The ISC Int'l Journal of Information Security

ISeCure

January 2020, Volume 12, Number 1 (pp. 1–12)
http://www.isecure-journal.org

Extension of Cube Attack with Probabilistic Equations and its Application on Cryptanalysis of KATAN Cipher

Zahra Eskandari [1], and Abbas Ghaemi Bafghi [1,*]

[1] *Data and Communication Security Lab., Computer Dept., Ferdowsi University of Mashhad, Iran*

ARTICLE INFO.

Article history:
Received: 28 August 2019
Revised: 14 December 2019
Accepted: 20 January 2020
Published Online: 30 January 2020

Keywords:
Approximation, Cube Attack, Probabilistic Equation, Solving Noisy Equation System.

Abstract

Cube Attack is a successful case of Algebraic Attack. Cube Attack consists of two phases, linear equation extraction and solving the extracted equation system. Due to the high complexity of the equation extraction phase in finding linear equations, we can extract nonlinear ones that could be approximated to linear equations with high probability. The probabilistic equations could be considered as linear ones under some noises. Existing approaches to solving noisy equation systems work well provided that the equation system has a low error rate; however, as the error rate increases, the success rate of finding the exact solution diminishes, making them rather inefficient in high error rate. In this paper, we extend Cube Attack to probabilistic equations. First, an approximation approach based on linear combinations of nonlinear equations is presented to find probabilistic linear equations with high probability. Then, we present an approach to improve the efficiency of current solving approaches and make them practical to solve a high error rate linear equation system. Finally, utilizing proposed approaches, we find the right key under an extended noisy equation system with lower complexity in comparison to the original Cube Attack.

1 Introduction

In the algebraic cryptanalysis of symmetric ciphers, based on the round function of the cipher, a large nonlinear polynomial system is generated. There are several approaches to solve such systems [1–3], but due to the high memory consumption and time complexity, it is infeasible to apply them to practical problems. A successful case of an algebraic attack is the cube attack [4], which was introduced in 2009. In this attack, instead of generating the equation system, some linear equations are extracted, and the right key value was obtained by solving the extracted equations.

The primary challenge to the cube attack is finding linear equations, which is conducted in some heuristics and is therefore highly complex. Hence, in some works [5], instead of finding linear equations, nonlinear equations with a linear approximation and high probability are extracted and regarded as linear ones influenced by some noises.

Two different approaches have been presented to solve the system of probabilistic equations known as the noisy equation system. In the first approach, after approximating the round function or using some annihilators to reduce its degree [6], the equation system

* Corresponding author.

Email addresses: zahra.eskandari@mail.um.ac.ir, ghaemib@um.ac.ir

ISSN: 2008-2045 © 2020 ISC. All rights reserved.

is generated. Then the correct solution is identified through repetition the solving with different subsets of key stream bits [1] or a key with the predetermined probability of correctness is determined [7].

In another approach, which is more popular today, the solving of a noisy equation system is modeled as the MAX-PoSSo problem [8]. This problem is a variant of Polynomial System Solving (PoSSo), which involves finding a solution to a system of polynomial equations. MAX-PoSSo, or the problem of solving polynomial equations with noises, involves finding the solution that satisfies the maximum number of polynomials. Obviously, MAX-PoSSo is at least as hard as PoSSo. Moreover, irrespective of the linearity of polynomials, MAX-PoSSo is an NP-hard problem [9].

One of the primary approaches based on this concept is [8], in which a method based on optimization approaches is recruited to solve such problems. The problem was converted to a Mixed Integer Linear Programming (MILP) problem by defining some variables and constraints over integers and employing some conversion methods to handle Boolean calculations [10]. Moreover, the authors convert a Partial Weighted MAX-PoSSo problem into a Mixed Integer Programming problem by defining some new slack variables and an objective function corresponding to a cost function, which should be minimized.

In the same vein, authors in [11] proclaimed that some of the equations containing information about the noisy sequence of cipher were not necessarily valid while all other equations in the system were satisfied. The problem of solving a noisy polynomial system was then modeled as a Mixed Integer Linear Programming problem, and they were able to recover the secret key with a success rate of more than 90%.

In [5], cube attacks were extended to a wider scope, and probabilistic linear equations were utilized. To solve the probabilistic equation system, the authors used maximum likelihood decoding (MLD) in the online phase. However, despite the exponential complexity, their contribution could not achieve significant results to extract the correct key under all scenarios, and its success rate was about 15%.

Recently [9], Incremental Solving, and Backtracking Search (ISBS) have been proposed to solve Boolean equation systems with noises. In comparison to other approaches that search all possible values of variables, ISBS searches all possible values of noises with backtracking and incrementally solving using the MFCS method [12]. It has been shown that it was much more efficient than the optimization approaches [8]. Further details of this approach are presented later in this paper.

Unfortunately, all of these approaches work well at a low error rate of the equation system, but as the error rate rises, the success rate of determining the right solution aggravate. Hence, at high error rate, these methods are inefficient and impractical.

Our Contribution: In this work, at first, we present an approximation approach based on linear combinations of nonlinear equations to find probabilistic linear equations with high probability. Then, the problem of solving a high error rate and determined noisy equation system is considered. By determined, we mean an equation system with almost the same number of variables and equations, contrary to the assumption of existing approaches where the equation system is overdetermined and has a low error rate. As a result, these methods were unable to achieve satisfactory results in our problem. They failed to find the right key under most scenarios, and a low success rate was reported. To overcome this drawback, considering the success of the ISBS method, we present an approach to improve its efficiency at a high error rate. This improved approach is then applied to solve the extended cube equation system, including deterministic and probabilistic equations, and we can find the right key with lower complexity in comparison with the application of only deterministic equations.

Organization: The remainder of this paper is organized as follows. In Section 2, we briefly review preliminary concepts such as cube attack, ISBS method, and KATAN cipher. In Section 3, we focus on our contribution and describe the proposed attack scenario in detail. Section 4 presents the results. Conclusions and future works are drawn in Section 5. Some details of results are provided in the Appendix.

2 Preliminaries

In this section, the basic concepts of this paper have been outlined. A brief overview of the cube attack is given followed by the specifications of ISBS method. Finally, the cipher specifications are described while presenting the results over KATAN block cipher.

