Fault Analysis of the KATAN Family of Block Ciphers

Shekh Faisal Abdul-Latip[1,2], Mohammad Reza Reyhanitabar[1], Willy Susilo[1], and Jennifer Seberry[1]

[1] Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering,
University of Wollongong, Australia
{sfal620, rezar, wsusilo, jennie}@uow.edu.au

[2] Information Security and Digital Forensics Lab (INSFORLAB),
Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka, Malaysia
shekhfaisal@utem.edu.my

Abstract. In this paper, we investigate the security of the KATAN family of block ciphers against differential fault attacks. KATAN consists of three variants with 32, 48 and 64-bit block sizes, called KATAN32, KATAN48 and KATAN64, respectively. All three variants have the same key length of 80 bits. We assume a single-bit fault injection model where the adversary is supposed to be able to corrupt a single random bit of the internal state of the cipher and this fault injection process can be repeated (by resetting the cipher); i.e., the faults are transient rather than permanent. First, we determine suitable rounds for effective fault injections by analyzing distributions of low-degree (mainly, linear and quadratic) polynomial equations obtainable using the cube and extended cube attack techniques. Then, we show how to identify the exact position of faulty bits within the internal state by precomputing difference characteristics for each bit position at a given round and comparing these characteristics with ciphertext differences (XOR of faulty and non-faulty ciphertexts) during the online phase of the attack. The complexity of our attack on KATAN32 is 2^{59} computations and about 115 fault injections. For KATAN48 and KATAN64, the attack requires 2^{55} computations (for both variants), while the required number of fault injections is 211 and 278, respectively.

Keyword: Block ciphers, cube attack, differential fault analysis, KATAN.

1 Introduction

Fault analysis as a type of side channel attack (or implementation attack) was originally introduced by Boneh et al. [6] by an attack against implementations of public key algorithms. The method was then adapted and extended by Biham and Shamir [5] to differential fault analysis, making it applicable to implementations of symmetric key algorithms as well [9, 10]. Several models for fault attacks have been introduced in the literature, among which we adopt a popular model, called transient single-bit fault model, as used for example in [10, 9]. In this model it is assumed that the adversary can inject one bit of error into the internal state of a cipher during its execution (e.g. using a laser beam) without damaging the bit position permanently; that is, the cipher can be reset to resume its normal (unfaulty) operation and this fault injection can be repeated as many times as required. For some interesting practical settings for carrying out these attacks we refer to [15].

In this paper we present fault attacks on the KATAN family of block ciphers [7]. KATAN consists of three variants with 32, 48 and 64-bit block sizes, named KATAN32, KATAN48 and KATAN64, respectively. All three variants have the same key length of 80 bits. KATAN aims at meeting the needs of an extremely resource-limited environment such as RFID tags. Assuming the transient single-bit fault attack model as used for example in [10, 9], we present a differential fault attack empowered by the algebraic techniques of the cube attack [8] and its extended variants [1].

The cube attack, put forth by Dinur and Shamir at EUROCRYPT 2009 [8], is a generic type of algebraic attack that may be applied against any cryptosystem, provided that the attacker has access to a bit of information that can be represented by a low-degree multivariate polynomial over GF(2) of the secret and public variables of the target cryptosytem. Dinur and Shamir in [8] compared the cube attack to some of the previously known similar techniques [14, 16]. Recently, we have presented an extended variant of the cube attack in [1] to extract low-degree (mainly quadratic) sparse system of equations in addition to the linear equations obtainable from the original cube attack. In this paper, we employ these techniques together with fault analysis to build a hybrid attack against KATAN.

Previous Work. Cryptanalytical results on the KATAN family have been presented in [13, 3]. Recall that all three members of the KATAN family (i.e. KATAN32, KATAN48, and KATAN64) have 254 rounds. Knellwolf et al. [13] presented partial key recovery attacks (called "conditional differential cryptanalysis") against 78 rounds of KATAN32, 70 rounds of KATAN48, and 68 rounds of KATAN64 and concluded that the full versions of these ciphers seem to have sufficiently large number of rounds (254 rounds) to provide a confident security margin against their proposed attack. Bard et al. [3] presented cube attacks against 60, 40, and 30 rounds, and algebraic attacks against 79, 64, 60 rounds of KATAN32, KATAN48 and KATAN64, respectively. They also showed a side channel attack against the full 254 rounds of KATAN32, which has been the only attack against a full-round member of the KATAN family, so far. Bard et al.'s attack against the full-round KATAN32 combines the cube attack technique with a side channel attack model; namely, it assumes that adversary can obtain one bit of information from the internal state of the cipher and this one-bit information leakage must be *error free*. Bard et al. stated that such information is supposed to be captured by some side channels; for example, power or timing analysis or electromagnetic emanation, but we note that such measurements are not error (noise) free in practice and it is not clear whether Bard et al's attack can be adapted to handle such errors. Another way to capture such information leakage (albeit again hardly error free) is to use intrusive probing techniques which are expensive and usually are destructive to the underlying device. We also note that the Bard et al.'s attack is not a fault attack. The idea behind a fault attack, as introduced by Boneh et al. [6], is that if a wrong (faulty) result is released from a cryptosystem (as well as the normal unfaulty results) then adversary can use that information to break the cryptosystem. (Bard et al. do not assume and do not use any faulty computations in their side channel model).

Our Contribution. We combine the cube attack [8] and its extended variant (as presented in our previous work) [1] with *fault analysis* to form successful hybrid attacks against the full-round versions of all three members of the KATAN family. To the best of our knowledge, this is the first time that the cube attack and its extended variants are combined with "*fault analysis*" to form a successful hybrid attack against a block cipher. We assume a single-bit transient fault injection model as our side channel model, where the adversary is supposed to be able to corrupt a single random bit of the internal state of the cipher and this fault induction process can be repeated (by resetting the cipher); i.e., the faults are transient rather than permanent.

First, we determine effective rounds for fault inductions by analyzing distributions of low-degree polynomial equations obtainable using the cube and extended cube attack methods. Then, we show how to identify the exact position of faulty bits within the internal state by precomputing difference characteristics for each bit position at a given round and comparing these characteristics with ciphertext differences during the online phase of the attack. Finally, we show how to recover a low-degree (linear and quadratic) system of multivariate polynomial equations in the internal state and subkey bits that are easily solvable. The complexity of our attack on KATAN32 is 2^{59} and it requires about 115 fault injections. For KATAN48 and KATAN64, the attack requires 2^{55} computations (for both variants), while the required number of fault injections is 211 and 278, respectively.

