without considering the cipher structure, it is highly complex. In 2017, division property, a successful approach to finding integral distinguishers, was used to extract cube distinguishers in a non-blackbox manner for stream ciphers, which led to a significant improvement of the previous results. This is the first paper employing division property for cube distinguisher extraction in block ciphers. To do this, first, an approach relying on Boolean satisfiability problem (SAT) is presented to evaluate the propagation of division property. Indeed, extraction of zero-sum distinguisher is mapped on a SAT problem and SAT solvers are used to finding division trails efficiently and automatically. Then, this approach is extended and adapted to extract cube distinguishers in block ciphers. However, there are similarities between our contribution and others but the different structure of block and stream ciphers lead to disparity in applying division property to extract cube distinguisher for block ciphers. To prove the efficiency of the presented approach, it is applied to the lightweight block cipher KATAN and the cube distinguishers are extended to a higher round in comparison with previous results.

1 Introduction

In algebraic view, the cryptanalysis of symmetric ciphers can be reduced to the problem of solving a large non-linear multivariate polynomial system, which represents an nondeterministic polynomial-complete problem [1]. There are several approaches to solve the non-linear polynomial system [2–7], but they have high memory consumption and time complexity beyond the power of current computers, which make them infeasible in practical problems. In 2009, cube attack [8], a simplified type of algebraic attacks, was proposed. Later its efficiency was improved in its successors, cube tester [9] and dynamic cube attack [10]. The idea behind cube attack is very similar to algebraic IV differential attack [11].

Cube attacks use the algebraic normal form (ANF) representation of each ciphertext bit as a function of plaintext and key bits in a blackbox manner where inputs comprise plaintext bits as public variables and key bits as secret variables. The main idea is to find information about the secret variables of the cipher by applying all values for a well-chosen subset of the public variables and summing up all the corresponding outputs. By evaluating this sum, some linear equations of key bits are extracted. Then the extracted equations are solved and key bits are recovered. Cube attack family has been used extensively in cryptanalysis of stream ciphers, but unlike its remarkable results in cryptanalysis of stream ciphers, it rarely has been applied to block ciphers [12–15].

As discussed in [8], cube attacks are similar to integral attacks [16] as a type of high-order differential attack [17] over the binary field F_2. In all of these attacks, the attacker attempts to sum the output of ciphers in different but related input values.

Division property [18], a method to find high-order differential distinguishers, specifically integral distinguisher, was proposed in 2015. In brief, for a set of chosen plaintexts, the division property is deduced from the propagation rules over several rounds and at the end, the presence of a balanced bit is determined as a zero-sum integral distinguisher. However, the division property proposed in [18] was only useful for S-box-based ciphers, and it was used to break the full MISTY1 block cipher, but it could not be applied to non-S-box-based ciphers effectively. To fill this gap, in [19], the bit-based variant was introduced, which focused on an automatic search of integral distinguishers at the bit level.

Evaluating the propagation of division property is not as easy as the size of the problem increases extremely. In 2016, Xiang *et al*. [20] showed that the propagation was efficiently evaluated using mixed integer linear programming (MILP) with the aim of applying propagation rules and extract zero-sum integral distinguishers. Some instances of applying this tool to find integral distinguishers are presented in [21, 22]. Recently, in [23], automatic tools were proposed to apply the division property method at the bit level over Add-Rotate-XOR (ARX) ciphers, which relied on Boolean satisfiability problem (SAT) instead of MILP. In addition, for word-based ciphers, they developed the automatic search tool based on the satisfiability modulo theories (SMTs), which was a generalisation of SAT. Using this tool, they found improved distinguishers as well as some optimised distinguishers with lower data complexity for some ARX and word-based ciphers.

Considering the structural similarities between the cube and integral attacks and the success of division property in finding integral distinguishers, in 2017, division property was used to extract cube distinguishers for stream ciphers [24]. As claimed by the authors, the main property of this new technique was that the cipher structure was never regarded as a blackbox and it could be analysed in detail. This approach used the division property to analyse the ANF of cipher output based on the evaluation of propagations from an initial division property in a cube. Given that the propagation of the division property could determine key bits involved in the superpoly, it diminishes the complexity of its recovery. The new cube attack was applied to TRIVIUM, GRAIN128, ACORN, and KREYVIUM. Therefore, these attacks are considered as the best key-recovery attack against these ciphers.

In [25], more accurate MILP models were designed in comparison with [24] due to the consideration of the algebraic construction of the superpoly, which led to decreased complexity of superpoly recovery. Hence, they mounted the attack in more rounds, and in some cases, they applied the attack to larger cube sizes. In [26], the lightweight stream cipher WG-5 was analysed using cube attacks based on division property. This cube attack was automated by MILP models to confine the complexity of the superpoly recovery theoretically. As a result, division property significantly decreased data complexity compared to other algebraic attacks on WG-5.

1.1 Our contributions

In this study, considering the importance of the automated tool for evaluation of division property, we show how the bit-based division property could be mapped to a SAT problem. Indeed, to find a division trail for a cipher, we construct a Boolean formula, which is satisfiable, if and only if it forms a valid division trail.

Then this approach is extended and adapted to find a cube distinguisher for block ciphers in a non-blackbox manner using the division property. To do this, in addition to issues raised in [24], the differences between block and stream ciphers should be considered. As far as we know, it is the first application of the cube extraction to block ciphers using division property.

1.2 Results

To prove the efficiency of division property in cube distinguisher extraction of block ciphers, it was applied to a lightweight block cipher called KATAN [27], which is an ARX block cipher. The results showed that it outperformed the results of the best cube distinguishers on KATAN. These improvements were as follows:

- Linear equation extraction for 72-round and 90-round KATAN32 and 50-round KATAN48, which was an extension of the 60-round KATAN32 and 40-round KATAN48 results reported in [12], respectively.
- Constant cube distinguisher extraction for KATAN32, which extended the previous results [14] from 74-round to 91-round and 83-round to 101-round in cube sizes 30 and 31, respectively.

1.2.1 Outline of the paper: The remainder of this paper is organised as follows. In Section 2, we briefly review preliminaries such as cube attack details, division property concept, and KATAN cipher description. In Section 3, we focus on our contribution and description of its details. Section 4 presents the results and draws a comparison with the previous studies. Conclusions are drawn in Section 5. Some details of the results are provided in the Appendix.

2 Preliminaries

In this section, we describe the basic concepts of this study. First, a brief overview of the cube attack is given, and then the concepts of division property are discussed. Finally, while presenting the results over KATAN block cipher family, the specifications of the cipher are described.

2.1 Cube attacks

In 2009, cube attack [8] was proposed as a type of algebraic attack, which considered the cipher as a blackbox and used the ANF representation of the ciphertext bit as a polynomial function of plaintext and key bits. In the cube attack, by evaluating this function for all possible values of some plaintext bits, the attacker attempts to extract some linear equations of key bits, i.e. consider each ciphertext bit as a polynomial function of plaintext bits, $V = \{v_0, v_1, \dots\}$, and key bits, $K = \{k_0, k_1, \dots\}$. Assume $x = V \bigcup_K$, so one can write any of the ciphertext bits as a polynomial $p(x)$ over GF(2) in ANF representation. Let I be a subset of the plaintext bits and t_I be the product of all variables whose indexes are in I. The ciphertext bit can be considered as a polynomial function of V and K variables

$$p(x) = F(V, K) = t_I \cdot Ps(I) + q(x) \tag{1}$$

Here, t_I and $Ps(I)$ are called cube and superpoly, respectively. $Ps(I)$ is a polynomial of the key bits and remaining plaintext bits. $q(x)$ is the sum of all terms that do not contain at least one term of I. According to the theorem proved in [8], $Ps(I)$ can be calculated by applying a higher-order derivative

$$Ps(I) \equiv \sum_{v \in C_I} p(x) (\bmod 2) \equiv \sum_{v \in C_I} F(V, K) (\bmod 2) \equiv F(W, K) \tag{2}$$

where C_I is the set of all possible values for variables in t_I and other bits not included in I are zero. Since the terms in $q(x)$ are added an even number of times, they are cancelled out in modulo 2. Therefore all the terms except those are contained in the superpoly $Ps(I)$ are eliminated. By defining $W = \{i \in V | i \notin I\}$ as all other public variables that are not included in subset I, the superpoly $Ps(I)$ is a polynomial function of W and K. Since in cube attack scenario, the bits in W have zero value, if $Ps(I)$ constitutes a linear polynomial of K, it can be utilised in an attack scenario.