2.1 Cube Attack

In 2009, Cube attack [4] was proposed as a type of algebraic attack. This attack employs the Algebraic Normal Form (ANF) representation of the ciphertext bit as a polynomial function of plaintext and key bits. By evaluating this function for all possible values of some plaintext bits, some linear equations of key bits are extracted. Each ciphertext bit is considered as a polynomial function of plaintext bits, $V = \{v_0, v_1, \ldots\}$, and key bits, $K = \{k_0, k_1, \ldots\}$. Assuming that $x = V \bigcup K$, one can write any of ciphertext bits as a polynomial $p(x)$ over F_2 in ANF representation. Let I be a subset of plaintext bits and t_I be the product of all variables whose indexes are in I. Accordingly, it can be written as:

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

Where t_I and $Ps(I)$ are cube and superpoly, respectively. $Ps(I)$ is a polynomial of key bits and remaining plaintext bits. $q(x)$ is the sum of all terms that lack at least one term of I. According to [4], $Ps(I)$ can be calculated by applying a higher-order derivative:

$$Ps(I) \equiv \sum_{v \in C_I} p(x)(mod2) \tag{2}$$

where C_I represents the set of all possible values for variables in t_I and other bits excluded from I are zero. Therefore, $Ps(I)$ is only a polynomial of key bits. If it is a linear polynomial, while the corresponding t_I is called a maxterm, it can be utilized as an equation under the 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 provide linear superpoly. The linear superpoly can be determined by performing some linearity tests [13]. The primary challenge of the cube attack is to find linear equations. This phase is carried out in some heuristics such as the random walk manner [4], which consequently causes high complexity.

In the online phase, the right hand side of the equation is evaluated for all possible values of maxterm variables with a fixed but unknown key, and the obtained equation system is solved using Gaussian elimination. If the extracted equations are less than the key length, the remained key bits are determined by an exhaustive search. By doing so, the values of key bits can be finally recovered.

The complexity of the cube attack is as follows: Assume m linear equations are extracted with a cube size of d for a key length of n. The complexity of offline phase is $m.(n+1).2^d$, which corresponds to calculation of the coefficient of $n+1$ linear terms, including the constant term for m equations with a cube size of d. The complexity of the online phase is bounded by $m.2^d + m^3 + C.2^{(n-m)}$. The first term calculates the right hand side of m linear equations. The second one is for solving the system using Gaussian elimination and the last one is for determining the remained key bits using an exhaustive search where term C is the cost for checking a key.

The most important extensions of the cube attack are cube testers [14], which search for non-random distinguishers and dynamic cube attacks [15] which use cube tester distinguishers for the key recovery.

2.2 ISBS

ISBS is a new method of solving MAX-PoSSo problems over F_2. The main idea of ISBS is incremental solving and backtracking of all the possible noises. Consider a noisy polynomial system $\{f_1, f_2, \ldots, f_m\}$. The aim of ISBS is to solve polynomial systems $\{f_1+e_1, f_2+e_2, \ldots, f_m+e_m\}$ with the noise vector $(e_1, e_2, \ldots, e_m)$ that has the smallest Hamming weight as the solution of the MAX-PoSSo problem. The noise vector can be equal to $(0,0,...,0),(1,0,...,0),...,(1,1,...,1)$.

Since the ISBS method works incrementally, if the partial polynomial system $\{f_1+e_{01}, f_2+e_{02}, \ldots, f_k+e_{0k}\}$ is a contradiction system without any solution for some fixed $(e_{01}, e_{02}, \ldots, e_{0k})$, it does not need to solve any system with the form $\{f_1+e_{01}, \ldots, f_k+e_{0k}, f_2+e_{0k+1}, \ldots, f_m+e_{0m}\}$ for every possible value of $(e_{k+1}, \ldots, e_m)$. In this way, many search branches can be pruned and this kind of redundant computations could be avoided. Indeed, ISBS merges the incremental solving and the backtracking search methods with the above idea.

The ISBS works as follows [9]:

(i) Equation system $\{f_1+e_1, f_2+e_2, \ldots, f_i+e_i\}$ is incrementally solved for i from 1 to m with each $e_i = 0$. If the system has no solution for some i, e_i is flipped to 1 and solving the remaining polynomials is continued based on the solution set for $\{f_1+e_1, f_2+e_2, \ldots, f_i+1\}$. Finally, a candidate solution is obtained from solving $\{f_1+e_1, f_2+e_2, \ldots, f_m+e_m\}$ where $(e_1, e_2, \ldots, e_m)$ is equal to some fixed $(e_{01}, e_{02}, \ldots, e_{0m})$. The Hamming weight of the obtained noise vector is considered as the upper bound of the noise.

(ii) To obtain a more suitable candidate, all possible values of the noise vector are searched with backtracking based on the value $(e_{01}, e_{02}, \ldots, e_{0m})$. That is, for i from k to 1 the first e_{0i} is found such that $e_i = 0$ and the Hamming weight of $(e_1, \ldots, e_{i-1}, 1)$ is less than the specified upper bound. Then, similar to step (i), $\{f_{i+1}, \ldots, f_m\}$ is solved incrementally. If a more suitable candidate is found, k is set to m, $(e_{01}, e_{02}, \ldots, e_{0m})$ and upper bound are replaced with $(e_{01}, \ldots, e_{0i-1}, 1, e_{00i+11}, \ldots, e_{00m})$ and the hamming weight of new noise vector, respectively. step (ii) is repeated again.

(iii) Finally, all the possible $(e_1, e_2, \ldots, e_m)$ are searched and the optimal solution is obtained.

In other words, by tracking all non-contradict branches in ISBS, the branch with minimum noise is found as the solution for the MAX-PoSSo problem. The theoretical complexity of ISBS is exponential,

Table 1. Parameters used in non-linear functions.

Cipher	$\lvert L1\rvert$	$\lvert L2\rvert$	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

but due to the pruning of many search branches, the practical complexity is significantly lower [9]. It is worth noting that since the height of search tree is dependent on the number of probabilistic equations, considering the complexity of partial solving and saving the inner solution sets at high error rate, the ISBS method was not efficient at error rates exceeding 0.08 [9].

2.3 KATAN Cipher

KATAN is a family of lightweight ARX block ciphers [16]. It has three variants of 32, 48 or 64-bit block size with an 80-bit main key and 254 rounds in all versions. KATAN consists of two linear feedback shift registers (LFSRs) called L1 and L2, which are initialized with the plaintext. They are then transformed by two non-linear Boolean functions as follows:

$$f_a(L1) = L1[x_1] \oplus L1[x_2] \oplus L1[x_3] \cdot L1[x_4] \oplus L1[x_5] \cdot IR \oplus k_a$$
$$f_b(L2) = L2[y_1] \oplus L2[y_2] \oplus L2[y_3] \cdot L2[y_4] \oplus L2[y_5] \cdot L2[y_6] \oplus k_b$$

IR consists of the output of an LFSR, which is considered as some constants in the cipher. In each round, LFSRs are left-shifted, and the output of each function is loaded on the least significant bits of the other LFSR. This operation is invertible. The parameters of the above function are presented in Table 1. For the i-th round, only two key bits are used. For KATAN48 and KATAN64, these functions are applied two and three times, respectively, for each round with the same pair of subkey bits. The key schedule algorithm for all KATAN ciphers is a linear mapping that expands an 80-bit key K into $2 \times 254 = 508$ subkey bits. See [16] for more details.

3 Description of The Proposed Attack Scenario

In this section, the proposed approach for extending the cube attack by utilizing probabilistic equations is proposed. Similar to the cube attack, the proposed attack scenario consists of two phases, as described later.