Our fault attack on KATAN32 turns out to need about 2^{8} times more (off-line) operations compared to the previous side channel attack by Bard et al. [3] which requires 2^{51} computations; nevertheless, our attack model (namely, the transient fault injection at random bit positions in the internal state) is essentially *different* from the (noise free) information leakage assumption by Bard et al. in [3], and is arguably more practical as supported by previously known results such as [15]. Furthermore, our attack is directly adapted to the cases of KATAN48 and KATAN64 (both requiring 2^{55} computations) and, so far, is the only attack against the latter variants of KATAN in the side channel attack model.

2 A Brief Description of KATAN

KATAN is a family of block ciphers [7] consisting of three variants, namely: KATAN32, KATAN48 and KATAN64. Each variant accepts an 80-bit secret key and performs 254 rounds to produce a ciphertext. All variants also share the same key schedule as well as the same nonlinear functions. KATAN ciphers aim at constrained environments such as hardware implementations with limited resources (power consumption, clock frequency and gate counts). KATAN32 with

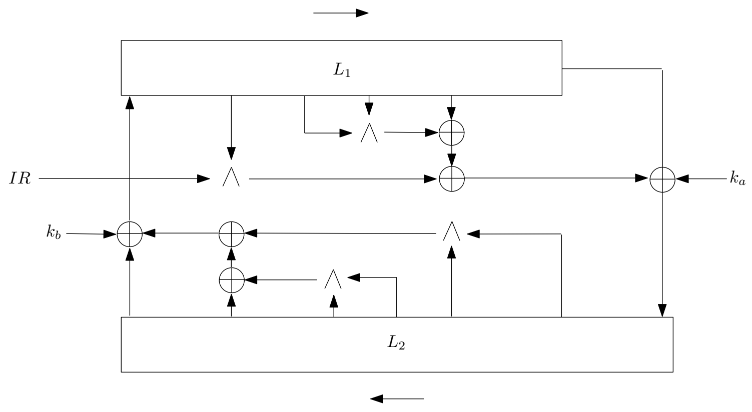

Fig. 1. The Outline of the KATAN Family of Block Ciphers

block size of 32 bits is the lightest variant in the family. A 32-bit plaintext block is loaded into two registers L_1 and L_2, respectively, of length 13 and 19 bits. The bits are indexed in the right-to-left order, from 0 to 12 for L_1 (i.e. $L_1 = (L_1[12], \cdots, L_1[0])$) and from 0 to 18 for L_2 (i.e. $L_2 = (L_2[18], \cdots L_2[0])$). The least significant bit (LSB) of the plaintext block is loaded to bit 0 of register L_2 followed by the other bits until the 18-th bit, and then remaining bits are loaded into register L_1 until the most significant bit (MSB) of the plaintext is loaded into bit 12 of register L_1. One round of KATAN32 consists of shifting the register L_1 and L_2 one bit to the left, and computing two new bit values using nonlinear functions f_a and f_b, respectively. These new bits are then loaded into the LSB bits of registers L_2 and L_1, respectively. The nonlinear functions f_a and f_b are defined as follows:

$$f_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus (L_1[x_5] \cdot IR) \oplus k_a \qquad (1)$$

$$f_b(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] \cdot L_2[y_4]) \oplus (L_2[y_5] \cdot L_2[y_6]) \oplus k_b \qquad (2)$$

where IR specifies an irregular update rule (i.e. $L_1[x_5]$ is used only when $IR = 1$), and k_a and k_b are two subkey bits. We refer to [7] for the details on the irregular update rules (IRs) for each round.

The key schedule for all variants of KATAN expands an 80-bit secret key K to 508 subkey bits using the following linear mapping

$$k_i = \begin{cases} K_i, & \text{for } 0 \leq i \leq 79, \quad (3) \\ k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13}, & \text{otherwise} \quad (3') \end{cases}$$

Given the precomputed subkey values, the values of k_a and k_b for a particular round t are defined as k_{2t} and k_{2t+1}, respectively. Thus the subkey for round t is defined as $k_a||k_b = k_{2t}||k_{2t+1}$. The selection for tap positions, x_is ($1 \leq i \leq 5$) and y_js ($1 \leq j \leq 6$), and the length of registers L_1 and L_2 are defined independently for each variant as shown in Table 1. Besides the tap positions and the length

Table 1. Parameters for the KATAN Family of Block Ciphers

Cipher	$\|L_1\|$	$\|L_2\|$	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

of the registers, the difference between all the three variants is the number of times the nonlinear functions f_a and f_b are applied in each round using the same subkey. One round of KATAN48 is shifting the registers L_1 and L_2 two bits to the left (i.e. requires two clock cycles). In each shift within the same round, the function f_a and f_b are applied using the same subkey $k_a||k_b$. Hence, full round of KATAN48 requires 508 clock cycles (i.e. 254 rounds $\times$ 2 clocks per round) to produce the ciphertext.

In contrast, one round of KATAN64 requires the registers L_1 and L_2 to be shifted three bits to the left (i.e. requires three clock cycles). Similarly, in each shift within the same round, the function f_a and f_b are applied using the same subkey $k_a||k_b$. As a result, the full round KATAN64 requires 762 clock cycles to produce the ciphertext. Fig. 1 shows the generic structure of the KATAN family of block ciphers. The initial state of KATAN-v (for v=32, 48, 64) is denoted by $IS = (s_{v-1}, \cdots, s_1, s_0) = L_1||L_2$ for the associated L_1 and L_2 registers.

3 An Overview of the Cube and Extended Cube Attacks

The main idea underlying the cube attack [8] is that the multivariate "master" polynomial $p(v_1, \cdots, v_m, k_1, \cdots, k_n)$, representing an output bit of a cryptosystem over GF(2) of secret variables k_i (key bits) and public variables v_i (i.e. plaintext or initial values), may inject algebraic equations of low degrees, in particular *linear* equations. The cube attack provides a method to derive such lower degree (especially linear) equations, given the master polynomial only as a black-box which can be evaluated on the secret and public variables.