There are two phases in the attack scenario, an offline preprocessing phase and an online phase. The goal of the offline preprocessing phase is to find some cubes that cause linear superpoly. The linear superpoly can be determined with some linearity tests [28]. In a cube with linear superpoly, t_I is called a maxterm. The primary challenge of the cube attack is to find proper maxterms. In the offline phase, in some heuristics such as the random walk manner [8], a subset I is selected as the cube. When the subset I is too small, the corresponding coefficient is likely to be a non-linear function of the key variables. To have a lower degree coefficient, some plaintext variables should be added to the subset. When I is too large, the sum will be a constant function, which in this case does not depend on the key bits. In this scenario, some plaintext variables should be eliminated from the subset.

Once sufficient maxterms are obtained, the online phase starts. In the online phase, with a fixed but unknown key, by evaluating the function for all possible values of maxterm variables, a system of linear equations is obtained, which can be solved efficiently by using Gaussian elimination. By doing so, we can finally recover the values of the key bits. This attack was made to reduced versions of the stream cipher TRIVIUM and a major improvement compared to previous studies was achieved.

The most important extensions of cube attack are cube testers [9] and dynamic cube attacks [10]. Cube testers introduced in [9] represent a family of distinguishers to detect non-random behaviour. Both cube attacks and cube testers sum up the output of a cipher for a subset of plaintext bit values. In the cube testers, however, instead of finding linear equations, as is the case with cube attack, which increases the complexity of the preprocessing phase, the attacker seeks to find statistical distinguishers for the cipher and utilises it to distinguish the cipher from a random function.

In 2011, dynamic cube attack [10] was proposed, utilising cube testers. This attack uses the internal structure of the cipher and simplifies its output ANF representation by nullifying some state bits by means of dynamic variables. In the simplified output, the cube tester properties such as constantness could be mounted to higher round. The preprocessing phase of this attack is based on cube testers and it is less complex than the cube attack. Finding a distinguisher for the cipher, this is utilised for the recovery of secret key bits. In [10], the dynamic cube attack was presented on GRAIN128.

2.2 Division property

Division property, which was proposed in 2015 [18], is an improved technique to find integral distinguisher for iterated ciphers. In brief, for a set of chosen plaintexts based on the initial division property, the division property for one round is deduced from the propagation rules in accordance with the round function. Therefore, the division property can be exploited in several rounds and the existence of integral distinguishers in r-round output can be determined. In [19], the bit-based variant was introduced to apply division property to non-S-box-based ciphers. This approach is studied in detail here.

2.2.1 Bit-based division property: In the following, a brief review of the definition of bit-based division property [19] and propagation rules for basic operations involved in the encryption process is presented.

Definition 1: (bit-based division property): Let $\mathbb{X}$ be a multiset whose elements take a value of F_2^n. Let $\mathbb{K}$ be a set whose elements take an n-dimensional bit vector. When the multiset $\mathbb{X}$ has the division property $D_K^{1^n}$, it meets the following conditions:

$$\bigoplus_{x \in X} x^u = \begin{cases} \text{unknown} & \text{if there exist} \quad k \in \mathbb{K} \quad \text{s.t.} \quad u \succeq k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

To find the integral distinguisher using division property, the attacker chooses the initial value for the division property and then based on the round function of the cipher, the initial division property is propagated through the rounds. The propagation rules for the bit-based operation involved in the round function of ciphers, as proven in [18, 19], are as follows:

Rule (*copy* [18]**)**. Let F be a *copy* function, where the input x takes value from $\mathbb{F}_2^n$ and the output is calculated as $(y_0, y_1) = (x, x)$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multisets, respectively. Assuming that $\mathbb{X}$ has the division property $D_{\{k\}}^n$, the division property of $\mathbb{Y}$ is $D_{\mathbb{K}'}^{n,n}$, where

$$\mathbb{K}' = \{(k-i, i) | 0 \le i \le k\} \tag{4}$$

Rule **(***XOR* [18]**)**. Let F be an *XOR* function, where the input (x_0, x_1) takes value from $\mathbb{F}_2^n \times \mathbb{F}_2^n$ and the output is calculated as $y = x_0 \oplus x_1$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multisets, respectively. Assuming that $\mathbb{X}$ has the division property $D_{\mathbb{K}}^{n,n}$, the division property of $\mathbb{Y}$ is $D_{\{k'\}}^n$, where

$$k' = \min \{k_0 + k_1 | (k_0, k_1) \in \mathbb{K}\} \tag{5}$$

Here, if k' is larger than n, the propagation characteristic of the division property is aborted, i.e. the value of $\bigoplus_{y \in \mathbb{Y}} \pi_v(y)$ is 0 for all $v \in \mathbb{F}_2^n$.

Rule (*AND* [19]). Let F be an *AND* function, where the input (x_0, x_1) takes value from $\mathbb{F}_2 \times \mathbb{F}_2$ and the output is calculated as $y = x_0 \wedge x_1$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multisets, respectively. Assuming that $\mathbb{X}$ has the division property $D_{\mathbb{K}}^{1,1}$, the division property of $\mathbb{Y}$ is $D_{\{k'\}}^1$, where

$$\mathbb{K}' = \{\lceil (k_0 + k_1)/2 \rceil | k = (k_0, k_1) \in \mathbb{K}\} \tag{6}$$

As discussed earlier, to find integral distinguisher, the attacker determines indexes $I = \{i_1, i_2, \ldots, i_{|I|}\}$, where I is a subset of plaintext indexes and variables indexed by I are taking all possible combinations of values. The division property of these chosen plaintexts is $D_k^{1^n}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Then, the propagation of division property from $D_k^{1^n}$ is evaluated as

$$\{k\} = \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \mathbb{K}_r, \tag{7}$$

where $D_{\mathbb{K}_i}$ is the division property after i-round propagation. If the division property $\mathbb{K}_r$ contains all vectors $\boldsymbol{e}_i$ with only ith element is 1, then there is no balance bit. Otherwise if $\mathbb{K}_r$ does not have a unit vector $\boldsymbol{e}_i$, then the ith bit of r-round ciphertext is balanced. This propagation is considered as a division trail.

2.2.2 Evaluating division property: As mentioned earlier, given the high-growth rate of size $\mathbb{K}_i$, it is difficult to evaluate the propagation of division property manually. In 2016, Xiang *et al*. [20] showed that the propagation could be efficiently evaluated using MILP. These tools, which work at the bit-based level, model the division property propagations of the operations as linear inequalities and constructs a linear inequality system that describes the division property propagations of a block cipher based on initial division property. Then, by selecting a proper objective function, the search for an integral distinguisher is converted into a MILP problem. Some studies based on MILP and constraint programming can be found in [21, 22, 29].