3.1 Equation Extraction Phase

As described earlier, the preprocessing phase of the cube attack to choose proper maxterms is highly complex. To overcome this problem, we used the approach presented in [17, 18], which is based on the division property. Division property was proposed by [19] in 2015 to extract integral distinguishers with a high success in progressing the results of this attack type [20–22]. Given the similarity of Integral and Cube attack, in [17, 18] division property was adapted to extract cube distinguishers with lower complexity, and some improvements were reported.

Using the adapted division property, we can determine the upper bound of involved key bits in the output bit of a cipher after a few rounds. Based on this information, we can choose output bits that depend on the low number of key bits as proper options to extract linear equations. As the coefficient of not involved key bits, i.e., neutral key bits, are zero in the superpoly, these bits are eliminated from the process of calculating coefficients, and hence the complexity of equation extraction phase is diminished.

In brief, to check the existence or nonexistence of a key bit, k_i, in the superpoly, this bit is activated in the initial vector of division property in addition to active plaintext bits, which are activated matching the maxterm. Then the initial value is propagated through rounds of the cipher based on the propagation rules. In the end, generated vectors in the propagation procedure are analyzed, and the coefficient of this specific key bit in the superpoly is determined. For more details, interested readers can refer to [18].

Here, we extend the proposed approach in [18] to extract nonlinear equations specifically quadratic ones. Assume set I as involved key bits in a superpoly with n_I element. To determine the quadratic terms in the superpoly, for $\forall i, j \in I, i < j$, the existence of the quadratic term $k_i k_j$ is checked via 1) activating these key bits in addition to corresponding bits for active plaintext bits in the initial value of division property, 2) propagating it along the rounds of the cipher and finally, 3) analyzing the final values. By doing so for all involved key bit pairs, the coefficient matrix Q is extracted as follows:

$$Q = \begin{bmatrix} coeff\,(k_{i1}k_{i2}) & coeff\,(k_{i1}k_{i3}) & \dots & coeff\,(k_{i1}k_{in_I}) \\ & coeff\,(k_{i2}k_{i3}) & \dots & coeff\,(k_{i2}k_{in_I}) \\ & & & \vdots \\ & & & coeff\left(k_{in_I-1}k_{in_I}\right) \end{bmatrix}.$$

for $\forall ix \in I, 1 \leq x \leq n_I$, $q_{ix,iy}$ present the coefficient of term $k_{ix}k_{iy}$ in the superpoly.

In continue, set I_x for $\forall ix \in I$ as: $I_{ix} = \{iy \mid ix < iy, q_{ix,iy} \neq 0\}$ is calculated. Indeed, set I_{ix} represents the index of key bits which are involved in the superpoly in quadratic form as a multiplier of k_{ix}. Based on the set I_{ix}, we can determine the degree of the terms as follows:

(1) if $|I_{ix}|=0$, since $ix \in I$ and $\forall y \in I \mid q_{ix,y}=0$, it means that the key bit k_{ix} does not appear in none of the quadratic terms and hence, this key bit is a linear term in the superpoly.
(2) if $|I_{ix}|=1$, it means that the term $k_{ix}k_{iy}$ is existed in the superpoly. Ensuring of the existence or non-existence of the linear terms k_{ix} and k_{iy}, the coefficient of them are calculated too.
(3) if $|I_{ix}|\geq 2$, it means that there are at least two quadratic terms which include these key bits or there is a cubic term $k_{ix}k_{iy}k_{iz}$. Consequently, the degree cannot be guessed deterministically.

After determining the superpolies with $|I_{ix}|\leq 1| \forall ix \in I$, some quadratic equations are extracted, which are presented in the Appendix. It should be mentioned that solving the multivariate quadratic equation system is considered as an NP-hard problem. Some linearization approaches [1, 23] were presented to solve such systems. In these approaches, a new variable is defined for every quadratic term. Because of the existence of too many quadratic terms in extracted quadratic equations, these approaches have high complexity due to a high number of new variables.

Here, focusing on utilizing linear equations, we use extracted quadratic equations to find linear approximations with high probability. In [1, 5], to approximate nonlinear equations with linear ones, nonlinear terms are eliminated, which causes low probability most often. To tackle this problem, we utilize the linear combinations of quadratic equations to approximate nonlinear equations with higher probability as presented in Algorithm 1.

As presented in Algorithm 1, at line 2 to 4, to have enough number of quadratic equations, some new quadratic equations are generated by multiplying the linear equations with x_i, where x_i stands for the corresponding variable of the key bit i. Then the coefficient matrix of all quadratic equations in $SetQ$ is calculated in matrix A.

At the next step, linear combinations of these quadratic equations are calculated by performing some row operations. At the step i of line 8, for quadratic term i, we try to find a row which it has 1 at entry i,i. This row is swapped with row i and XORed to all rows which have 1 at this column.

Repeating this step for all quadratic terms, matrix A is converted to an upper triangular matrix which each quadratic term is corresponds with only one row with entry 1 on the diagonal. In other words:

$$\forall i \mid 1 \leq i \leq \binom{n_I}{2}, a_{i,i}=1$$

And $\forall j < i$, we have $a_{j,i}=0$.

In continue, at line 14, every nonlinear equation is approximated to a linear one. To do this, at step i of line 16, if nonlinear equation has quadratic term i, the equation is XORed with corresponding row of matrix A. It means that the corresponding quadratic term is replaced with an equation instead of eliminating it.

After determining the linear approximation of a nonlinear superpoly, the probability of equation correctness is calculated. Assuming that the output of the real superpoly is o and output of linear approximated part is $o^{'}$, the probability of approximation can be estimated under N random keys as follows:

$$p = pr(o \bigoplus o^{'}=0) = \frac{\sum_{i=1}^{N}\left(o=o^{'}\right)}{N} \quad (3)$$

For the probability of being reliable, a sufficient number of random keys is required. This number can be calculated based on the theorem presented at [5]. For further details, see Appendix.

At the end of this phase, there will be m_d deterministic and m_p probabilistic independent linear equations, $\{f_1,\ldots,f_{m_d},f_{m_d+1},\ldots,f_{m_d+m_p}\}$, with n variables. Thus, the error rate, the ratio of probabilistic equations to all equations, is calculated as $\frac{m_p}{m_d+m_p}$. Since secret key bits are confused in the encryption, most of the extracted superpolies are nonlinear. Hence, given the scarcity of linear ones, the probability of probabilistic equations occurrence as the linear approximations of nonlinear superpolies is high. As a result, this type of equation constitutes a large share of the equation system, which consequently raises the error rate of the equation system. It should be noted that due to the hardness of equation extraction, the extracted equation system is mostly determined.

3.2 Key Recovery Phase

Given the high error rate of the equation system extracted in the previous phase, the existing approaches that underline the finding of a solution with minimum noises were not successful in key recovery. Thus, in most scenarios, they failed to find the right key [9]. To tackle this problem, we propose an approach to solving a linear equation system with a high error rate, and it was utilized to solve the extracted equation system in the previous phase and recover the key deterministically.

