Algebraic Fault Analysis of Katan

November 21, 2014

Frank-M. Quedenfeld[1]

[1]Ruhr University Bochum, Germany

frank.quedenfeld@googlemail.com

Abstract

This paper presents a new and more realistic model for fault attacks and statistical and algebraic techniques to improve fault analysis in general. Our algebraic techniques is an adapted solver for systems of equations based on ElimLin and XSL. We use these techniques to introduce two new fault attacks on the hardware oriented block cipher Katan32 from the Katan family of block ciphers. We are able to break full Katan using 4 faults and $2^{29.04}$ Katan evaluations with a theoretical statistical fault attack and 7.19 faults in $2^{27.2}$ Katan evaluations with a tested algebraic one. This is a great improvement over the existing fault attacks which need 115 and 140 faults respectively. Furthermore, our algebraic attack can be executed on a normal computer.

Keywords: Katan, ElimLin, equation solving over $\mathbb{F}_2$, fault analysis, algebraic fault attack, filter for improved guessing, differential fault attack

1 Introduction

Algebraic attacks against symmetric ciphers are more than a decade old. While being promising, they did not deliver exactly what was promised and were mostly dismissed in cryptanalytic literature.

Recently, the so-called "ElimLin" algorithm was used to attack several ciphers, in particular CTC2, LBlock and MIBS. According to [CSSV12] from FSE 2012, only 6 rounds can be broken for CTC2. This attack requires up to 180h on a standard PC and the authors guessed 210 bits and used 64 chosen cipher texts. Guessing 220 bits and 16 chosen plain text brings the attack down to 3h. An initial implementation of the ElimLin algorithm was employed on DES in [CB07]. Here, plain ElimLin could break 5 rounds of DES with 3 known plain texts and 23 guessed bits; using a SAT solver, this number can be increased to 6 rounds for an unspecified number of known plain texts (most likely 1) and 20 key bits fixed.

While ElimLin was not as successful, there were recent improvements made in [QW14]. The authors combined it among other things with a variant of eXtended Sparse Linearization (XSL) from [CP02, CKPS00]. The solver presented is able to handle a large number of variables and equations while not consuming as much memory as Gröbner basis algorithms [Fau02b, Fau02a, Alb10], especially when many variables are involved. However, ElimLin is less efficient which can be seen in [YC04b, YC04a, AFI+04, Die04]. Still, ElimLin united with a new variant of XSL, named SL, is fast and can handle many variables and equations.

The afore mentioned algorithms for solving systems of equations detect unsolvable systems quickly as described in [BFP09, FJ03, QW14]. That allows us to handle systems which we know may not be solvable.

More recent attacks on ciphers are fault attacks which were introduced in [BDL97, BS97]. When dealing with hardware devices like smartcards or RFID tags, we are able to acquire the device in question. Then we can perform a fault analysis as first described in [BDL97] and [BS97]. In [BDL97], the authors describe a fault analysis on public key primitives and in [BS97] a fault analysis on symmetric primitives is performed. Afterwards the authors of [HR08a] describe a differential fault analysis for a stream cipher with non-linear feedback shift register for the first time.

We want to apply a fault analysis to the hardware oriented block cipher Katan [CDK09]. It is based on two non-linear feedback shift registers and uses an 80-bit secret key. These shift registers are updated 254 rounds before an output is produced.

In a fault analysis, we inject an error into the cipher's internal state. With this error we can reduce the effective round number for a symmetric primitive or force the cipher to leak information about the secret key.

The fault model describes how the fault is injected. Previous literature (see for instance [HR08a, HR08b, MBB11, ALRSS12, SH13]) use a fault model which flips exactly one bit in the internal state of a cipher at an unknown random position. This is unrealistic for most cases. Even for Laser Fault Injection (LFI) that can be considered as the most locally precise method, the laser spot might hit neighboring bits as well, especially for the latest technology nodes. Furthermore we do not adjust the laser beam once we have used it. We calculate a characteristic with an algebraic solver for systems of equations over $\mathbb{F}_2$ which allows us to exactly determine the lasers effect for the last rounds of Katan, in which we inject faults.