Let's ignore the distinction between the secret and public variables' notations and denote all of them by $x_i, \cdots, x_\ell$, where $\ell = m + n$. Let $I \subseteq \{1, ..., \ell\}$ be a subset of the variable indexes, and t_I denote a monomial term containing multiplication of all the x_is with $i \in I$. By factoring the master polynomial p by the monomial t_I, we have:

$$p(x_1, \cdots, x_\ell) = t_I \cdot p_{S(I)} + q(x_1, \cdots, x_\ell) \tag{4}$$

where $p_{S(I)}$, which is called the *superpoly* of t_I in p, does not have any common variable with t_I, and each monomial term t_J in the residue polynomial q misses at least one variable from t_I. A term t_I is called a "*maxterm*" if its superpoly in p is linear polynomial which is not a constant, *i.e.* $deg(p_{S(I)}) = 1$.

The main observation of the cube attack is that, the summation of p over t_I, i.e. by assigning all the possible combinations of $0/1$ values to the x_is with $i \in I$ and fixing the value of all the remaining x_is with $i \notin I$, the resultant polynomial equals $p_{S(I)}$ (mod 2). Given access to a cryptographic function with public and secret variables, this observation enables an adversary to recover the value of the secret variables (k_is) in two steps, namely the preprocessing and online phases.

During the preprocessing phase, the adversary first finds sufficiently many maxterms, i.e. t_Is, such that each t_I consists of a subset of public variables $v_1, \cdots, v_m$. To find the maxterms, the adversary performs a probabilistic linearity test (such as the BLR test of [4]) on $p_{S(I)}$ over the secret variables $k_i \in \{k_1, \cdots, k_n\}$ while the value of the public variables not in t_I are fixed (to 0 or 1) (cf. [8] for more details).

Then the next step is to derive linearly independent equations in the secret variables k_is from $p_{S(I)}$ that are closely related to the master polynomial p, such that, solving them enables the adversary to determine the values of the secret variables. Once sufficiently many linearly independent equations in the secret variables are found, the preprocessing phase is completed. In the online phase, the adversary's aim is to find the value of the right-hand side of each linear equation by summing the black box polynomial p over the same set of maxterms t_Is which are obtained during the preprocessing phase. Now, the adversary can easily solve the resultant system of the linear equations, e.g. by using the Gaussian elimination method, to determine the values of the secret (key) variables.

A generalized variant of the cube attack, called extended cube, has been shown in [1] for extracting "low-degree nonlinear" equations efficiently. It revises the notion of tweakable polynomials from the original cube attack as

$$p(x_1, ..., x_\ell) = t_I \cdot X_K \cdot p_{S(I \cup K)} + q(x_1, ..., x_\ell) \tag{5}$$

where t_I is a subterm of size s over x_is with $i \in I$; X_K is a subterm of size r over x_is with $i \in K$, and $p_{S(I \cup K)}$ is the superpoly of $t_I \cdot X_K$ in p. Note that since both subterms t_I and X_K are factored out from p, the superpoly $p_{S(I \cup K)}$ does not contain any common variable with t_I and X_K, and each term t_J in the residue polynomial q misses at least one variable from $t_I \cdot X_K$. Now using the main observation of the cube attack, the summation of p over '$t_I \cdot X_K$', by assigning all the possible combinations of $0/1$ values to the x_is with $i \in I \cup K$ and fixing the value of all the remaining x_is with $i \notin I \cup K$, the resultant polynomial equals to $p_{S(I \cup K)}$ (mod 2).

The only difference between the original cube attack and the extended cube attack is in the preprocessing phase; the online phase for both of the methods are the same. During the preprocessing phase of the extended cube attack, the adversary finds many monomials t_Is, such that each t_I consists of a subset of public variables $v_1, \cdots, v_m$, and the corresponding superpoly $p_{S(I)}$ is a polynomial of degree D. To find those t_Is, the adversary performs the generalized version of the BLR test as proposed by Dinur and Shamir in [8] on $p_{S(I)}$ over the secret variables $k_1, \cdots, k_n$.

To derive efficiently a nonlinear equation $p_{S(I)}$ of degree D over secret variables k_is, the adversary should identify the subset $S \subseteq \{1, \cdots, n\}$ that consists of the secret variable indexes within $p_{S(I)}$, in which each k_i with $i \in S$ is either a term or a subterm of $p_{S(I)}$. To do this, the subterm X_K (cf. equation (5)) is assigned with each secret variable $k_i \in \{k_1, \cdots, k_n\}$ one at a time while the subterm t_I is fixed to the monomial in which its superpoly $p_{S(I)}$ is of degree D, and all public variables v_is with $i \notin I$ are fixed to 0 or 1. For each assignment of X_K, the adversary chooses κ sets of vector $\mathbf{x} \in \{0,1\}^{n-1}$ representing samples of $n-1$ secret variables k_is with $i \notin K$ independently and uniformly at random, and verify that X_K (or similarly the secret variable k_i that is assigned to X_K) exists as a variable in the superpoly $p_{S(I)}$ if $p_{S(I \cup K)} = 1$ for at least an instance vector $\mathbf{x}$.

Having the set of secret variables k_is with $i \in S$ of the nonlinear superpoly $p_{S(I)}$ of degree D enables the adversary to derive the nonlinear equation over the secret variables by finding all terms of degrees $0, 1, \cdots, D$ within the superpoly equation. Suppose $N = |S|$ is the number of secret variables k_is with $i \in S$ of the superpoly $p_{S(I)}$ of degree D. To derive $p_{S(I)}$, firstly the adversary assigns the subterm X_K one at a time with a monomial indexed by a subset $K \in T$ where T is a set of cube indexes of monomials constructed from all combinations of k_is from degree 1 until degree D with $i \in S$. In each assignment, all $v_i, k_i \notin t_I \cdot X_K$ are set to zero. Then to verify the existence of the monomial $X_K \in T$ as a term in $p_{S(I)}$, the adversary sums p over the monomial $t_I \cdot X_K$. If the result is equal to 1, then with probability 1, X_K is a term in the superpoly $p_{S(I)}$. Finally, the existence of a constant term (i.e. a term of degree 0) in the superpoly $p_{S(I)}$ is also determined by setting all public variables, v_is, for $i \notin I$ and all secret variables $k_1, \cdots, k_n$ to zero, and sum the polynomial p over t_I. Similarly, if the result is equal to 1, then with probability 1, a constant term exists within the superpoly $p_{S(I)}$.