To solve such an equation system and find the exact solution, we should assign proper val-

Algorithm 1 Linear Approximation of Nonlinear Equations

```
Input L=set of all linear equations
      Q=set of all quadratic equations
      NL=set of all nonlinear equations with degree more than 2
Output App_L:set of approximated linear equations
1: $SetQ = \{Q\}$
2: for $\forall i \in I$ do          ▷ generate sparse and Overdefined Quadratic equation system
3:     for $\forall L_j \in L$ do
4:         $SetQ$.append $(x_i.L_j)$
5:     end for
6: end for
7: $A = Coefficient\_matrix(SetQ)$
8: for $i = 1\, to \begin{pmatrix} n_I \\ 2 \end{pmatrix}$ do          ▷ linear combination of nonlinear equations
9:     $Find\_and\_Swap(A, i)$
10:    for $j = i + 1\, to A.rows$ do
11:        $A[j] = A[j] \bigoplus A[i]$          ▷ row operations
12:    end for
13: end for
14: for $\forall L_k \in NL$ do          ▷ replace quadratic terms
15:    $entry = Q\_and\_L\_Coeff(L_k)$          ▷ the coefficient of linear and quadratic terms are moved to entry
16:    for $i = 1\, to \begin{pmatrix} n_I \\ 2 \end{pmatrix}$ do
17:        if $entry[i] == 1$ then
18:            $entry[i] = A[i] \bigoplus entry[i]$          ▷ row operations
19:        end if
20:    end for
21: end for
22: $app\_equation = linear\_equation(entry)$          ▷ linear approximation
23: $App\_L.append(app\_equation)$
```

ues to the noise of the probabilistic equations. The most obvious way is to exhaustively search all such noise vector in the order of increasing Hamming weight where $(e_1, e_2, \ldots, e_m)$ is equal $(0, 0, ..., 0), (1, 0, ..., 0), ..., (1, 1, ..., 1)$ and to solve the equation system for each noise vector. If there is a solution for the corresponding system, this solution is checked as a candidate for the right key. Otherwise, the process is continued with the next value for the noise vector until the right key is found. The main drawback of this method is redundant computations due to contradictions in the equation system leading to high computational complexity.

By considering the way that ISBS detects contradictions and eliminates redundant computations, we create the search tree as ISBS method to find the proper noise values in lower complexity; however, as mentioned earlier, ISBS is not efficient for the equation systems with a high error rate. Thus, some improvements are required to enhance the efficiency and practicality of solving approaches for such systems:

- The search tree is created by considering the probability of equations. Indeed, the probabilistic equations are appeared in the search tree based on their probability values in descending order. Hence, the branches are visited in the order of their correctness probability, which leads to finding the exact solution with less backtracking.
- To decrease the complexity, consistency checking is used instead of partial solving in the inner nodes of the tree. As the equations are linear, the Rouché−Capelli theorem can be used to check the consistency of the equation system.

Theorem 1. ***Rouché−Capelli Theorem [24]:*** *A system of linear equations with n variables has a solution if and only if the rank of its coefficient matrix A is equal to the rank of its augmented matrix $[A \mid b]$. In particular:*

- *If $n = rank\,(A)$, the solution is unique.*
- *If $rank\,(A) = rank\,([A \mid b])$, there will be infinite solutions.*
- *Otherwise, the system is inconsistent and there*

will be no solutions

By the rank of a matrix, we mean the maximum number of linearly independent vectors in a matrix which here it shows the maximum number of independent equations.

Based on the above descriptions, the steps of our approach to solving the probabilistic equation system are as follow:

1) All deterministic equations are included in the set S.

2) The probabilistic equations are incrementally added to S, until all probabilistic equations can be included. At level i, $1 \leq i \leq m_p$, equation $f_{m_d+i}+e_i = 0$ is added to set S with $e_i = 0$, and the consistency of equation set S is checked.

2-1) A consistent set of equations suggests that there are no contradiction in the equation set and therefore we can continue with the next probabilistic equation at the next level.

2-2) However, in the case of inconsistency, we prune the branch, change the value of e_i to 1 and then continue with the next equation.

3) At leaf nodes of the search tree, which are located at the height of m_p, there are all probabilistic equations with proper noises which constitute a full rank matrix. They can be solved by Gaussian elimination. Then the solution is checked as a candidate for the right key:

3-1) If it offers the right key, the process is ended.

3-2) Otherwise, backtracking is needed to reach an inner node where e_i can be flipped to 1. Step (2) is performed to search for new noise value in this new branch. As equations are added to the tree based on their probability values, we ensure that the solutions are visited based on the probability of occurrence. As such, the next branch will have the highest probability.

Remarks:

In comparison with the original ISBS:

- ISBS continues visiting leaf nodes of the search tree and backtracking to find lower noise value until the minimum value in the tree is reached. In our work, all consistent branches are searched based on probability values in a descending order to find the right key. However, the number of branches visited may be higher than that of ISBS, though it is capable of finding the right key deterministically under all the scenarios.
- Due to incremental solving at inner nodes and saving the partial solutions in ISBS, at high error rates, the tree height grows and the efficiency of solving such equation systems impairs. To overcome this drawback, we only check the consistency of equations at inner nodes with complexity $O\left(n^3\right)$ instead of mentioned operations as ISBS.

Compared to the cube attack, by extending the equation system to more equations, we can determine more key bits by solving these equations. Consequently, since the number of remained key bits which should be determined via the exhaustive search is decreased, this attack scenario is not as complex as the original cube attack.

4 Results

To prove the efficiency of the proposed attack scenario, in this section, we apply the attack scenario to the lightweight block cipher KATAN and present achieved results. The results showed that although we cloud not reach the highest number of rounds attacked by non-algebraic techniques [25], considering that our emphasis in this paper is cube equations extraction and solving, the proposed approach outperformed the best existing results of this type of attack on KATAN32. It is notable that the cube attack explored in [26] has a different approach to break the cipher. Indeed in this work, a cube tester distinguisher was extracted and extended over upper and lower rounds, given that some key bits were guessed. As shown in Table 2, the complexity of the proposed attack scenario is considerably less than the complexity of others. It should be mentioned that in algebraic view, as illustrated in [18], higher round attacks with fewer equation numbers could be targeted but they have high complexity.

In [27], the algebraic and cube attacks were applied to the KATAN family. More specifically, the cube attack broke 60-round KATAN32 by extracting 41 equations. In addition, they could break 79 rounds of KATAN32 with 45 key guesses using SAT solvers to solve the quadratic equations. In [18] cube attack using division property was applied to 72-round KATAN32, and 44 linear equations were extracted. Here we applied the proposed attack scenario to 80-round KATAN32 and overtook existing algebraic attacks. It is noteworthy that in comparison to [18], although we use the same method to extract equations, we could improve the results of [18] to higher round and more equations. The results are summarized in Table 2.