Based on the new fault model and the given characteristic we build a filter for eliminating wrong guessed round key bits. With this we guess more efficiently and get the key faster than an exhaustive search, while using a low number of faults. Theoretically, we can get the full key with 4 faults in $2^{29.04}$ Katan computations.

Instead of guessing key bits we can represent Katan instances as a system of equations and try to solve it. When we have a characteristic we can also represent faulty Katan instances as a system of equations. We represent faulty Katan instances even with faults in lower rounds where we do not have a characteristic. By doing so we do not know if our system of equations is valid. Therefore, we try all possible systems of equations for our adjusted laser beam. The mentioned solvers detect systems of equations that are not valid quickly. So we build a filter for inconsistent systems of equations. We get a practical attack which can be executed with 7.19 faults on average and running time $2^{27.2}$ Katan computations.

1.1 Our contributions

We discuss how to obtain a realistic model for fault attacks on hardware oriented symmetric primitives. We state that we cannot use a one-bit standard model in most cases and propose a new model which is based on an electrical engineering point of view.

Secondly, we propose two techniques to attack cryptographic primitives in a fault attack scenario. In the first we build a filter to guess round key bits efficiently. We are able to find the key with 4 faults in $2^{29.04}$ Katan computations. While the first attack is quite theoretical, the second one uses a property from solvers for systems of equations. Namely, to determine whether a system is inconsistent or not. With that we are able to apply a practical attack which is fully tested on a standard computer. We are able to break Katan with 7.19 faults in $2^{27.2}$ Katan computations. This outperforms previous attacks on Katan (see [ALRSS12, SH13]) in a convincing fashion as they need 114 and 140 faults, respectively.

1.2 Organization

First we introduce the hardware oriented block cipher family Katan, in particular Katan32. Existing fault attacks on it with the corresponding fault model are described in Section 2. Afterwards we describe the algebraic representation and the solver which we use to launch an algebraic fault attack on Katan in Section 3. This is followed by a description and discussion of a realistic fault model that takes real world scenarios into account in Section 4. In Section 5 we perform a statistical attack that outperforms every previously known attack on Katan32. This attack is purely theoretical. In Section 6 we use a solver for quadratic equations over $\mathbb{F}_2$

to settle an algebraic fault attack on Katan32 which is practical. The paper concludes with a summary on these topics in Section 7.

2 Preliminaries

In this chapter we give a brief introduction to the Katan family of block ciphers and to Katan32 in particular. Afterwards, we introduce fault attacks on symmetric primitives in section 2.2.

2.1 Katan32

In [CDK09], a family of small and hardware oriented block ciphers is presented. Namely, the Katan family of block ciphers. There are three variants of Katan. The main difference is the block length of the ciphers. Katan is available with $32, 48$ and 64 bit block length. The key length is 80 bits for all of the ciphers. The variants with 48 and 64 bit length are updated twice / three times per round with the same round key. In the sequel we work with Katan32 and refer to it Katan for simplicity.

Katan works with addition and multiplication over $\mathbb{F}_2$. We consider the two shift registers $A := (A_i, \ldots, A_{i-12})$ and $B := (B_i, \ldots, B_{i-18})$. We initialize them with a plain text block $P = (p_0, \ldots, p_{31})$ to

$$\begin{aligned} A &= (p_{19}, \ldots, p_{31}) \\ B &= (p_0, \ldots, p_{18}). \end{aligned}$$