In [23], an automatic tool to evaluate division property was proposed. This tool is underlined by Boolean satisfiability problem (SAT) rather than MILP to evaluate the bit-based division property in ARX ciphers. For the word-based division property, the automatic search is performed based on SMTs, which is a generalisation of SAT. In their work, the propagation of division property is translated into a system of logical equations in conjunctive normal form (CNF). As noted in their work, in addition to these logical equations that correspond to r-round propagations, some logical equations based on initial division properties and stopping rule are also adapted. Using their tool, some new distinguishers and distinguishers with more rounds were found for a number of ARX and word-based ciphers.

2.3 KATAN block cipher

KATAN is a family of lightweight block ciphers with three variants of 32-, 48- or 64-bit block size. All versions have 254 rounds and use an 80-bit main key. KATAN consists of two linear feedback shift registers, called L1 and L2, which are loaded with the plaintext. They are then transformed by two non-linear Boolean functions as follows:

$$\begin{aligned} f_a(L1) &= L1[x_1] \oplus L1[x_2] \oplus L1[x_3] \cdot L1[x_4] \oplus L1[x_5] \cdot IR \oplus k_a \\ f_b(L2) &= L2[y_1] \oplus L2[y_2] \oplus L2[y_3] \cdot L2[y_4] \oplus L2[y_5] \cdot L2[y_6] \oplus k_b \end{aligned} \tag{8}$$

The infrared used in the first function represents the irregular update rule, which consists of the output of an linear feedback shift register (LFSR). The output of each function is loaded on the least significant bits of the other LFSR after being left-shifted. This operation is invertible. The parameters of the above function are presented in Table 1.

For the ith round, only two key bits ($k_a = k_{2i}$ and $k_b = k_{2i+1}$) are used. For KATAN48, these functions are applied twice in one round of the cipher. First, a pair of functions is applied, and then after the update of registers, they have applied again, using the same subkeys. For KATAN64, each Boolean function is applied three times for each round, with the same pair of key bits.

The key schedule algorithm of all KATAN ciphers is a linear mapping that expands an 80-bit key K into $2 \times 254 = 508$ sub-key bits, as described below

$$k_i = \begin{cases} K_i & \text{for} \quad 0 \le i \le 79 \\ k_{i-80} + k_{i-61} + k_{i-50} + k_{i-13} & \text{otherwise} \end{cases} \tag{9}$$

See [27] for more details.

Table 1 Parameters used in non-linear functions

Cipher	$\|L1\|$	$\|L2\|$	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5	y_6
KATAN32	13	19	12	7	8	5	3	18	7	12	10	8	3
KATAN48	19	29	18	12	15	7	6	28	19	21	13	15	6
KATAN64	25	39	24	15	20	11	9	38	25	33	21	14	9

3 Utilising division property to extract cube distinguishers

As discussed earlier, finding an integral distinguisher is difficult on account of the extreme growth of the problem size. Particularly for bit-based designs, the analysis often requires extensive manual work, which may be prone to errors. Thus, automatic tools can be useful to simplify the analysis of cryptographic primitives. Here, considering the importance of automated evaluation of division property, we argue that the issue of finding the integral distinguishers can be efficiently automated using SAT solvers. To do this, the bit-based division property is mapped into the conditions on state variables and consequently, the search for division trail is converted into a SAT problem.

Then, the above approach is extended and adapted to extract cube distinguishers for block ciphers in a non-blackbox setting based on the division property. As far as we know, it is the first application of the division property in the cube distinguisher extraction for block ciphers. There are some similarities between our contribution and that of Todo *et al*. [24], but the different structure of block and stream ciphers produces disparities in applying division property to the cube distinguisher extraction in block ciphers.

3.1 Automated findings of integral distinguisher with SAT

The Boolean satisfiability problem (SAT) is a well-known problem in computer science. The problem is to decide whether there exists an assignment of variables in a Boolean formula in CNF such that the evaluation of the formula is true. In the following, an automated approach is presented, which simplifies the problem of finding a division trail to a Boolean satisfiability problem. Indeed, to find a division trail for a cipher, a Boolean formula is constructed, which is satisfiable if and only if it develops a valid division trail.

In the first step, we introduce a variable for each bit of the cipher state at round i, $S^i = (s_0, \ldots, s_{n-1})$, where n is the size of the state. After choosing a proper initial division property value for the S^0, the next step is to propagate the initial value through rounds based on the propagation rules for operations used in the round function. The main operation has been studied in [18, 19], so we focused on how a Boolean formula in CNF could be constructed for valid transitions of each operation.

3.1.1 Division property propagation: Here, for the main operations discussed in Section 2.2.1, we generated the valid clauses in the CNF form. For each occurrence of these operations in the round function of the cipher, the corresponding clauses are appended to the SAT problem clause set. Further details about the rules to define the valid transitions and how the Boolean formulas in CNF format were constructed, can be found in [30].

Copy: This operation copies an input bit a on an output bit b. The valid transitions are given by

$$\begin{aligned}&\mathbf{copy}(a_{\text{old}}, b_{\text{old}}) \to \{(a_{\text{new}}, b_{\text{new}})\}\\ &\mathbf{copy}(0,0) \mapsto \{(0,0)\}\\ &\mathbf{copy}(1,0) \mapsto \{(1,0),(0,1)\}.\end{aligned}$$

The set of clauses C_{copy} which form a Boolean formula are given by

$$\begin{aligned}C_{\text{copy}} = \{&(\neg b_{\text{old}}), (\neg a_{\text{old}} \vee b_{\text{new}} \vee a_{\text{new}}), (a_{\text{old}} \vee \neg b_{\text{new}}),\\ &(a_{\text{old}} \vee \neg a_{\text{new}}), (\neg a_{\text{new}} \vee \neg b_{\text{new}})\}.\end{aligned} \tag{10}$$

XOR: This operation corresponds to $a \oplus b \to b$. The valid transitions are given by

$$\begin{aligned}&\mathbf{XOR}(a_{\text{old}}, b_{\text{old}}) \to \{(a_{\text{new}}, b_{\text{new}})\}\\ &\mathbf{XOR}(0,0) \mapsto \{(0,0)\}\\ &\mathbf{XOR}(0,1) \mapsto \{(0,1)\}\\ &\mathbf{XOR}(1,0) \mapsto \{((0,1),(1,0)\}\\ &\mathbf{XOR}(1,1) \mapsto \{((1,1)\}.\end{aligned}$$

and the corresponding clauses are as follows:

$$\begin{aligned}C_{\text{XOR}} = \{&(a_{\text{old}} \vee \neg a_{\text{new}}), (\neg b_{\text{old}} \vee b_{\text{new}}),\\ &(\neg b_{\text{old}} \vee \neg b_{\text{new}} \vee \neg a_{\text{new}}), (\neg a_{\text{old}} \vee a_{\text{new}} \vee b_{\text{new}}),\\ &(b_{\text{old}} \vee a_{\text{old}} \vee \neg b_{\text{new}}), (\neg b_{\text{old}} \vee \neg a_{\text{old}} \vee a_{\text{new}})\}.\end{aligned} \tag{11}$$

AND: This operation corresponds to $a \wedge b \to b$. The valid transitions are given by

$$\begin{aligned}&\mathbf{AND}(a_{\text{old}}, b_{\text{old}}) \to \{(a_{\text{new}}, b_{\text{new}})\}\\ &\mathbf{AND}(0,0) \mapsto \{(0,0)\}\\ &\mathbf{AND}(0,1) \mapsto \{(0,1)\}\\ &\mathbf{AND}(1,0) \mapsto \{(1,0),(0,1)\}\\ &\mathbf{AND}(1,1) \mapsto \{(0,1)\}.\end{aligned}$$

As for other operations, its translation to a SAT sentence is given by

$$\begin{aligned}C_{\text{AND}} = \{&(a_{\text{old}} \vee \neg a_{\text{new}}), (\neg b_{\text{old}} \vee b_{\text{new}}), (\neg b_{\text{new}} \vee \neg a_{\text{new}}),\\ &(\neg a_{\text{old}} \vee b_{\text{new}} \vee a_{\text{new}}), (a_{\text{old}} \vee \neg b_{\text{old}} \vee \neg b_{\text{new}})\}.\end{aligned} \tag{12}$$

3.1.2 Finding integral distinguishers: To identify an integral distinguisher for a cipher, an initial value of S^0 was chosen and it was propagated in r round. Then S^r was checked after r rounds to find whether it reached a certain output or not. Indeed, if we show that an output vector of S^r, which is all zero except for a single **1** in one bit, is unreachable, we will know that this bit is balanced.