Based on the procedure described to extract equations, we could extract 58 deterministic and 19 probabilistic independent linear equations. The equations were presented in the Appendix. To prove the effi-

Table 2. Overview of the attacks on KATAN32.

Attack type	Round	Time Comp.	reference
Cube	60	2^{39}	[27]
Cube	72	2^{36}	[18]
Cond. Diff.	78	2^{22}	[28]
Algebraic	79	$2^{54.7}$	[27]
Pro. Cube	80	$2^{35.77}$	Here
Cube	90	2^{77}	[18]
Diff.	114	2^{77}	[29]
ASR MITM	119	$2^{79.1}$	[30]
Matchbox MITM	153	2^{78}	[31]
Dynamic Cube	155	$2^{78.3}$	[26]
MD MITM	206	2^{79}	[25]

ciency of the proposed solving approach in high error rates, we experiment different scenarios for a different number of probabilistic equations in range 4 to 19, which correspond with error rates in range 0.05 to 0.24, respectively. Indeed, by efficiency, we mean that the problem can be solved in a reasonable time. Achieved results were presented in Table 3. In all scenarios, we have a full rank coefficient matrix of linear equations and solve it using Gaussian elimination. Since the number of independent equations, including deterministic and probabilistic ones, should be equal to the number of variables, we fix some key bits. It should be mentioned that the efficiency of the proposed attack is independent of these key bits, so they were chosen randomly. The experiments were repeated for 100 random key scenarios on a PC with 3.40 GHz 8 core CPU, and 31.4 GB Memory.

The columns min.time, max.time and avg.time give the minimum, maximum, and average running time to find the right key in all random scenarios, respectively. Consumed time depended on the height of the search tree and the number of checked branches to find consistent noises and solve the achieved equation system to determine the right solution. Because of the mentioned improvements on the structure of the search tree in comparison with [9], the time complexity of constructing and backtracking search tree to find the solution is decreased drastically. Considering the fact that the number of probabilistic equations corresponds with the height of the search tree, in 12 probabilistic equations corresponding with error rate 0.05 in [9], ISBS solved the problem in 382 seconds, where we could achieve the solution in much lower time near 23 seconds for 13 probabilistic equations. In addition, for near 20 probabilistic equations at error rate 0.08 in [9], it was claimed that near 6 hours was needed to solve such equation system, whereas we spend near half hour for 19 probabilistic equations which is much lower than ISBS.

In the original cube attack, by using 58 deterministic linear equations,, the equation system is solved, and values of 58 key bits are determined. The remained key bits are determined via exhaustive search. Considering the time spend to check a key as unit time, the computational complexity of determining these key bits is 2^{22}. Hereby extending the equation system using 19 probabilistic equations, we could determine the remained key bits with complexity $2^{GK}.(1973s) \approx 2^3.2^{10.95} \approx 2^{14}$, which is lower than the complexity of the exhaustive search. As described earlier, some randomly chosen key bits should be fixed to have a full rank matrix. Here GK parameter shows the number of these guessed key bits.

As shown in the Appendix, we extract 58 equations of cube size 29, 30, and 31. To calculate the complexity of the attack, it is notable that as some of the ciphertext bits revealed more information about key bits, we could extract different equations for various ciphertext bits using the same maxterm, which reduces the complexity of the attack. Thus, the time complexity of online phase is $13.2^{29}+12.2^{30}+18.2^{31}+2^{14} \approx 2^{35.77}$ to determine secret key bits. It should be mentioned that although because of calculating the right hand side of cube equations, the complexity of deterministic and probabilistic attack scenarios are almost similar, but we cannot ignore the difference between the complexity of determining key bits in probabilistic scenario, 2^{14}, and deterministic scenario, 2^{22}, which represents the improvement of attack complexity in probabilistic cube attack. Indeed, utilizing probabilistic equations and finding proper noise values based on the proposed approach lead to decreased attack complexity.

5 Conclusion and Future Works

In this paper, probabilistic equation extraction and solving a high error rate equation system were studied. In the equation extraction phase, to approximate nonlinear equations, linear combinations of quadratic equations were utilized, which consequently could increase the probability of linear approximations. Since existing approaches failed to find an exact solution of the equation system with a high error rate, we improved the best current approach, ISBS, to solve the high error rate linear equation system efficiently. Utilizing the improved approach, equation system was extended to probabilistic ones, and the complexity of the cube attack was decreased. As described earlier, instead of incremental solving and saving the partial solution set in ISBS, we check the consistency of the equation system to improve the complexity of the solving procedure. As future work, it will be interesting if we can extend the consistency checking to the quadratic equation system. Since all nonlinear polynomials can be converted to quadratic ones, we can generalize this improvement for nonlinear equation systems, too.

Table 3. Obtained results for different error rates

Error rate	Num. Of Prob. Equ.	Num. Of Indep. Equ.	Num. Of GK	Min. time in Sec.	Max. time in Sec.	Avg. time in Sec.
0.05	4	62	18	0.06	0.27	0.16
0.08	7	65	15	0.12	2.65	0.66
0.16	13	71	9	0.20	143.51	22.95
0.19	15	73	7	0.24	648.47	103.08
0.21	17	75	5	0.29	2715.97	432.55
0.24	19	77	3	0.34	11952.54	1973.22

References

[1] T. Nicolas Courtois. Higher order correlation attacks, xl algorithm and cryptanalysis of toyocrypt. In *Information Security and Cryptology — ICISC 2002*, pages 182–199. Springer Berlin Heidelberg, 2003.

[2] Ilya Mironov and LintaoZhang. Applications of sat solvers to cryptanalysis of hash functions. In *Theory and Applications of Satisfiability Testing - SAT 2006*, pages 102–115. Springer Berlin Heidelberg, 2006.

[3] Jean-Charles Faugere and Antoine Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using grobner bases. In *Advances in Cryptology - CRYPTO 2003*, pages 44–60. Springer Berlin Heidelberg, 2003.

[4] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology - EUROCRYPT 2009*, pages 278–299. Springer Berlin Heidelberg, 2009.

[5] Yuan Yao, Bin Zhang, and Wen-Ling Wu. Utilizing probabilistic linear equations in cube attacks. *Journal of Computer Science and Technology*, 31(2):317–325, 2016.

[6] Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of boolean functions. In *Advances in Cryptology - EUROCRYPT 2004*, pages 474–491. Springer Berlin Heidelberg, 2004.

[7] Pratish Datta, Dibyendu Roy, and Sourav Mukhopadhyay. A probabilistic algebraic attack on the grain family of stream ciphers. In *Network and System Security*, pages 558–565. Springer International Publishing, 2014.

[8] Martin Albrecht and Carlos Cid. Cold boot key recovery by solving polynomial systems with noise. In *Applied Cryptography and Network Security*, pages 57–72. Springer Berlin Heidelberg, 2011.

[9] Zhenyu Huang and Dongdai Lin. Solving polynomial systems with noise over f2: Revisited. *Theoretical Computer Science*, 676:52 – 68, 2017.