The two registers will be updated for 254 rounds according to the update function:

$$\begin{aligned} A_i &= B_{i-18} + B_{i-7} + B_{i-12} \cdot B_{i-10} + B_{i-8} \cdot B_{i-3} + K_{2r+1} \\ B_i &= A_{i-12} + A_{i-7} + A_{i-8} \cdot A_{i-5} + A_{i-3} \cdot f_i + K_{2r} \end{aligned}$$

with r being the current round and f and K are linear shift registers over F_2. The linear shift register f introduces the monomial A_{i-3} if $f_i = 1$ into the update function. It is an 8-bit shift register with update function

$$f_i = f_{i-1} + f_{i-3} + f_{i-5} + f_{i-7}.$$

It is initialized with $(1, \ldots, 1)$ and it is set back to this vector after 254 rounds. The key schedule K is another linear shift register. It is initialized with the key $(k_0, \ldots, k_{79})$ and updated according to the function

$$K_i = K_{i-79} + K_{i-60} + K_{i-49} + K_{i-12}\,.$$

The output of Katan is the cipher text block $Z = (z_0, \ldots, z_{31}) = B||A$ after 254 rounds.

In figure 1 the circuit of Katan is presented. To decipher a message we initialize the registers A and B with the cipher text $(c_0, \ldots, c_{31})$ to

$$\begin{aligned} A &:= (c_{19}, \ldots, c_{31}) \\ B &:= (c_0, \ldots, c_{18}) \end{aligned}$$

and store all round keys and irregular values from f in lists. Then we update the registers A and B 254 times according to the inverse update function

$$\begin{aligned} A_{i-12} &= s_B + A_{i-7} + A_{i-8} \cdot A_{i-5} + A_{i-3} \cdot f_r + K_{2R} \\ B_{i-18} &= s_A + B_{i-7} + B_{i-12} \cdot B_{i-10} + B_{i-8} \cdot B_{i-3} + K_{2R+1}. \end{aligned}$$

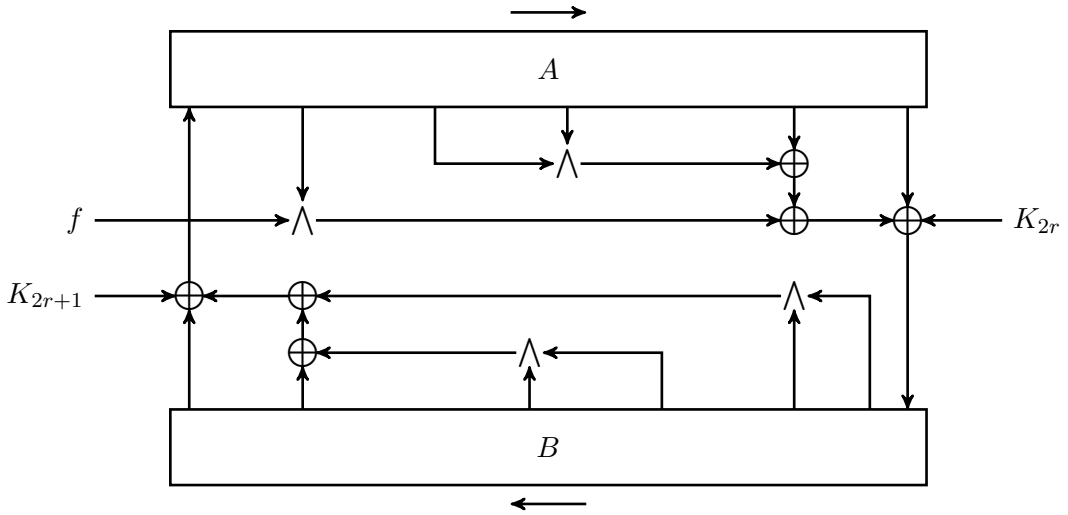

Figure 1: Structure of the encryption function of Katan

This time we update the register backwards. This means we save the first bits $s_A = A_i$ and $s_B = B_i$ and shift the registers to the left hand side. So we overwrite the first bit in the registers which we have stored before. Afterwards we update the registers according to the inverse update function and store the result as the last entry of the respective register.

In our experiments for the algebraic fault attack we measure time in seconds. For comparison to other fault attacks we calculate the equivalent in Katan computations. For this purpose we use the formula

$$T_p := \frac{D}{254}((2^g - 1)T_f + T_s)$$

with the throughput D of the cipher, the number g of guessed variables, T_f the time we need to falsify a wrong guess and T_s to calculate the right solution. We do not calculate the throughput with our implementation, but take the fastest known implementation. According to [CDK], this is $D = 1.0714 \times 10^9$ bps. One Katan computation takes $D/254$.