4 Fault Analysis of KATAN

We simulate a fault attack assuming that the adversary can cause one transient single-bit error at a time in the internal state during the encryption/decryption process. It is assumed that the adversary can choose the target round(s) in which faults should be injected, for example, based on the side channel information inferred from power consumption traces and/or the clocking sequence (e.g., this can be done by triggering a laser beam with the target number of clocks of the cryptographic module). However, it is assumed that adversary cannot influence the exact position of the faulty bit within the internal state; he can only inject the fault randomly with the hope that it will hit the target bit positions by chance.

Using this fault model, our aim is to recover the 80-bit secret key used in KATAN. Our attack consists of two phases, namely offline and online phases. During the offline phase, firstly we identify the rounds (of the enciphering process) that can provide linear and quadratic equations due to single-bit faults. We call such rounds as *effective rounds* for our fault attacks.

Next, we determine the position of the faulty bits within the internal state using *difference characteristics* which we construct using the cube methods. Given a faulty ciphertext resulting from a random fault injection into an "unknown" internal state bit s_{j+t} after t-th round, to determine the position j, first we compute the ciphertext differential, Δc, by XORing (summing modulo 2) the non-faulty ciphertext c with the faulty ciphertext c' such that $\Delta c_j = c_j \oplus c'_j$, for $0 \leq j < |L_1| + |L_2|$. Then, guided by the lookup table, we refer to positions with values '0' and '1' (and ignore those with a '-' sign) within each characteristic and compare them with bits in the same positions in Δc. If all the corresponding bits in Δc match the bits in the characteristic of the faulty bit s_{j+t} then we can ensure that a fault has been injected into the bit at position j.

Finally, we extract a *low-degree* system of multivariate polynomial equations which are obtainable within the effective rounds using the difference between faulty and non-faulty ciphertexts facilitate by the cube and extended cube methods. More precisely, we only concentrate on extracting simple independent linear and quadratic equations that are easily solvable. After having a sufficient number of independent equations, we determine the target internal state bit positions for fault injections.

Knowing both the rounds and bit positions to be aimed, next the adversary moves to the online phase. In the online phase, the adversary repeatedly clocks the ciphers from the beginning until achieving one of the target rounds. Upon achieving this round, the adversary randomly injects a fault to the internal state with the hope to effect one of the target bits by chance. As the fault injection might not hit any one of the target bits, and the effect of one injection should be mutually exclusive from the effect of other injections, the error caused by the injections need to be transient rather than permanent. This is to enable the adversary to inject faults into the internal state repeatedly until all the target bits of each target round have been injected successfully. The aim of the online phase is to determine the value of right hand side of each equation obtained during the offline phase. Having the value of right-hand side of the equations, enables the adversary to recover the subkey bits provided by the key schedule algorithm. The knowledge about the value of the subkey bits enables the adversary to exploit the key schedule algorithm to recover the 80-bit secret key.

Our attack on the KATAN ciphers exploits the observation that after recovering n neighboring "subkey bits", the adversary can recover the 80-bit "secret key" with time complexity of 2^{80-n} computations. This is because the 80-bit secret key is directly loaded into an 80-bit LFSR (the key register) and the subkey bits for round $t > 79$ are computed using a linear update function and shift operations (cf. Equation 3 and Equation $3'$). Therefore, at any round $t > 79$, if we can recover the value of some of the LFSR bits (or similarly the value of the subkey bits), we can guess the remaining $80 - n$ values of the LFSR internal state bits and iteratively clock the LFSR backward until round $t = 0$ to recover the secret key. Suppose the highest and the lowest index values of the subkey bits to be recovered are H and L respectively. Hence, our aim is to recover the subkey bits such that $H - L \leq 79$, as all subkey bits between these index range will be the content of the 80-bit LFSR at a particular round t.

4.1 Attack on KATAN32

To apply differential fault attack on KATAN, first we determine the effective rounds for fault injection. To do this, we find the number of linear and quadratic equations within each round during the offline phase using cube methods. For each faulty round t, we consider each internal state bit as the monomial t_I (cf. equation 4) one at a time, and each bit of the ciphertext as a polynomial p over the internal state bits at round t. Next we detect the linear and quadratic equations by applying linearity and quadraticity tests using the BLR and generalized BLR tests as described in Section 3. Each time an equation (linear and quadratic) being detected, we accumulate the number of equations accordingly until all the internal state bits of round t have been considered. We repeat this procedure for each round of all KATAN's variants. As a result, Fig. 2 is derived indicating the range of rounds that should be considered for fault injections. In the figure, "Faulty Round" denotes the number of rounds that the cipher has run before injecting a fault into the internal state.

In order to examine the actual internal state position of faulty round t which has been affected by a random fault injection, the difference characteristic for each of the internal state bit of round t needs to be constructed. To construct the difference characteristics, we select each of the internal state bit of round t as the monomial, t_I, one at a time and apply the BLR linearity test on the corresponding superpoly, $p_{S(I)}$, to determine whether the test will result constant 0, constant 1, linear or higher degree superpoly. Constant 0 and constant 1 superpolys indicate values '0' and '1' in the difference characteristic bits, respectively. However linear and higher degree superpolys indicate unknown values in the characteristic bits, i.e. the '-' sign. Table 6 in Appendix shows an example of difference characteristics for KATAN32 for faulty round $t = 237$.

The fault attack can be efficiently applied if the rounds that have high number of quadratic equations are considered. As for KATAN32, this refers to the fault injections after $t = 237$ rounds as shown in Fig. 2. Considering this faulty round, we provide a sample set of linear and quadratic equations that can help in recovering the target subkey bits as shown in Table 3 in Appendix.