[10] Julia Borghoff, Lars R. Knudsen, and Mathias Stolpe. Bivium as a mixed-integer linear programming problem. In *IMA Int. Conf.*, 2009.

[11] Mohamed Ahmed Abdelraheem, Julia Borghoff, Erik Zenner, and Mathieu David. Cryptanalysis of the light-weight cipher a2u2. In *Cryptography and Coding*, pages 375–390. Springer Berlin Heidelberg, 2011.

[12] Xiao shan Gao and Zhenyu Huang. Efficient characteristic set algorithms for equation solving in finite fields and application in analysis of stream ciphers. Cryptology ePrint Archive, Report 2009/637, 2009. `https://eprint.iacr.org/2009/637`.

[13] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. 47(3):549–595, 1993.

[14] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round md6 and trivium. In *Fast Software Encryption*, pages 1–22, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[15] Itai Dinur and Adi Shamir. Breaking grain-128 with dynamic cube attacks. In *Fast Software Encryption*, pages 167–187, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. Katan and ktantan — a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 272–288, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[17] 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*, pages 250–279. Springer International Publishing, 2017.

[18] Z. Eskandari and A. Ghaemi Bafghi. Cube distinguisher extraction using division property in block ciphers. *IET Information Security*, 14(1):72–80, 2020.

[19] Yosuke Todo. Structural evaluation by generalized integral property. In *Advances in Cryptology – EUROCRYPT 2015*, pages 287–314. Springer Berlin Heidelberg, 2015.

[20] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In *Fast Software Encryption*, pages 357–377. Springer Berlin Heidelberg, 2016.

[21] Ling Sun, Wei Wang, and Meiqin Wang. Milp-aided bit-based division property for primitives

with non-bit-permutation linear layers. Cryptology ePrint Archive, Report 2016/811, 2016. https://eprint.iacr.org/2016/811.

[22] Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for arx ciphers and word-based division property, 2017.

[23] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology — ASIACRYPT 2002*, pages 267–287, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[24] Igor Shafarevich and Alexey R. Remizov. *linear algebra and geometry.* Springer Science & Business Media, 2012.

[25] Shahram Rasoolzadeh and Håvard Raddum. Improved multi-dimensional meet-in-the-middle cryptanalysis of katan. 2016. https://eprint.iacr.org/2016/077.

[26] Zahra Ahmadian, Shahram Rasoolzadeh, Mahmoud Salmasizadeh, and Mohammad Reza Aref. Automated dynamic cube attack on block ciphers: Cryptanalysis of simon and katan. Cryptology ePrint Archive, Report 2015/040, 2015. https://eprint.iacr.org/2015/040.

[27] Gregory V. Bard, Nicolas T. Courtois, Jorge Nakahara, Pouyan Sepehrdad, and Bingsheng Zhang. Algebraic, aida/cube and side channel analysis of katan family of block ciphers. In *Progress in Cryptology - INDOCRYPT 2010*, pages 176–196. Springer Berlin Heidelberg, 2010.

[28] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of nlfsr-based cryptosystems. In *Advances in Cryptology - ASIACRYPT 2010*, pages 130–145, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[29] Martin R. Albrecht and Gregor Leander. An all-in-one approach to differential cryptanalysis for small block ciphers. In *Selected Areas in Cryptography*, pages 1–15, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[30] Takanori Isobe and Kyoji Shibutani. Improved all-subkeys recovery attacks on fox, katan and shacal-2 block ciphers. In *Fast Software Encryption*, pages 104–126, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[31] Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against katan. In *Fast Software Encryption*, pages 61–81, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

Zahra Eskandari is a Ph.D. candidate in computer engineering at Ferdowsi University of Mashhad (FUM). She received the B.S. degree in Computer Engineering from Kharazmi University, Tehran, Iran, in 2006 and the M.S. degree in Computer Engineering at FUM, Iran, in 2008. She was with the cybersecurity section at DTU compute, Denmark as a visiting researcher from July 2016 to March 2017. Her research interests include algebraic cryptanalysis, cube, and integral attacks. She is particularly interested in solving probabilistic/noisy equation system, linear and nonlinear ones.

Abbas Ghaemi Bafghi received his B.S. degree in Applied Mathematics in Computer from Ferdowsi University of Mashhad, Iran, in 1995. He received his M.S. and Ph.D. degrees in Computer engineering from Amirkabir (Tehran Polytechnique) University of Technology, Iran, in 1997 and 2004 respectively. He is a member of the Computer Society of Iran (CSI) and Iranian Society of Cryptology (ISC). He is an associate professor in the Department of Computer Engineering, Ferdowsi University of Mashhad, Iran. His research interests are in cryptology and security.

6 Appendix: Detail of the Extracted Equations

The detail of the extracted equations, 58 deterministic and 19 Probabilistic linear equations for 80-round KATAN32 were presented in Table 4 and Table 5, respectively. In all deterministic equations, we ran 500 linearity tests, and then the equations were checked for more than 2^{10} distinct random keys to ensure their correctness. About the probabilistic equations, based on the [12], to have a reliably estimated probability with a confidence interval of length 0.05 and confidence level 0.999, the probability of each equation, was estimated under 1128 pairs of (plaintext, ciphertext). To do these experiments, the cube attack was simulated in the CUDA framework on Nvidia GPUs (GeForce GTX 1080) with 8 GB RAM memory and 2560 CUDA cores. Some of the extracted quadratic equations were presented in Table 6. It is notable that we could extract more equations, but they have a high number of quadratic terms between 22 and 203.

Table 4: Extracted Deterministic Linear Equations for 80-round KATAN32.