Technically we measure our used time in bits. But Katan computes one bit in one computation so our measurement is justified.

2.2 Previous fault attacks on Katan

When dealing with hardware devices like smartcards or RFID tags, we have a chance to attack the device itself. Then, we can do a fault analysis as previously described in [BDL97, BS97]. The authors of [BDL97] focused on public key cryptosystems while the authors of [BS97] focused on symmetric primitives.

In a fault attack, we assume that the adversary can cause an error in the internal state of a cipher during the encryption / decryption process. This error can force internal state bits to one or zero or flip bits depending on the fault model used.

Figure 2 shows the security game for a fault analysis on block ciphers. We assume that we can control time, so we can choose the round in which we inject the fault. The Challenger chooses a key K. We pick some plain texts pt_i from the plain text space $\mathcal{P}$ and fault rounds R_i^*. The Challenger encrypts the plain text with the key K with and without the injection of the fault and gives us the corresponding output pair (Z_i, Z_i'). After that, we analyze the difference of the output with an arbitrary algorithm A. When we require more data for our analysis,

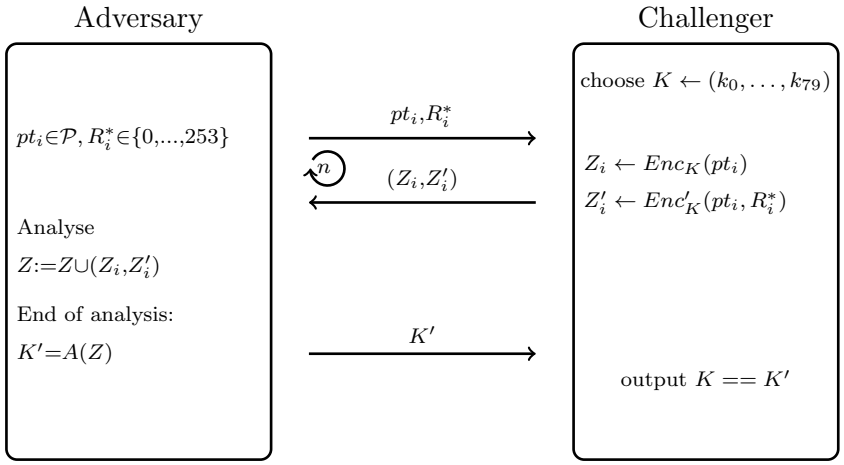

Figure 2: Security game for a fault analysis of symmetric primitives

Origin	Method	ε	τ
[ALRSS12]	Cube	115	2^{59}
[SH13]	SAT	140	–

Table 1: Previous fault attacks on Katan with the number of faults ε and the running time τ

we can request a new pair (Z_{i+1}, Z'_{i+1}) which will be encrypted with the same key. When our analysis is finished, and we have a key candidate, we can validate it by running the cipher again.

The fault model used plays a significant role in a fault attack. The existing analysis of Katan block ciphers in [ALRSS12] and [SH13] uses the following model:

We can control time. This allows us to choose the fault rounds R^*.

We flip exactly one bit at a random unknown position e in the internal state of a cipher.

We can repeat the fault injection with the same key.

In our experiments, we do not use this fault model but the described one in section 4.
The difficulty of fault attacks is to find the position of the fault. Therefore, the authors of [ALRSS12] use cube analysis to get a characteristic for fault rounds $R^* \geqslant 237$. A characteristic is a list of the output difference $\Delta = Z_i + Z'_i$ for specified fault rounds and all positions of the faults that allow us to determine the fault position e.

While guessing 59 bits and injecting 114 faults, the authors of [ALRSS12] where able to break Katan32. The authors of [SH13] were able to use a SAT-Solver in order to eliminate the guessing step of the attack. Therefore they inject 140 faults to break Katan32. In table 1 we have listed previous fault attacks on Katan32.