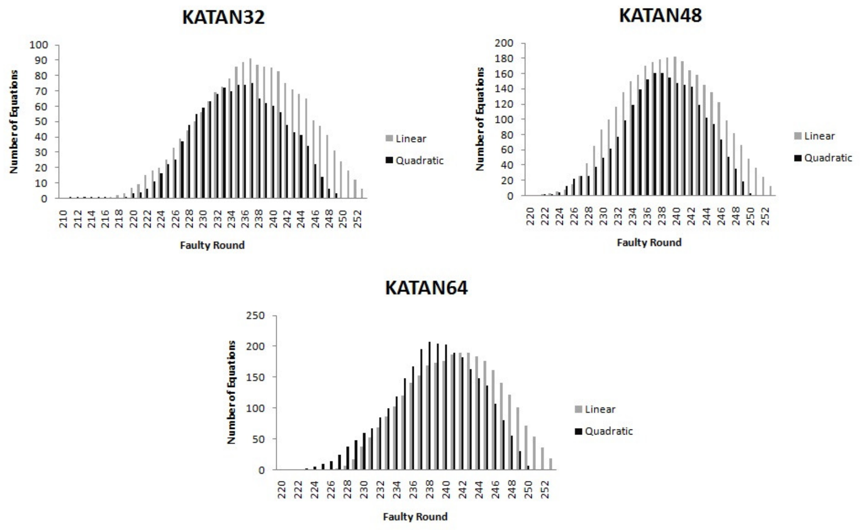

Fig. 2. Distribution of Linear and Quadratic Equations in KATAN

In the table, $L_2 = (s_{18+t}, \cdots, s_{0+t})$ and $L_1 = (s_{31+t}, \cdots, s_{19+t})$. Δc_j denotes a ciphertext bit difference where the difference is obtained by XORing the non-faulty ciphertext bit c_j with the faulty ciphertext bit c'_j, i.e. $\Delta c_j = c_j \oplus c'_j$, for $0 \leq j \leq 31$. For subkey bits we use a slightly different notation to facilitate our analysis, in which we denote the k_is as subkey bits whose indexes range from $0 \leq i \leq 507$ (in which bits indexed 0 until 79 are from the original secret key bits). We do not consider each subkey bit indexed $i > 79$ as a boolean function over the 80 secret key bits. Instead, to facilitate our analysis we only consider each one of them as an independent new variable.

Considering fault injection after $t = 237$ rounds, 10 subkey bits can be found within the quadratic equations, i.e. $k_{474}, \ldots, k_{482}$ and k_{484} (cf. Table 3 for the polynomial equations and Table 6 for the difference characteristics in Appendix). Recovering these subkey bits, requires solving the corresponding quadratic equations in which some of the linear equations listed in the table should also be involved, as they can provide the solution for the internal state bits of registers L_1 and L_2 within the quadratic equations. For example, to find the solution for k_{474}, we consider s_{1+t} as the faulty bit after $t = 237$ rounds. Considering the difference between non-faulty and faulty ciphertext bit c_{24}, i.e. Δc_{24}, the symbolic representation of the differential is

$$s_{22+t} + s_{26+t} + s_{31+t} + k_{474} + s_{24+t}s_{27+t} = \Delta c_{24}. \quad (6)$$

The value of the right hand side (RHS) of this equation (either 0 or 1) can be determined by numerically computing Δc_{24}, such that $\Delta c_{24} = c_{24} \oplus c'_{24}$. To recover the value of k_{474} for example, requires the values of all other bits within the equation to be known. If there exist a case in which the value of certain bits cannot be recovered considering the equations derived from the faulty round t only, the adversary needs to consider earlier rounds to find equivalent bits since the value of the internal state bits are only shifted from the LSB to MSB except the LSB bits. Tables 2–5 in Appendix show the set of equations that are solvable and resulting from faulty rounds $t = 231, 237, 243$ and 249, respectively. Note that, comparing more earlier rounds results in having difficulty to determine the faulty bit positions within register L_1 and L_2. This is because the uniqueness of the difference characteristics will slowly disappear as we consider earlier rounds.

4.2 Attack on KATAN48

Following the method used on KATAN32, we consider the KATAN48 block cipher as our next target. Since KATAN48 requires two clocks for each round, if a certain internal state bit s_{j+t} cannot be solved directly in certain faulty round t, then its solution may be found by referring to bit s_{j-2n+t} in an earlier faulty round $t-n$, for $j - 2n \geq 29$ and $31 \leq j \leq 47$, or $j - 2n \geq 0$ and $2 \leq j \leq 28$.

Our attack on KATAN48 considers faulty rounds $t = 234, 238, 242, 246$ and 250 as the target rounds. Similar to the analysis of KATAN32, the selection of these rounds is based on the number of quadratic equations that can be found within the effective rounds. Fig. 2 shows that the highest number of quadratic equations for KATAN48 can be found at faulty rounds $t = 237$ and $t = 238$. Since the difference characteristics are more clearly defined when we consider later rounds, we choose $t = 238$ rather than $t = 237$ as our first target round. Table 8 in Appendix shows the polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after $t = 238$ rounds of KATAN48. Table 7 in Appendix shows the equations obtained using fault induction after $t = 234$ rounds of KATAN48. For equations obtainable using fault induction after $t = 242, 246, 250$ rounds of KATAN48 we refer to the full version of this paper in [2].

4.3 Attack on KATAN64

Now we consider a fault attack on the third variant of the KATAN block cipher, namely, KATAN64. In KATAN64 we have $L_2 = (s_{38+t}, \cdots, s_{0+t})$ and $L_1 = (s_{63+t}, \cdots, s_{39+t})$. Since each round in KATAN64 requires 3 clocks, if certain internal state bits s_{j+t} cannot be recovered at faulty round t, we can try to recover their values from bit s_{j-3n+t} of faulty round $t-n$, for $j - 3n \geq 39$ and $42 \leq j \leq 63$, or $j - 3n \geq 0$ and $3 \leq j \leq 38$. Our attack on KATAN64 considers faulty rounds $t = 236, 238, 242, 246$ and 250 as the target rounds. Equations obtainable using fault induction after these rounds are provided in the the full version of this paper in [2].

4.4 Attack Complexity

Result on KATAN32. Our experimental simulation of the attack on KATAN32 shows that 21 subkey bits from faulty rounds $t = 231, 237, 243, 249$ can be recovered, requiring collectively 20 specific internal state bit positions (regardless of the round number) to be considered as target faulty bits, as shown in Tables 2–5 in Appendix. The average number of fault injections needed to successfully hit these 20 target faulty bits is 115 (where the average is taken over 10,000 trials).