In particular, we are interested in knowing whether any bit at the output will be balanced, which is equal to show that at least one of the vectors in the set

$$S^{r+1} \in \{w \in F_2^n \parallel hw(w) = 1\} \tag{13}$$

is unreachable, where hw is the Hamming weight of the vector.

After generating the clause set of the problem as described above, the problem is solved with a SAT solver. This solution indicates the existence or non-existence of the balance bits at the output of the cipher based on (13). It means that if the model is infeasible, there will be one w that is unreachable, and therefore there is a balanced bit in the output of the cipher. Otherwise, if the model is feasible, it means that all w exist in the S^{r+1} and the cipher output is non-balanced.

In our previous work [30], as described above, we presented a framework to automatically find integral distinguishers by reducing the problem to a SAT problem. It was implemented by providing a simple way to describe primitives, allowing both designers and cryptanalysts to evaluate cryptographic primitives against this attack. Using this tool, we managed to find several new or improved bit-based division property distinguishers for ciphers in comparison with [20, 23]. It should be noted that compared to [23], our proposed tool covers a larger class of cryptographic primitives with various design strategies. It is notable that using our framework, we obtained the integral distinguisher presented at [21] for KATAN cipher, which was extracted by the MILP approach.

3.2 Adapting division property to extract cube distinguishers

Based on the non-blackbox manner of division property propagation through the rounds, which leads to the advancement of the cube extraction for stream ciphers, here we used division property to extract cube distinguishers in block ciphers and as we expect, it improved the results of the existing cube attack. All

Input: Maxterm: M, ciphertext bit position: *out*
Output: Linear equation or *NULL* value if Linear equation does not exist for Maxterm M at ciphertext bit *out*

$K_I = \emptyset$ //Involved key bits Set
for $i = 0$ to *Key_Length* **do**
  $Clause\_Set = \emptyset$
  //Add plaintext bits initial division property values
  $Clause\_Set.ADD(p_j = 1$ if $binary\_Show(M).at[j] == 1$ else $p_j = 0$, for $0 \leq j < PlainText\_Length)$
  //Add key bits initial division property values
  $Clause\_Set.ADD(k_i = 1)$
  $Clause\_Set.ADD(k_j = 0$, for $0 \leq j < Key\_Length$ and $j \neq i)$
  $Update\_States(Clause\_Set, Rounds)$
  //Add ciphertext bits expected division property values
  $Clause\_Set.ADD(c_j = 1$ if $j = out$ else $c_j = 0$, for $0 \leq j < CipherText\_Length$ and $j \neq out)$
  $Satisfiable = Solve(Clause\_Set)$
  if (Satisfiable) **then**
    $K_I = K_I \cup \{i\}$
  end if
end for
Assume $Equation : C_{out} = \bigoplus_{i \in K_I} k_i$
Perform Linear_Test on *Equation* under N random keys
if (*Equation* pass Linear_Test) **then**
  return *Equation*
else
  return *NULL*
end if

Fig. 1 *Algorithm 1: find a linear equation*

papers on extracting cube distinguishers using division property [24–26] have studied stream ciphers. In this study, considering the differences between block and stream ciphers, we use the division property approach to cube distinguisher extraction for block ciphers.

There are slight differences between integral and cube attacks. With the exception of some minor differences, there seems to be a similarity between zero-sum integral distinguisher, where the sum of output over the cube is 0 for any secret key, and constant property of cube tester, which means a constant superpoly without any involved key bits.

In the division property approach, the attacker chooses some plaintext bits to have all possible values, which we call subset I, and others are set to a constant value at the initial state. Then, based on the round function operations of the cipher, the initial value of the division property is propagated to the next round. After some rounds, the cipher output is evaluated to see whether cipher output bits are balanced or not.

Recall the definition of W from Section 2.1, W represents the set of all public variables not included in set I. Whereas the bits in W have constant values in the division property approach, the existence or non-existence of the term t_I for any value of key bits can be determined. For a non-balance bit, given that the sum value in (2) is non-zero, $Ps(I)$ represents a non-zero function $F(W, K)$. For a balanced bit, however, since the sum is zero, it means that the coefficient of t_I is zero in (1), suggesting that it is equal to the constant cube distinguisher.

Unlike the integral attack, which only considers the existence or non-existence of the term t_I in the output of a cipher, cube attacks move forward and analyse the coefficient of t_I as a function of key bits. To do so, in cube attacks, secret bits are considered as part of the primitive, so that coefficients could be evaluated as a function of the secret input bits. To adapt the division property approach to the scenario of cube attacks, the variables for the secret bits need to be included in the analysis of the round function, as we progress through the rounds.

It should be emphasised that in stream ciphers, key bits load into the initial state of the cipher and the values of these states are updated every round. Thus, by defining new variables for the key states, as in [24], the propagation rules are applied to these states and the values are propagated to the next round similar to other public variables. In block cipher, nonetheless, the main key is loaded into a state and the key schedule function is applied to this state to generate subkeys for each round. Thus, in addition to defining new variables for key bits, the key schedule should be considered in propagation of block ciphers so that key variables are propagated to the next round for the final evaluation of the output function based on the key schedule function.

To analyse the output function, variables k and v are considered for secret and public variables, respectively. Then, to represent the initial division property, $S^0 = (v_0, \ldots, v_{n-1}, k_0, \ldots, k_{m-1})$, where $v_i = 1$ for $i \in I$ and $v_i = 0$ for $i \notin I$. To check the existence or non-existence of a key bit, one element of the secret variables k, j is additionally initiated to 1 in our cube attack. Therefore, we have $k_j = 1$ and $k_i = 0$ for $i \in \{0, \ldots, m-1\}, i \neq j$.

According to (2), here we have

$$Ps(I) \equiv \sum_{v \in C_I'} p(x) (\bmod 2) \equiv \sum_{v \in C_I'} F(V, K) (\bmod 2) \equiv F(K) \quad (14)$$

where C_I' represents the set of all possible values for variables in I and all other bits having zero values. As discussed earlier, since all bits in subset W are zero, $Ps(I)$ is a polynomial function of only key bits. Sum up this function for all possible values of the key bit in index j with all other bits in K having constant values, it exhibits the coefficient of this specific key bit in the superpoly which might be one or zero.

It is worth noting that in the cube attack, to eliminate the maximum possible number of non-linear terms, the constant value is set to zero [8]. In [24], to apply zero value, they added some constraints to the MILP model to manage its propagation. Here, we adopted the same strategy. Since the propagation rules of division property hold for every constant, they are also true for the zero-constant value, and therefore, the propagation rules can be used for the cube attack. The only difference between an arbitrary constant value and the zero value is in non-linear operation such as the AND operation or S-boxes. Hence, when at least one operand of AND operation is zero, the output equals zero. As such, we eliminate this specific AND from the propagation and propagate a zero constant in this case. Also, about S-boxes, these non-linear components can be represented as polynomial functions using the copy, AND, XOR operation with respect to the division property. Thus, the above-mentioned approach to propagate zero value is also applicable to them. In Section 8.1, to study the propagation of division property values through the operations, a toy example was presented. In the example, the different propagation manner of integral and cube attack facing zero values was shown.