Maxterm	Cube Size	Equation	Out. Bit
FFFFBFFA	29	k0+k19+k30+k61+k67+k76	16
FFFFEFF5	29	k47+k68+k69+k72+k77+1	17
FFFFEFFA	29	k1+k20+k31+k68+k72	15
FFFFFEFA	29	k1+k6+k10+k19+k20+k25+k29+k31+k36+k38+k40+k49+k63+k68+k69+k74+k75+k77+k78	11
FFFFFF75	29	k1+k3+k16+k20+k22+k31+k33+k35+k46+k68+k70	13
FFFFFFBA	29	k6+k25+k36+k73+1	31
FFFFFFD5	29	k1+k3+k9+k20+k22+k28+k31+k33+k39+k68+k73+k76+1	11
FFFFFFE5	29	k0+k1+k6+k19+k20+k25+k30+k31+k36+k57+k63+k64+k67+k68+k72+k73+k76+k78+k79+1	13
FFFFFFEA	29	k0+k3+k11+k19+k22+k33+k41+k67+k70+k72+k75+k78+1	10
FFFFFFEC	29	k1+k4+k7+k8+k17+k20+k23+k26+k27+k31+k34+k36+k37+k38+k47+k51+k57+k67+k68+k71+k73 +k74+k75+k76+k77+1	12
FFFFFFF1	29	k2+k21+k32+k59+k67+k69+k70+k71+k72+k75+1	12
FFFFFFF2	29	k1+k2+k20+k21+k31+k32+k66+k68+k69+k71+k74+k75+k77+1	12
FFFFFFF4	29	k2+k21+k32+k69+k70+k72+k75+k79	12
		k2+k21+k32+k59+k69+k70+k71+k72+k74+k76+k79	31
FFFFEFFE	30	k0+k4+k10+k19+k23+k29+k30+k34+k40+k57+k61+k64+k67+k68+k69+k70+k72+k74+k77+k78+k79	11
FFFFFEFE	30	k2+k15+k21+k32+k34+k45+k69+k75+1	8
FFFFFF6F	30	k0+k5+k13+k19+k24+k30+k32+k35+k43+k53+k57+k63+k67+k72+k73+k74+k79+1	11
		k2+k21+k32+k57+k69+k75+k78+k79	12
FFFFFFCF	30	k2+k3+k4+k11+k21+k22+k23+k30+k32+k33+k34+k41+k69+k70+k71+k78+1	30
FFFFFFDE	30	k3+k5+k22+k24+k33+k35+k70+k72+k73+1	29
		k3+k7+k13+k22+k26+k32+k33+k37+k43+k70+k72+k74+k77	9
FFFFFFEE	30	k0+k1+k3+k6+k10+k11+k14+k20+k22+k25+k29+k31+k36+k38+k40+k41+k44+k49+k67+k68+k70 +k73+k77+k78+1	6
		k8+k27+k38+k75+k77	7
FFFFFFF3	30	k2+k6+k7+k8+k15+k21+k25+k26+k27+k32+k34+k36+k37+k38+k45+k69+k73+k74+k75+1	28
FFFFFFF5	30	k3+k16+k22+k33+k35+k46+k70+1	26
FFFFFFF6	30	k6+k7+k25+k26+k36+k37+k65+k73+k74+1	8
		k0+k1+k4+k8+k11+k13+k17+k19+k20+k21+k23+k27+k31+k32+k34+k36+k38+k40+k41+k43+k47 +k51+k61+k65+k67+k68+k69+k75+k76+k78+1	7
FFFFFFF9	30	k4+k8+k9+k17+k23+k27+k28+k34+k36+k38+k39+k47+k71+k75+k76	27
		k1+k3+k10+k20+k22+k29+k31+k33+k40+k68+k70+k77	8
FFFFFFFA	30	k5+k18+k24+k35+k37+k48+k72+1	25
FFFFFFFC	30	k6+k10+k11+k12+k19+k25+k29+k30+k31+k36+k38+k40+k41+k42+k49+k73+k77+k78+k79	26
FBFFFFFF	31	k0+k2+k6+k19+k21+k25+k30+k32+k36+k60+k67+k69+k73+k74	29
FDFFFFFF	31	k2+k4+k6+k8+k21+k23+k25+k27+k32+k34+k36+k38+k62+k69+k71+k73+k75+k76+1	28
FEFFFFFF	31	k4+k6+k10+k23+k25+k29+k34+k36+k40+k64+k71+k73+k77+k78	27
FF7FFFFF	31	k0+k6+k8+k12+k19+k25+k27+k30+k31+k36+k38+k42+k66+k67+k73+k75+k79+1	26
FFBFFFFF	31	k1+k2+k8+k10+k14+k20+k21+k27+k29+k31+k32+k33+k38+k40+k44+k69+k75+k77	25
FFDFFFFF	31	k3+k4+k10+k12+k16+k22+k23+k29+k31+k33+k34+k35+k40+k42+k46+k71+k77+k79+1	24
FFF7FFFF	31	k1+k3+k7+k8+k11+k14+k16+k22+k24+k26+k27+k30+k31+k35+k37+k38+k39+k41+k43+k44+k46 +k50+k54+k68+k70+k75+k78	22
		k10+k29+k40+k77+1	23
FFFBFFFF	31	k2+k21+k32+k37+k41+k49+k53+k56+k57+k58+k61+k62+k63+k65+k67+k68+k69+k78	15
		k3+k10+k22+k29+k33+k40+k70+k75+k77	16
FFFDFFFF	31	k4+k23+k34+k71+k74	14
		k5+k12+k24+k31+k35+k42+k72+k77+k79+1	15
FFFEFFFF	31	k1+k14+k20+k31+k33+k44+k68+k79	14
		k1+k6+k20+k25+k31+k36+k68+k73	13
FFFFDFFF	31	k8+k27+k38+k75+1	30
FFFFF7FF	31	k1+k2+k3+k6+k11+k12+k16+k20+k21+k22+k25+k30+k32+k33+k35+k36+k41+k42+k46+k51+k55 +k63+k67+k68+k69+k71+k72+k73+k76+k77+k78+1	8
		k12+k31+k42+k79	28
FFFFFBFF	31	k1+k14+k20+k31+k33+k44+k68+1	27
		k0+k2+k6+k11+k13+k15+k21+k25+k26+k34+k36+k38+k41+k43+k49+k56+k67+k69+k73+k78+1	8
FFFFFFBF	31	k9+k22+k28+k39+k41+k52+k76+1	23
FFFFFFDF	31	k3+k8+k10+k16+k21+k22+k23+k27+k33+k35+k36+k38+k42+k46+k48+k51+k53+k55+k59+k66 +k70+k75+k77	3
		k11+k24+k30+k41+k43+k54+k78+1	22
FFFFFFEF	31	k0+k13+k19+k26+k30+k32+k43+k45+k56+k67+1	21
		k4+k7+k17+k20+k23+k26+k30+k34+k36+k37+k39+k47+k49+k50+k60+k71+k74	1
FFFFFFF7	31	k2+k15+k21+k28+k32+k34+k45+k47+k58+k69	20
		k1+k12+k14+k20+k25+k27+k33+k40+k42+k46+k55+k57+k59+k68+k70+k79+1	1

FFFFFFFB	31	k4+k17+k23+k30+k34+k36+k47+k49+k60+k71+1	19

Table 5. Extracted Probabilistic Linear Equations for 80-round KATAN32.