Since the highest index of the subkey bits is $H = 498$ and the lowest index is $L = 474$ (hence $H - L = 24 < 80$) the target subkey bits can be found in the 80-bit key register within rounds $209 \leq t \leq 236$. Therefore, to recover the secret key, we need to guess the remaining 59 bits of the key register and then to clock the key register backward until round $t = 0$. This reduces the complexity of the attack to 2^{59} computations compared to 2^{80} by exhaustive key search.

Result on KATAN48. The attack on KATAN48 results in recovering 25 subkey bits considering faulty rounds $t = 234, 238, 242, 246, 250$ which requires collectively 27 specific internal state bits positions to be considered as target faulty bits (refer to Tables 7 and 8 in Appendix and the full version of this paper in [2]). The average number of required fault injections to successfully hit these 27 target faulty bits is 211 (where the average is taken over 10,000 trials). The highest and the lowest subkey bit indexes are $H = 500$ and $L = 476$, respectively (hence $H - L = 24 < 80$), so all the subkey bits can be found within the content of the 80-bit key register at rounds $210 \leq t \leq 237$. Therefore, to recover the secret key we need to guess the remaining 55 bits of the key register and then to clock backward until round $t = 0$ to recover the secret key. Thus, finding the correct key requires 2^{55} computations in this attack.

Result on KATAN64. In attacking KATAN64 we consider faulty rounds $t = 236, 238, 242, 246, 250$ to recover (at least) the same 25 subkey bits as in the attack on KATAN48 which requires collectively 44 specific internal state bit positions to be faulty (refer to the full version of this paper in [2]). The average number of required fault injections to successfully hit these 44 target faulty bits is 278 (where the average is taken over 10,000 trials). This results in an attack with complexity 2^{55} (Noticing that the highest index of the subkey bits is $H = 491$ and the lowest index is $L = 476$ (i.e. $H - L = 15 < 80$); hence, these 25 target subkey bits can be found in the 80-bit secret key register and we only need to guess the remaining 55 bits of the key register).

5 Conclusion

In this paper, we showed fault attacks using a transient single-bit fault model against all three members of the KATAN family; namely, KATAN32, KATAN48 and KATAN64. Our attacks employ the cube attack and its extensions to determine the effective fault injection rounds, to generate the difference characteristics and to generate linear and quadratic equations. The complexity of our attack on KATAN32 is 2^{59} computations and about 115 fault injections. For KATAN48 and KATAN64, the attack requires 2^{55} computations (for both variants), while the required number of fault injections is 211 and 278, respectively. Our fault attacks on KATAN48 and KATAN64, so far, are the only attacks against the full-round versions of these ciphers.

Acknowledgments. We thank Flavio D. Garcia and the anonymous reviewers of ISPEC 2012 for their constructive comments and suggestions.

References

[1] Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: Extended Cubes: Enhancing the Cube Attack by Extracting Low-Degree Non-Linear Equations. In: Cheung, B. et al. (Eds.) ASIACCS 2011. ACM, pp. 296–305 (2011)

[2] Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: Fault Analysis of the KATAN Family of Block Ciphers. Cryptology ePrint Archive: Report 2012/030. (Full version of this paper.)

[3] Bard, G.V., Courtois, N.T., Jr, J.N., Sepehrdad, P., Zhang, B.: Algebraic, AIDA/Cube and Side Channel Analysis of KATAN Family of Block Ciphers. In: Gong, G., Gupta, K.C. (Eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 176–196. Springer, Heidelberg (2010)

[4] Blum, M., Luby, M., Rubinfield, R.: Self-Testing/Correcting with Application to Numerical Problems. In: STOC, pp. 73–83. ACM, New York (1990)

[5] Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski, B.S (Ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)

[6] Boneh, D., DeMillo, R., Lipton, R.: On the Importance of Checking Cryptographic Protocols for Faults. In: Fumy, W. (Ed.) EUROCRYPT 1997. LNCS, vol. 1223, pp. 37–51. Springer, Heidelberg (1997)

[7] de Cannière, C., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (Eds.) CHES 2009. LNCS, vol. 5754, pp. 272–288. Springer, Heidelberg (2009)

[8] Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (Ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)

[9] Hoch, J.J., Shamir, A.: Fault Analysis of Stream Ciphers. In: Joye, M., Quisquater, J.-J. (Eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg (2004)

[10] Hojsík, M., Rudolf, B.: Differential Fault Analysis of Trivium. In: Nyberg, K. (Ed.) FSE 2008. LNCS, vol. 5086, pp. 158–172. Springer, Heidelberg (2008)

[11] Hojsík, M., Rudolf, B.: Floating Fault Analysis of Trivium. In: Chowdhury, D.R., Rijmen, V., Das, A. (Eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 239–250. Springer, Heidelberg (2008)

[12] Hu, Y., Zhang, F., Zhang, Y.: Hard Fault Analysis of Trivium. Cryptology ePrint Archive, Report 2009/333 (2009)

[13] Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In: Abe, M. (Ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)

[14] Lai, X.: Higher Order Derivatives and Differential Cryptanalysis. In: Communication and Cryptology, pp. 227–233. Kluwer Academic Publisher (1994)

[15] Skorobogatov, S.P., Anderson, R.J.: Optical Fault Injection Attacks. In: Kaliski Jr, B.S., Koç, C.K., Paar, C. (Eds.) CHES 2002. LNCS, vol. 2523, pp. 31–48. Springer, Heidelberg (2002)

[16] Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. IACR ePrint Archive, Report 2007/413 (2007), http://eprint.iacr.org/2007/413

[17] Vielhaber, M.: AIDA Breaks BIVIUM (A&B) in 1 Minute Dual Core CPU Time. Cryptology ePrint Archive, Report 2009/402, IACR (2009)

A Appendix

Table 2. Polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after $t = 231$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{8+t}	Δc_7	s_{10+t}
s_{9+t}	Δc_8	s_{11+t}
s_{10+t}	Δc_9	s_{12+t}
s_{11+t}	Δc_{17}	s_{9+t}
s_{19+t}	Δc_{28}	s_{22+t}
s_{20+t}	Δc_{29}	s_{23+t}
s_{21+t}	Δc_{30}	s_{24+t}
s_{22+t}	Δc_{31}	s_{25+t}