By considering these points, in Algorithm 1 (see Fig. 1), we present the steps to find a cube distinguisher using division property specifically. For a determined cube Maxterm, the division

property values of public variables are initialised, as shown in line 4. To check the existence of a specific key bit in the superpoly, as seen in lines 5 and 6, the initial values of secret bits are set. Then these initial values are propagated through rounds by considering both round and key schedule functions and the generated clauses are appended to the clause set of the problem, as shown in line 7. In the next step, to examine the specific ciphertext bit to find cube distinguisher, the corresponding clauses are appended based on (13).

In the end, the problem is solved using a SAT solver. If the model is infeasible, it means that the coefficient of this term is zero and there is no secret variable involved in the superpoly $Ps(I)$ and if the model is feasible, it means that the mentioned key bit exists in superpoly.

To extract cube distinguishers such as linearity and neutrality, we need to determine the subset of key bits that are present or absent in the superpoly. To do this, the above procedure is repeated for all key bits. After determining all the key bits involved in the superpoly, to extract equations, we perform linearity or quadratic test [28, 31] to determine the degree of superpoly and then we can extract its equation. It is notable that as a trivial outcome, the key bits not-involved in the superpoly are considered as neutral key bits.

4 Results

To demonstrate the effectiveness of division property in the cube distinguisher extraction of block ciphers, we applied our approach to the lightweight block ciphers KATAN32 and KATAN48 [27]. In this section, we first applied cube attack to KATAN32 and KATAN48 and some linear equations were extracted for higher-round numbers in comparison with the existing cube attack results [12]. Then, extracted cube distinguishers for KATAN32 were presented, which mounted to higher rounds in the same cube size compared to [14]. The results showed that although the highest number of rounds attacked by non-algebraic techniques [32] could not be reached, considering that our emphasis in this study is on cube distinguishers, the proposed approach outperformed the best existing results for the cube distinguishers on KATAN.

4.1 Linear equation extraction

In [27], the algebraic attack and also cube attack were applied to KATAN family. More specifically, cube attack broke 60-round KATAN32, 40-round KATAN48, and 30-round KATAN64 by extracting 41, 31 and 25 linear equations, respectively. Here, we applied cube attack to KATAN32 and KATAN48 using division property and found cube distinguishers for 72-round KATAN32 and 50-round KATAN48 that outperformed existing cube attacks. The results are summarised in Table 2.

Using the division property, we were able to extract 44 cubes for 72-round KATAN32, as presented in Table 6 (see Section 8.2), including 23 cubes of degree 29 and 21 cubes of degree 31. Some of the ciphertext bits reveal more information about key bits, and we can extract different equations for various ciphertext bits using the same maxterm. This means that the data and computational complexity can be reduced during the online phase.

Thus, the data complexity is

$$20 \cdot 2^{29} + 15 \cdot 2^{31} \simeq 2^{35.3} \tag{15}$$

which constitute 2^{32} chosen plaintexts to determine 44 key bits. The computational complexity is $2^{35.3} + 2^{80-44} \simeq 2^{36}$, which is dominated by the exhaustive search for the remaining 36 key bits. For readers interested in details, we upload the code at https://github.com/zraestgithub/CubeAttack_DP.

Furthermore, three linear equations were extracted for 90-round KATAN32 as presented in Table 7. We can determine three key bits by solving these equations and mount 90-round attack with complexity 2^{77}. Additionally, the extracted equations are the weakness of the cipher and could be utilised besides other attacks to reduce the complexity [12].

The extracted linear equations for 50-round KATAN48 are presented in Table 8 (see Section 8.2). We extracted 24 cubes of varying degrees. Based on the extracted equations, the data complexity is

$$1 \cdot 2^{27} + 4 \cdot 2^{28} + 3 \cdot 2^{29} + 9 \cdot 2^{30} + 4 \cdot 2^{31} \simeq 2^{34.3} \tag{16}$$

chosen plaintexts to determine 24 key bits. The computational complexity is computed by $2^{34.3} + 2^{80-24} \simeq 2^{56}$, which is dominated by the exhaustive search for the remaining 56 bits of the secret key.

In all of our results, we ran 500 linearity tests and then tested the equations for more than 2^{10} distinct random keys to ensure their correctness. To do this, the cube attack was simulated in the compute unified device architecture (CUDA) framework which is used for programming in *Nvidia GPUs* [33] and a high performance computer equipped with *GeForce GTX* 1080 with 8 GB RAM memory and 2560 CUDA cores was used.

4.2 Constant cube distinguisher

To have cube distinguishers in higher round numbers, the maximum value for the cube size should be selected. It means that the maximum numbers of active bits were selected and other bits have zero value. As discussed earlier, the selection of zero value in the cube attack led to maximum elimination of non-linear terms. Thus, it should be noted that the positions of these zero bits affected this elimination in initial rounds, which underscores the importance of the meticulous selection of these zero bit positions.

In the dynamic cube attack [10], considering the above points, the attacker analyses the cipher structure in a backward manner and selects the dynamic variables with maximum possible simplification in inner states of the cipher to identify the longer distinguisher. In a recent study [34], the authors considered more precise nullification technique for inner state bits as well as the frequency of public variables in inner states terms, selecting public variables with high frequency as zero candidates. By applying these techniques, they improved the time complexity of the previous attack on GRAIN 128. Thus, it can be concluded that identifying cube distinguisher in higher round number require precise selection of the zero bits.

It should be mentioned that since the most significant bit of the L2 had the lowest degree compared to other state bits in the same round, it will be further analysed to find a constant cube distinguisher in the higher round. Based on the nullifying techniques discussed in [10, 34], we tried different scenarios for the same cube size and different zero positions such as bits which participated in non-linear terms with lower or upper indexes or bits in the least significant indexes. The results of these experiments are presented in Table 9 (see Section 8.3).

Hence, by zeroing the least significant bit, the longer distinguisher in comparison with other positions was achieved. Owing to the shift operation in the round function of the KATAN family, the zero values in the least significant bits appeared in more terms and were mounted to higher rounds, leading to the

Table 2 Overview of the results obtained for cube attacks on KATAN32 and KATAN48

Cipher	Rounds	Number of equations	Time complexity	Data complexity	Ref
KATAN32	60	41	2^{39}	$2^{30.28}$	[12]
	72	44	2^{36}	2^{32}	section 4.1
KATAN48	40	31	2^{49}	$2^{24.95}$	[12]
	50	24	2^{56}	$2^{34.3}$	section 4.1

Table 3 Extracted distinguishers for KATAN32

Cube size	Distinguisher type	Rounds	Ref.
31	constant cube distinguisher	83	[14]
	zero-sum integral distinguisher	99	[21]
	constant cube distinguisher	101	section 4.2

Table 4 Neutral key bits at cipher bit C_{18} of KATAN32

Cube size	Rounds	Neutral key bit indexes
30	91	all key bits
	92	12, 17, 18, 47, 51, 52, 54, 55, 56, 57, 58, 60, 62, 64, 66
	93	49, 54, 56, 57, 58, 60, 62, 64, 66
	94	58, 62, 66
	95	—
31	101	all key bits
	102	—

elimination of more non-linear terms. Thus, this selection provided distinguishers in higher round number.