Maxterm	Cube Size	Equation	Out. Bit	Prob.
FFFFFFE6	29	k5+k24+k35+k53+k59+k61+k65+k72+k75+k76	13	0.758
		k4+k5+k9+k11+k18+k23+k24+k28+k30+k34+k35+k37+k39+k41+k43+k48+k49+k53+k57+k61+k63+k65 +k67+k69+k70+k71+k72+k73+k77+1	12	0.771
FFFFFFF4	29	k3+k5+k9+k22+k24+k28+k33+k35+k39+k53+k65+k68+k69+k70+k71+k72+k73+k74+k79+1	11	0.755
FFFEFFFE	30	k12+k31+k42+k53+k63+k70+k71+k76+k79	15	0.737
FFFF7FFE	30	k0+k19+k30+k47+k53+k61+k62+k63+k64+k67+k72+k73+k74+k75+k76	16	0.767
FFFFEFFE	30	k0+k5+k10+k15+k16+k19+k24+k29+k30+k34+k40+k45+k46+k49+k61+k67+k68+k70+k71+k72+k74 +k75+k76+k77+k79	31	0.748
FFFFF7FE	30	k0+k1+k3+k5+k6+k16+k19+k20+k22+k24+k25+k30+k31+k33+k36+k46+k53+k63+k67+k68+k70+k72 +k73+k74+k75+k76+1	13	0.758
FFFFFBFE	30	k0+k3+k4+k5+k14+k18+k19+k21+k22+k23+k24+k30+k34+k35+k37+k40+k44+k48+k51+k67+k69+k70 +k71+k72+k73+k74+k77+1	30	0.745
FFFFFEFE	30	k0+k3+k7+k8+k19+k22+k26+k27+k30+k33+k37+k38+k70+k73+k74+k75+k77	27	0.758
FFFFFFE9	29	k0+k1+k3+k4+k6+k7+k11+k13+k22+k23+k25+k26+k31+k32+k33+k34+k36+k37+k38+k39+k41+k43+k45 +k50+k63+k67+k68+k69+k70+k73+k74+k75+k78	11	0.747
FFFFFF6F	30	k1+k2+k5+k6+k16+k19+k20+k21+k24+k25+k31+k32+k36+k38+k45+k46+k49+k57+k61+k63+k65+k67 +k68+k69+k70+k73+k74+k75	31	0.756
FFFFFFBE	30	k1+k2+k3+k7+k12+k21+k22+k26+k32+k33+k37+k39+k42+k50+k65+k67+k68+k69+k70+k73+k74+k79+1	26	0.742
FFFFFFF6	30	k1+k9+k10+k13+k20+k24+k27+k28+k29+k31+k32+k39+k40+k46+k53+k54+k57+k65+k68+k69+k73 +k75+k76+k77+k78+k79+1	27	0.764
7FFFFFFF	31	k56+k66+k70+k74	14	0.741
BFFFFFFF	31	k3+k16+k22+k33+k35+k46+k48+k66+k70+1	13	0.751
DFFFFFFF	31	k7+k26+k37+k42+k46+k48+k50+k58+k60+k64+k68+k78	12	0.87
FEFFFFFF	31	k6+k8+k10+k13+k25+k27+k29+k32+k33+k34+k36+k38+k40+k43+k62+k64+k66+k70+k73+k74+k75+k77	6	0.66
		k0+k19+k30+k67+k70+k72+k74+k76+1	7	0.877
FFFFBFFF	31	k5+k7+k8+k9+k24+k26+k27+k28+k31+k35+k37+k38+k39+k40+k57+k71+k72+k76+k77+k79	10	0.747

Table 6. Extracted Quadratic Equations for 80-round KATAN32.

Maxterm	Equation	Out. Bit
F7FFFFFF	k0+k2+k4+k19+k21+k23+k30+k32+k34+k54+k58+k62+k67+k69+k71+k76+k54.k76+k58.k70+k58.k76 +k64.k76	30
FFFFFF5F	k0+k1+k4+k5+k6+k7+k9+k10+k13+k19+k23+k24+k25+k26+k28+k29+k30+k31+k32+k34+k35+k36+k37 +k40+k43+k50+k59+k67+k68+k71+k72+k4.k79+k10.k74+k23.k79+k29.k74+k34.k79+k40.k74+k51.k79 +k71.k79+k74.k77+k76.k79+1	29
BFFFFFFF	k3+k16+k22+k33+k35+k46+k48+k66+k70+k2.k54+k2.k58+k3.k54+k3.k58+k16.k54+k16.k58+k21.k54 +k21.k58+k22.k54+k22.k58+k32.k54+k32.k58+k33.k54+k33.k58+k35.k54+k35.k58+k46.k54+k46.k58+k54.k69 +k54.k70+k58.k69+k58.k70+1	13
FFFFFFE6	k5+k24+k35+k53+k59+k61+k65+k72+k75+k76+k0.k65+k19.k65+k30.k65+k65.k67	13
FFFFFFE6	k4+k5+k9+k11+k18+k23+k24+k28+k30+k34+k35+k37+k39+k41+k43+k48+k49+k53+k57+k61+k63+k65 +k67+k69+k70+k71+k72+k73+k77+k2.k74+k7.k74+k21.k74+k26.k74+k32.k74+k37.k74+k57.k74+k69.k74 +k74.k75+k74.k76+1	12
FFFFFF6F	k1+k2+k5+k6+k16+k19+k20+k21+k24+k25+k31+k32+k36+k38+k45+k46+k49+k57+k61+k63+k65+k67 +k68+k69+k70+k73+k74+k75+k68.k75+k75.k76	31
FFFFFFF4	k3+k5+k9+k22+k24+k28+k33+k35+k39+k53+k65+k68+k69+k70+k71+k72+k73+k74+k79+k4.k9+k4.k28 +k4.k39+k4.k76+k9.k23+k9.k34+k9.k71+k9.k74+k23.k28+k23.k39+k23.k76+k28.k34+k28.k71+k28.k74 +k34.k39+k34.k76+k39.k71+k39.k74+k71.k76+k74.k76+1	11
EFFFFFFF	k0+k2+k6+k19+k21+k25+k30+k32+k36+k60+k67+k69+k73+k78+k52.k74+k56.k68+k56.k74+k62.k74	31
FFEFFFFF	k1+k3+k4+k5+k12+k14+k16+k18+k20+k22+k23+k24+k34+k37+k42+k44+k46+k48+k68+k70+k71+k79 +k4.k72+k10.k68+k10.k78+k23.k72+k29.k68+k29.k78+k34.k72+k40.k68+k40.k78+k68.k77+k71.k72+k77.k78	23
7FFFFFFF	k56+k66+k70+k74+k0.k52+k0.k56+k1.k52+k1.k56+k14.k52+k14.k56+k19.k52+k19.k56+k20.k52+k20.k56 +k30.k52+k30.k56+k31.k52+k31.k56+k33.k52+k33.k56+k44.k52+k44.k56+k52.k67+k52.k68+k56.k67+k56.k68	14
FFFFFFE9	k2+k11+k21+k30+k32+k41+k63+k66+k69+k72+k75+k78+k79+k7.k72+k7.k74+k26.k72+k26.k74+k37.k72 +k37.k74+k72.k74+1	12
FFFFFFF6	k1+k9+k10+k13+k20+k24+k27+k28+k29+k31+k32+k39+k40+k46+k53+k54+k57+k65+k68+k69+k73+k75 +k76+k77+k78+k79+k3.k4+k3.k23+k3.k34+k3.k71+k3.k76+k4.k22+k4.k33+k4.k70+k22.k23+k22.k34 +k22.k71+k22.k76+k23.k33+k23.k70+k33.k34+k33.k71+k33.k76+k34.k70+k70.k71+k70.k76+1	27