Table 3. Polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after $t = 237$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{1+t}	Δc_{28}	$s_{19+t} + s_{23+t} + s_{28+t} + k_{480} + s_{21+t}s_{24+t}$
	Δc_{24}	$s_{22+t} + s_{26+t} + s_{31+t} + k_{474} + s_{24+t}s_{27+t}$
	Δc_{6}	s_{6+t}
	Δc_{4}	$s_{4+t} + s_{15+t} + k_{481} + s_{0+t}s_{5+t} + s_{7+t}s_{9+t}$
s_{2+t}	Δc_{29}	$s_{20+t} + s_{24+t} + s_{29+t} + k_{478} + s_{22+t}s_{25+t}$
	Δc_{27}	s_{4+t}
	Δc_{25}	s_{0+t}
	Δc_{5}	$s_{5+t} + s_{16+t} + k_{479} + s_{1+t}s_{6+t} + s_{8+t}s_{10+t}$
s_{3+t}	Δc_{30}	$s_{25+t} + s_{30+t} + k_{476} + s_{23+t}s_{26+t}$
	Δc_{28}	s_{5+t}
	Δc_{26}	s_{1+t}
	Δc_{12}	s_{8+t}
	Δc_{6}	$s_{6+t} + s_{17+t} + k_{477} + s_{2+t}s_{7+t} + s_{9+t}s_{11+t}$
s_{4+t}	Δc_{27}	s_{2+t}
	Δc_{7}	$s_{7+t} + s_{18+t} + k_{475} + s_{3+t}s_{8+t} + s_{10+t}s_{12+t}$
s_{5+t}	Δc_{30}	s_{7+t}
	Δc_{28}	s_{3+t}
	Δc_{21}	$s_{21+t} + s_{26+t} + k_{484} + s_{19+t}s_{22+t}$
	Δc_{8}	s_{19+t}
s_{9+t}	Δc_{2}	s_{11+t}
s_{10+t}	Δc_{12}	s_{12+t}
s_{11+t}	Δc_{7}	s_{9+t}
s_{12+t}	Δc_{12}	s_{10+t}
s_{19+t}	Δc_{22}	s_{22+t}
s_{20+t}	Δc_{23}	s_{23+t}
s_{21+t}	Δc_{24}	s_{24+t}
s_{22+t}	Δc_{25}	s_{25+t}
s_{23+t}	Δc_{26}	s_{26+t}
	Δc_{12}	s_{20+t}
s_{24+t}	Δc_{27}	s_{27+t}
	Δc_{20}	$s_{7+t} + s_{18+t} + s_{22+t} + s_{27+t} + k_{475} + k_{482} + s_{3+t}s_{8+t} + s_{10+t}s_{12+t} + s_{20+t}s_{23+t} + 1$
	Δc_{13}	s_{21+t}

Table 4. Polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after $t = 243$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{0+t}	Δc_{21}	$s_{22+t} + s_{27+t} + k_{494} + s_{20+t}s_{23+t}$
s_{1+t}	Δc_{27}	s_{6+t}
	Δc_{22}	$s_{23+t} + s_{28+t} + k_{492} + s_{21+t}s_{24+t}$
	Δc_{20}	s_{3+t}
s_{2+t}	Δc_{28}	s_{7+t}
	Δc_{23}	$s_{24+t} + s_{29+t} + k_{490} + s_{22+t}s_{25+t}$
	Δc_{21}	s_{4+t}
	Δc_{19}	s_{0+t}
s_{3+t}	Δc_{24}	$s_{25+t} + s_{30+t} + k_{488} + s_{23+t}s_{26+t}$
	Δc_{22}	s_{5+t}
	Δc_{20}	s_{1+t}
	Δc_{0}	$s_{6+t} + s_{17+t} + k_{489} + s_{2+t}s_{7+t} + s_{9+t}s_{11+t}$
s_{4+t}	Δc_{25}	$s_{26+t} + s_{31+t} + k_{486} + s_{24+t}s_{27+t}$
	Δc_{21}	s_{2+t}
	Δc_{1}	$s_{7+t} + s_{18+t} + k_{487} + s_{3+t}s_{8+t} + s_{10+t}s_{12+t}$
s_{18+t}	Δc_{4}	s_{21+t}
	Δc_{1}	$s_{4+t} + s_{15+t} + k_{493} + s_{0+t}s_{5+t} + s_{7+t}s_{9+t}$
s_{19+t}	Δc_{5}	s_{22+t}
	Δc_{2}	$s_{5+t} + s_{16+t} + k_{491} + s_{1+t}s_{6+t} + s_{8+t}s_{10+t}$
s_{21+t}	Δc_{7}	s_{24+t}
s_{22+t}	Δc_{8}	s_{25+t}
	Δc_{5}	s_{19+t}
s_{23+t}	Δc_{9}	s_{26+t}
	Δc_{6}	s_{20+t}
s_{24+t}	Δc_{10}	s_{27+t}
	Δc_{7}	s_{21+t}
s_{26+t}	Δc_{20}	$s_{21+t} + s_{23+t} + s_{31+t} + k_{486} + k_{496} + s_{19+t}s_{22+t} + s_{24+t}s_{27+t} + 1$
	Δc_{9}	s_{23+t}

Table 5. Polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after $t = 249$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{4+t}	$\varDelta c_{19}$	$s_{22+t} + s_{26+t} + s_{31+t} + k_{498} + s_{24+t}s_{27+t}$
s_{5+t}	$\varDelta c_{20}$	s_{0+t}
s_{21+t}	$\varDelta c_{1}$	s_{24+t}
s_{23+t}	$\varDelta c_{3}$	s_{26+t}
	$\varDelta c_{0}$	s_{20+t}
s_{25+t}	$\varDelta c_{2}$	s_{22+t}

Table 6. Difference characteristics for KATAN32 (faulty round t=237). '0' and '1' denote differential values 0 and 1 of the corresponding ciphertext bit differential $\varDelta c_j$, and '-' denotes an unknown differential value.