In Table 3, the results for the longest distinguisher are presented, which were found in the maximum allowable cube size and compared to other distinguishers. Compared to [14], as described earlier, the accurate selection of the zero bit position mount the distinguisher to a higher round. It is notable that as described in Section 2.3, because of LFSR being left-shifted in KATAN, the 83-round distinguisher in [14] and 101-round distinguisher found here, are the same. So, we can conclude that our findings verify the previous results. In comparison with integral distinguishers, because of zero values in the cube attack, this attack mounts the distinguisher to more rounds as compared to the case in which non-cube bits take random values. Hence, the cube attack generates distinguishers of higher rounds in comparison with [21] for the same input values as active bits.

It is worth noting that unlike the stream cipher in which a zero-sum integral distinguisher is non-trivial to recover secret variables, in the block cipher, it allows using it as a distinguisher and extend this intermediate round to the final round of the cipher under some key guess. Then, in the online phase, the distinguisher is not achieved in wrong key guess, but in the correct key guess, after some backward partial decryption, the distinguisher is obtained and the key guess is considered as a key candidate.

As discussed in [24], the division property puts an upper bound on the key bits involved in the superpoly. Thus, by determining the existence and non-existence of the key bits in the superpoly, as noted in Section 3.2, the neutrality property of the cube tester can be extracted [9]. In this property, the presence of key bits in at least one monomial of the superpoly was analysed and the key bits not involved in the superpoly were considered as neutral key bits. Accordingly, superpoly was independent of these key bits, which was one of its weak points.

In [35], it has been shown that determining key bits without significant influence on the value of coefficients in the ANF representation of a polynomial function can be utilised to derive information about the key in a way that is faster than exhaustive key search in approximation approaches. In Table 4, the key bits that are not involved in the superpoly are presented as neutral key bits for the cube sizes 30 and 31 under best results derived from Table 9.

5 Conclusion and future work

In this study, the division property in the cube distinguisher extraction was employed for block ciphers. Automatic tools for identifying integral distinguishers simplified the procedure of finding the division trail. Hence, we focused on presenting an automated and efficient tool for finding division property trail using SAT solvers. We converted the problem of finding a division trail to a SAT problem by defining initial division property, propagation rules in the CNF form and other conditions to determine the existence of balanced bits based on the feasibility or infeasibility of the SAT problem.

This approach was then extended and adapted to find cube distinguishers for block ciphers in a non-blackbox manner. To do this, the tool was adapted considering the differences between integral and cube attacks. Compared to the utilisation of division property in the cube distinguisher extraction for stream ciphers, here we considered the differences between the stream and block cipher structures in the propagation of division property through rounds to find the cube distinguisher. In conclusion, we proved the correctness and efficiency of the proposed approach by applying it to block cipher KATAN. The proposed approach improved the results of the best existing cube attack on the block cipher KATAN.

6 Acknowledgments

We are grateful to our colleagues Stefan Kolbol and Tyge Tiesson from Technical University of Denmark (DTU) for their guidance, insights, and expertise, which wereof great assistance to the research. Also, we thank the reviewers for their so-called insights and constructive comments.

7 References

[1] Garey, M.R., Johnson, D.S.: '*Computers and intractability: a guide to the theory of NP-completeness*' (W.H. Freeman, Gordonsville, VA, USA, 1979)

[2] Cid, C., Leurent, G.: 'An analysis of the XSL algorithm'. 2005 Proc. 11th Int. Conf. on the Theory and Application of Cryptology and Information Security, Advances in Cryptology – (ASIACRYPT 2005), Chennai, India, 4–8 December 2005, pp. 333–352. Available at https://doi.org/10.1007/11593447_18

[3] Courtois, N.: 'Higher order correlation attacks, XL algorithm and cryptanalysis of toyocrypt'. 5th Int. Conf. on Information Security and Cryptology (ICISC 2002), Seoul, Korea, 28–29 November 2002, pp. 182–199. Revised Papers, 2002. Available at https://doi.org/10.1007/3-540-36552-4_13

[4] Courtois, N., Pieprzyk, J.: 'Cryptanalysis of block ciphers with overdefined systems of equations'. 2002 Proc. 8th Int. Conf. on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2002), Queenstown, New Zealand, 1–5 December 2002, pp. 267–287. Available at https://doi.org/10.1007/3-540-36178-2_17

[5] Courtois, N.T., Sepehrdad, P., Susil, P., *et al.*: 'Elimlin algorithm revisited'. 19th Int. Workshop, Fast Software Encryption (FSE 2012), Washington, DC, USA, 19–21 March 2012, pp. 306–325. Revised Selected Papers, 2012. Available at https://doi.org/10.1007/978-3-642-34047-5_18

[6] Faugère, J., Joux, A.: 'Algebraic cryptanalysis of hidden field (HFE) cryptosystems using Gröbner bases'. 2003 Proc. 23rd Annual Int. Cryptology Conf. on Advances in Cryptology (CRYPTO 2003), Santa Barbara, California, USA, 17–21 August 2003, pp. 44–60. Available at https://doi.org/10.1007/978-3-540-45146-4_3

[7] Mironov, I., Zhang, L.: 'Applications of SAT solvers to cryptanalysis of hash functions'. 2006 Proc. 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2006), Seattle, WA, USA, 12–15 August 2006, pp. 102–115. Available at https://doi.org/10.1007/11814948_13

[8] Dinur, I., Shamir, A.: 'Cube attacks on tweakable black box polynomials'. 2009 Proc. 28th Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2009), Cologne, Germany, 26–30 April 2009, pp. 278–299. Available at https://doi.org/10.1007/978-3-642-01001-9_16

[9] Aumasson, J., Dinur, I., Meier, W., *et al.*: 'Cube testers and key recovery attacks on reduced-round MD6 and trivium'. 16th Int. Workshop on Fast Software Encryption (FSE 2009), Leuven, Belgium, 22–25 February 2009, pp. 1–22. Revised Selected Papers 2009. Available at https://doi.org/10.1007/978-3-642-03317-9_1

[10] Dinur, I., Shamir, A.: 'Breaking grain-128 with dynamic cube attacks'. 18th Int. Workshop Fast Software Encryption (FSE 2011), Lyngby, Denmark, 13–16 February 2011, pp. 167–187. Revised Selected Papers 2011. Available at https://doi.org/10.1007/978-3-642-21702-9_10

[11] Vielhaber, M.: 'Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack', IACR Cryptology ePrint Archive, 2007, 2007, p. 413. Available at http://eprint.iacr.org/2007/413

[12] Bard, G.V., Courtois, N.T. Jr., Sepehrdad, P., *et al.*: 'Algebraic, AIDA/CUBE and side channel analysis of Katan family of block ciphers'. 2010 Proc. 11th Int. Conf. on Cryptology Progress in Cryptology in India (INDOCRYPT 2010), Hyderabad, India, 12–15 December 2010, pp. 176–196. Available at https://doi.org/10.1007/978-3-642-17401-8_14

[13] Rabbaninejad, R., Ahmadian, Z., Salmasizadeh, M., *et al.*: 'Cube and dynamic cube attacks on simon32/64'. 2014 11th Int. ISC Conf. on Information Security and Cryptology, Tehran, Iran, 2014, pp. 98–103

[14] Ahmadian, Z., Rasoolzadeh, S., Salmasizadeh, M., *et al.*: 'Automated dynamic cube attack on block ciphers: cryptanalysis of SIMON and Katan'. IACR Cryptology ePrint Archive, 2015, 2015, p. 40. Available at http://eprint.iacr.org/2015/040

[15] Dinur, I., Shamir, A.: 'Side channel cube attacks on block ciphers'. IACR Cryptology ePrint Archive, 2009, 2009, p. 127. Available at http://eprint.iacr.org/2009/127