	Δc_j																															
s_{j+t}	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	-	0	-	0	-	-	0	-	-	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	1	-
1	0	0	1	-	0	-	0	-	-	0	-	-	-	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	1	-	-
2	0	1	-	0	-	0	-	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	1	-	-	-
3	1	-	0	-	-	-	-	0	-	-	-	-	-	0	0	0	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
4	-	0	-	0	-	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	-	-	1	-	-	-	-	-
5	0	-	0	-	0	0	0	0	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	-	-	1	-	-	-	-	-	-
6	-	0	-	0	0	0	0	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	-	-	1	-	-	-	-	-	-	0
7	0	-	0	0	-	0	-	-	-	-	-	-	-	0	0	0	0	0	0	1	0	-	-	1	-	-	-	-	-	-	-	-
8	-	0	0	0	-	0	-	0	-	-	-	-	-	0	0	0	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
9	0	0	0	0	0	1	0	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	0	-	-	-
10	0	0	0	0	-	0	-	0	-	-	-	-	-	0	0	0	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
11	0	0	0	1	0	0	0	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	-	0	-	-	0	0	-	-	-	-
12	0	0	1	0	-	0	-	0	-	-	-	-	-	0	0	0	0	0	0	-	0	-	0	-	-	0	0	-	-	-	-	-
13	0	1	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0
14	1	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0
15	0	0	0	0	0	0	0	0	0	-	0	-	-	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1
16	0	0	0	0	0	0	0	0	-	0	-	-	-	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1	0
17	0	0	0	0	0	0	0	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1	-	0
18	0	0	0	0	-	0	-	0	-	-	-	-	-	0	0	0	0	0	0	1	0	-	0	1	-	0	0	-	1	-	0	-
19	0	0	0	-	0	-	0	-	-	-	-	1	-	0	0	0	0	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-
20	0	0	-	0	-	0	-	-	-	-	1	-	-	0	0	0	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-	-
21	0	0	0	-	0	-	-	-	-	1	-	-	-	0	0	0	0	0	-	0	1	-	0	0	0	1	0	0	-	0	-	-
22	-	0	-	0	-	-	-	-	1	-	-	-	-	0	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-	-	-	-
23	0	-	0	-	-	-	-	1	-	-	-	-	-	0	0	0	-	0	1	-	0	0	0	1	0	0	-	-	-	-	-	-
24	-	0	-	-	-	-	1	-	-	-	-	-	-	0	0	-	0	1	-	0	0	0	1	0	0	-	0	-	-	-	-	-
25	0	-	-	0	0	1	-	-	-	-	-	-	-	0	0	0	1	-	0	0	0	1	0	0	0	0	-	-	-	-	-	-
26	-	-	0	0	1	-	-	-	-	-	-	-	-	0	0	1	-	0	0	0	1	0	0	0	0	-	-	-	-	-	-	-
27	-	0	0	0	-	-	0	-	1	-	-	-	-	0	0	-	0	0	0	1	0	0	0	0	-	0	-	-	-	-	-	-
28	0	0	0	-	0	0	0	1	-	0	-	0	-	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-
29	0	0	-	0	0	0	1	-	0	-	0	-	-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-
30	0	-	0	0	0	1	-	0	-	0	-	-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-
31	-	0	0	0	1	-	0	-	0	-	-	0	-	0	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	1

Table 7. Polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after t=234 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{6+t}	$\varDelta c_{26}$	s_{15+t}
	$\varDelta c_{25}$	s_{14+t}
s_{9+t}	$\varDelta c_{28}$	s_{17+t}
s_{11+t}	$\varDelta c_{18}$	$s_{19+t} + 1$
s_{29+t}	$\varDelta c_{41}$	s_{37+t}
s_{30+t}	$\varDelta c_{42}$	s_{38+t}

Table 8. Polynomial equations obtained using the difference between non-faulty and faulty ciphertexts for fault induction after t=238 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{4+t}	Δc_9	$s_{12+t} + 1$
	Δc_{16}	s_{13+t}
s_{5+t}	Δc_{17}	s_{14+t}
s_{6+t}	Δc_{24}	s_{15+t}
s_{8+t}	Δc_{47}	s_{0+t}
	Δc_{13}	s_{16+t}
s_{9+t}	Δc_{14}	s_{17+t}
s_{10+t}	Δc_{15}	s_{18+t}
s_{11+t}	Δc_{16}	s_{19+t}
s_{12+t}	Δc_{17}	s_{20+t}
	Δc_{16}	s_{29+t}
	Δc_{15}	s_{3+t}
s_{13+t}	Δc_{17}	s_{30+t}
s_{14+t}	Δc_{18}	s_{31+t}
	Δc_{17}	s_{5+t}
s_{15+t}	Δc_{19}	s_{32+t}
	Δc_6	s_{7+t}
s_{16+t}	Δc_{20}	s_{33+t}
	Δc_{13}	s_{8+t}
s_{17+t}	Δc_{38}	$s_{29+t} + s_{35+t} + s_{41+t} + k_{482} + s_{30+t}s_{38+t}$
	Δc_{21}	s_{34+t}
	Δc_{14}	s_{9+t}
	Δc_{12}	$s_{16+t} + s_{25+t} + k_{479} + s_{3+t}s_{12+t} + s_{10+t}s_{18+t}$
s_{18+t}	Δc_{22}	s_{35+t}
	Δc_{15}	s_{10+t}
s_{19+t}	Δc_{46}	s_{1+t}
	Δc_{40}	$s_{31+t} + s_{37+t} + s_{43+t} + k_{480} + s_{32+t}s_{40+t}$
	Δc_{14}	$s_{18+t} + s_{27+t} + k_{477} + s_{5+t}s_{14+t} + s_{12+t}s_{20+t}$
s_{29+t}	Δc_{33}	s_{37+t}
s_{30+t}	Δc_{25}	s_{38+t}
	Δc_{17}	$s_{13+t} + s_{22+t} + k_{483} + s_{0+t}s_{9+t} + s_{7+t}s_{15+t}$
s_{31+t}	Δc_{36}	$s_{33+t} + s_{39+t} + s_{45+t} + k_{478} + s_{34+t}s_{42+t} + 1$
	Δc_{35}	s_{39+t}
	Δc_{18}	$s_{14+t} + s_{23+t} + k_{481} + s_{1+t}s_{10+t} + s_{8+t}s_{16+t}$
	Δc_7	s_{4+t}
	Δc_1	$s_{4+t} + s_{40+t} + s_{46+t} + k_{476} + s_{35+t}s_{43+t}$
s_{32+t}	Δc_{36}	s_{40+t}
s_{33+t}	Δc_{37}	s_{41+t}
s_{34+t}	Δc_{38}	s_{42+t}
s_{35+t}	Δc_{39}	s_{43+t}