[16] Knudsen, L.R., Wagner, D.: 'Integral cryptanalysis'. Int. Workshop on Fast software encryption (FSE 2002), Berlin, Germany, 2002, (LNCS, 2365), pp. 112–127
[17] Lai, X.: 'Higher order derivatives and differential cryptanalysis', in Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (Eds.): '*Communications and cryptography*' (Springer, Boston, MA, USA, 1994), pp. 227–233. Available at https://doi.org/10.1007/978-1-4615-2694-0_23
[18] Todo, Y.: 'Structural evaluation by generalized integral property'. 2015 Proc., Part I 34th Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2015), Sofia, Bulgaria, 26–30 April 2015, pp. 287–314. Available at https://doi.org/10.1007/978-3-662-46800-5_12
[19] Todo, Y., Morii, M.: 'Bit-based division property and application to Simon family'. 23rd Int. Conf. on Fast Software Encryption (FSE 2016), Bochum, Germany, 20–23 March 2016, pp. 357–377. Revised Selected Papers, 2016. Available at https://doi.org/10.1007/978-3-662-52993-5_18
[20] Xiang, Z., Zhang, W., Bao, Z., *et al.*: 'Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers'. 2016 Proc., Part I 22nd Int. Conf. on the Theory and Application of Cryptology and Information Security, Advances in Cryptology – (ASIACRYPT 2016), Hanoi, Vietnam, 4–8 December 2016, pp. 648–678. Available at https://doi.org/10.1007/978-3-662-53887-6_24
[21] Sun, L., Wang, W., Liu, R., *et al.*: 'MILP-aided bit-based division property for ARX-based block cipher'. IACR Cryptology ePrint Archive, 2016, 2016, p. 1101. Available at http://eprint.iacr.org/2016/1101
[22] Sun, L., Wang, W., Wang, M.: 'MILP-aided bit-based division property for primitives with non-bit-permutation linear layers'. IACR Cryptology ePrint Archive, 2016, 2016, p. 811. Available at http://eprint.iacr.org/2016/811
[23] Sun, L., Wang, W., Wang, M.: 'Automatic search of bit-based division property for ARX ciphers and word-based division property'. 2017 Proc., Part I 23rd Int. Conf. on the Theory and Applications of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2017), Hong Kong, China, 3–7 December 2017, pp. 128–157. Available at https://doi.org/10.1007/978-3-319-70694-8_5
[24] Todo, Y., Isobe, T., Hao, Y., *et al.*: 'Cube attacks on non-blackbox polynomials based on division property'. 2017 Proc., Part III 37th Annual Int. Cryptology Conf. on Advances in Cryptology (CRYPTO 2017), Santa Barbara, CA, USA, 20–24 August 2017, pp. 250–279. Available at https://doi.org/10.1007/978-3-319-63697-9_9
[25] Wang, Q., Hao, Y., Todo, Y., *et al.*: 'Improved division property based cube attacks exploiting algebraic properties of superpoly'. IACR Cryptology ePrint Archive, 2017, 2017, p. 1063. Available at http://eprint.iacr.org/2017/1063
[26] Rohit, R., AlTawy, R., Gong, G.: 'MILP-based cube attack on the reduced-round WG-5 lightweight stream cipher'. 2017 Proc. 16th IMA Int. Conf. on Cryptography and Coding (IMACC 2017), Oxford, UK, 12–14 December 2017, pp. 333–351. Available at https://doi.org/10.1007/978-3-319-71045-7_17
[27] Cannière, C.D., Dunkelman, O., Knezevic, M.: 'KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers'. 2009 Proc. 11th Int. Workshop Cryptographic Hardware and Embedded Systems (CHES 2009), Lausanne, Switzerland, 6–9 September 2009, pp. 272–288. Available at https://doi.org/10.1007/978-3-642-04138-9_20
[28] Blum, M., Luby, M., Rubinfeld, R.: 'Self-testing/correcting with applications to numerical problems', *J. Comput. Syst. Sci.*, 1993, **47**, (3), pp. 549–595. Available at http://dx.doi.org/10.1016/0022-0000(93)90044-W
[29] Sun, S., Gerault, D., Lafourcade, P., *et al.*: 'Analysis of AES, skinny, and others with constraint programming', *IACR Trans. Symmetric Cryptol.*, 2017, **2017**, (1), pp. 281–306. Available at https://doi.org/10.13154/tosc.v2017.i1.281-306
[30] Eskandari, Z., Kidmose, A.B., Kölbl, S., *et al.*: 'Finding integral distinguishers with ease'. 25th Int. Conf. on Selected Areas in Cryptography (SAC 2018), Calgary, AB, Canada, 15–17 August 2018, pp. 115–138. Revised Selected Papers, 2018. Available at https://doi.org/10.1007/978-3-030-10970-7_6
[31] Mroczkowski, P., Szmidt, J.: 'The cube attack on stream cipher Trivium and quadraticity tests'. IACR Cryptology ePrint Archive, 2010, 2010, p. 580. Available at http://eprint.iacr.org/2010/580
[32] Rasoolzadeh, S., Raddum, H.: 'Multidimensional meet in the middle cryptanalysis of Katan'. IACR Cryptology ePrint Archive, 2016, 2016, p. 77. Available at http://eprint.iacr.org/2016/077
[33] Sanders, J., Kandrot, E.: '*CUDA by example: an introduction to general purpose GPU programming*' (Addison-Wesley, Boston, MA, USA, 2011). Available at http://www.worldcat.org/oclc/699702402
[34] Fu, X., Wang, X., Chen, J.: 'Determining the nonexistent terms of non-linear multivariate polynomials: how to break grain-128 more efficiently'. IACR Cryptology ePrint Archive, 2017, 2017, p. 412. Available at http://eprint.iacr.org/2017/412
[35] Fischer, S., Khazaei, S., Meier, W.: 'Chosen iv statistical analysis for key recovery attacks on stream ciphers', in Vaudenay, S., (ed.): *Progress in Cryptology - AFRICACRYPT 2008*, (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), pp. 236–245

8 Appendix

8.1 Toy example

Let us consider an initial state of three bits (x_0, x_1, x_2). We are interested to check the existence or non-existence of term x_0x_2 in the output. So, the initial division property is (1, 0, 1). In Table 5, in each row, the states were updated under the application of the operation in the first column. The new values for division property were calculated based on the propagation rules mentioned in Section 3.1.1.

As seen in the last row of the second column, the two vectors $\{(\mathbf{1}, \mathbf{0}, \mathbf{1}), (\mathbf{0}, \mathbf{0}, \mathbf{1})\}$ have hamming weight 1. Based on description in Section 3.1.2, it means that the term x_0x_2 exists at the second and the last bits of the output, respectively. We can verify the obtained result from the division property propagation with the ANF representation in the last row of the first column.

As we described in Section 3.2, in cube attacks, we have $I = \{x_0, x_2\}$ and other inputs, x_1 has zero value. As discussed in division propagation for cube attacks, if one operand of the AND operation was zero, the propagation is ignored. So, the second row of the third column has a different value. In the end, as seen in the last row of the third column, the only $\{(\mathbf{0}, \mathbf{0}, \mathbf{1})\}$ vector has hamming weight 1, so the term $x_0 \cdot x_1 \cdot x_2$ at the second bit of the output is ignored because of $x_1 = 0$ and the term x_0x_2 exists only at the last bit of the output.

8.2 Extracted linear equations

Here the details of linear equations extracted for 72-round and 90-round KATAN32 and 50-round KATAN48, as discussed in Section 4.1, are presented (see Tables 6–8).

8.3 Extracted distinguishers

As discussed in Section 4.2, different scenarios were tried for the same cube size and different zero positions to achieve constant distinguishers in higher round numbers. The details of these experiments are presented here (see Table 9).

Table 5 Example of division property propagation

Operation	States	Div. Prop. values in integral attack	Div. Prop. values in cube attack
—	(x_0, x_1, x_2)	{(1, 0, 1)}	{(1, 0, 1)}
$x_1 = x_1 \,.\, x_2$	$(x_0, x_1 \cdot x_2, x_2)$	{(1, 0, 1), (1, 1, 0)}	{(1, 0, 1)}
$x_2 = x_1 \oplus x_2$	$(x_0, x_1 \cdot x_2, x_1 \cdot x_2 \oplus x_2)$	{(1, 0, 1), (1, 1, 0)}	{(1, 0, 1), (1, 1, 0)}
$x_1 = x_0 \,.\, x_1$	$(x_0, x_0 \cdot x_1 \cdot x_2, x_1 \cdot x_2 \oplus x_2)$	{(1, 0, 1), (0, 1, 1), (0, 1, 0)}	{(1, 0, 1), (0, 0, 1)}
$x_2 = x_0 \,.\, x_2$	$(x_0, x_0 \cdot x_1 \cdot x_2, x_0 \cdot x_1 \cdot x_2 \oplus x_0 \cdot x_2)$	{(**0**, **0**, **1**), (0, 1, 1), (0, 1, 0)}	{(0, 0, 1), (0, 1, 1)}

$\oplus$ and $\cdot$ stand for *XOR* and *AND* operation, respectively.

Table 6 Extracted linear equations for 72-round KATAN32

Maxterm	Degree	Equation	Cipher bit
FFFFEFFA	29	$k1 + k20 + k31 + k68 + k72$	c7
FFFFFFD5	29	$k67 + k78$	c4
FFFFFFBA	29	$k6 + k25 + k36 + k73 + 1$	c23
FFFFFF75	29	$k1 + k3 + k16 + k20 + k22 + k31 + k33 + k35 + k46 + k68 + k70$	c5
		$k4 + k23 + k34 + k71 + 1$	c24
FFFFFEFA	29	$k74 + k78$	c4
		$k4 + k23 + k34 + k71 + 1$	c24
FFFFFDF5	29	$k2 + k21 + k32 + k69 + 1$	c25
FFFFFBFA	29	$k74 + k78 + 1$	c24
FFFFF7F5	29	$k72 + k76$	c25
FFFBFFFA	29	$k56 + k57 + k58 + k68 + k72$	c30
FEFFFFFA	29	$k42 + k44 + k52$	c30
F7FFFFFA	29	$k2 + k21 + k32 + k45 + k46 + k48 + k58 + k62 + k69 + k75 + 1$	c12
FFFFFFE5	29	$k65 + k75 + k77$	c25
FFFDFFFA	29	$k45 + k54 + k55 + k56 + k66 + 1$	c12
FFFFBFFA	29	$k0 + k19 + k30 + k61 + k67 + k76$	c8
FFFFDFFA	29	$k57 + k67 + k69$	c29
FFFFFFF1	29	$k57$	c5
FFFFEFF5	29	$k47 + k68 + k69 + k72 + k77 + 1$	c9
F7FFFFF5	29	$k36$	c13
FDFFFFF5	29	$k40 + k44 + 1$	c12
		$k40 + k42 + k50 + k70 + 1$	c31
FFBFFFF5	29	$k62 + k66$	c30
FBFFFFFF	31	$k0 + k2 + k6 + k19 + k21 + k25 + k30 + k32 + k36 + k60 + k67 + k69 + k73 + k74$	c21
		$k76$	c22
7FFFFFFF	31	$k0 + k19 + k30 + k50 + k64 + k67 + k70 + k72 + k76 + 1$	c26
		$k66$	c27
BFFFFFFF	31	$k52 + k66 + k72 + k74 + k78 + 1$	c25
DFFFFFFF	31	$k0 + k19 + k30 + k54 + k67 + k68 + k74 + k76 + 1$	c24
F7FFFFFF	31	$k74 + 1$	c23
FEFFFFFF	31	$k4 + k6 + k10 + k23 + k25 + k29 + k34 + k36 + k40 + k64 + k71 + k73 + k77 + k78$	c19
FFFBFFFF	31	$k3 + k10 + k22 + k29 + k33 + k40 + k70 + k75 + k77$	c8
		$k78$	c27
FFFDFFFF	31	$k5 + k12 + k24 + k31 + k35 + k42 + k72 + k77 + k79 + 1$	c7
		$k0 + k19 + k30 + k67$	c26
FFFEFFFF	31	$k1 + k14 + k20 + k31 + k33 + k44 + k68 + k79$	c6
		$k2 + k21 + k32 + k69 + 1$	c25
FFFF7FFF	31	$k1 + k3 + k16 + k20 + k22 + k31 + k33 + k35 + k46 + k68 + k70$	c5
		$k4 + k23 + k34 + k71 + 1$	c24
FFFFBFFF	31	$k6 + k25 + k36 + k73 + 1$	c23
FFFFDFFF	31	$k8 + k27 + k38 + k75 + 1$	c22
FFFFEFFF	31	$k10 + k29 + k40 + k77$	c21
FFFFF7FF	31	$k12 + k31 + k42 + k79$	c20
FFFFFBFF	31	$k1 + k14 + k20 + k31 + k33 + k44 + k68 + 1$	c19

Table 7 Extracted linear equations for 90-round KATAN32

Maxterm	Degree	Equation	Cipher bit
FFFFFFFB	31	$k4 + k17 + k23 + k30 + k34 + k36 + k47 + k49 + k60 + k71 + 1$	C29
FFFFFFF7	31	$k2 + k15 + k21 + k28 + k32 + k34 + k45 + k47 + k58 + k69$	C30
FFFFFFEF	31	$k0 + k13 + k19 + k26 + k30 + k32 + k43 + k45 + k56 + k67 + 1$	C31

Table 8 Extracted linear equations for 50-round KATAN48

Maxterm	Degree	Equation	Cipher bit
01248FEDBFE7	27	$k8 + 1$	c27
FEFFF8B1D001	28	$k3$	c20
		$k9$	c21
FFF381000FFF	28	$k1$	c28
5E266F3E9A37	28	$k8 + k13 + k19 + 1$	c27
5E646F3E9A37	28	$k5 + k12 + 1$	c25
		$k4 + k10 + k13 + k19$	c27
1F666F3E9A37	29	$k7 + k24$	c27
FFF0032E5A7F	29	$k12$	c27
1F40377C1FFF	29	$k2$	c27
FFF3C0003FFF	30	K14 + $k31 + 1$	c28
FC051EE31FFF	30	$k6 + k16$	c27
		$k6$	c28
FFFF08B1D347	30	$k9 + k15 + 1$	c23
5F666F6E9A37	30	$k4 + k5 + k12 + 1$	c25
FFFF08B1E347	30	$k11 + 1$	c25
5F666FAE9A37	30	$k8 + k12 + k14$	c26
42BF16AFFFF0	30	$k18 + k24$	c28
CF04377C1FFF	30	$k2$	c27
FFE456460FFF	31	$k5 + k24$	c28
FFF5C0B6637E	31	$k7$	c27
FFFC03B6637E	31	$k7 + k15 + k22$	c25
2D6D9DB9BF3D	31	$k0 + k20$	c24

Table 9 Extracted distinguishers at cipher bit C_{18} of KATAN32

Cube size	Zero bits positions	Rounds	Ref.
30	17,18	74	[14]
	0,1	90	[21]
	3,10	84	
	8,12	82	
	0,1	**91**	
	3,24	81	section 4.2
	8,24	80	
	10,27	76	
	12,27	76	
31	18	83	[14]
	0	99	[21]
	0	**101**	
	1	100	
	3	98	
	8	93	section 4.2
	10	91	
	12	89	
	24	90	
	27	88	

The bold rows present the best results for the same cube size but in different zero bits positions.