The number of faults ε is important because of the countermeasures in the devices. Such a device will usually not change the secret key very often. In [JT12] and [MSGR10] the authors suggest to run the cipher twice in parallel and compare the output. When a difference in the output occurs too often, the cipher changes the key and prevents fault attacks to succeed. Thus, we require ε to be small.

3 Solving systems of equations over $\mathbb{F}_2$

For our algebraic attack, we need an algebraic representation of Katan and an algorithm to solve the resulting system of equations. We want a representation that handles many Katan instances at once with the same key but with different faults injected. Therefore we fix a plain text block pt. We initialize the shift register for the key schedule with symbolic values $k_0, \ldots, k_{79} \in P := \mathbb{F}_2[k_0, \ldots, k_{79}, a_{0,0}, \ldots, a_{254,\varepsilon-1}, b_{0,0}, \ldots, b_{254,\varepsilon-1}]$ from a Boolean polynomial ring with key and intermediate variables. We define an intermediate variable every time we update one of the registers A_t or B_t with $t \in \{0, \ldots, \varepsilon - 1\}$ the number of the current faulty instance. So we get

$$\begin{aligned} a_{i,t} := A_{i,t} &= B_{i-18,t} + B_{i-7,t} + B_{i-12,t} \cdot B_{i-10,t} + B_{i-8,t} \cdot B_{i-3,t} + K_{2r+1} \\ b_{i,t} := B_{i,t} &= A_{i-12,t} + A_{i-7,t} + A_{i-8,t} \cdot A_{i-5,t} + A_{i-3,t} \cdot f_r + K_{2r}. \end{aligned}$$

We get a sparse quadratic equation system in P with 508 intermediate variables per instance and 80 variables for the key.

To handle those systems we propose a solver based on ElimLin ([CSSV12]) and a variant of XSL ([CP02]). A similar solver has been proposed in [QW14]. The authors build a solver which can handle large sparse systems with 10^6 variables, monomials and equations. Our systems are much smaller, as we have one original instance and ε faulty ones. Furthermore the systems are the same until the round in which we inject a fault. Hence we do not use all the techniques described in [QW14] and refine their techniques for rather small systems.

ElimLin is an algorithm for solving systems of equations presented in [CB07] and further investigated in [CSSV12] which eliminates linear variables through linear equations. Algorithm 1 shows the detailed ElimLin algorithm. The function `elimLin` takes a system of equations F and transforms it into a Macaulay matrix by the function `macaulay`. The Macaulay matrix over $\mathbb{F}_2$ is a matrix with equations as rows and the corresponding monomials as columns. The function `echelonize` echolonizes this matrix. After that the system is divided by `split` in a linear system of equations L and a quadratic one Q. Note that there will be no equations of higher order since we introduce intermediate variables clockwise and the update function of Katan is quadratic. Afterwards we choose a variable $\nu \in \ell$ out of each linear equation $\ell \in L$ and eliminate it in Q by substituting ℓ. We repeat this until the system of equations F is solved, there is no further linear equation in F or the system is unsolvable.

We want to stress that ElimLin preserves the overall degree of our system Q. Moreover ElimLin is able to deal with rather large but sparse systems of equations.

The original ElimLin algorithm chooses variables out of linear equations which appears least often in Q. In [QW14] the authors propose to use a monomial ordering for selection of variables. We use this approach since it is convenient in implementation and allows more control of the selection process.

ElimLin cannot conclude $a \cdot b + a = 1 \Rightarrow a = 1 \wedge b = 0$ for some variables $a, b \in \mathbb{F}_2$. For Gröbner bases algorithms and XSL this is not a problem. On the other side Gröbner bases and XL destroy the monomial structure and make the system dense in order to solve it. To overcome this weakness the authors of [QW14] propose a new variant of XSL named *SL*. They multiply every polynomial $p \in Q$ by all variables in the quadratic system ν_i. If the authors get a degree greater than 2 or a new monomial which is not already in Q, they do not work with the polynomial $\nu_i p$. Since we only have quadratic polynomials, we get the following algorithm.

1. For all polynomials p in Q, consider the variables ν_0 and ν_1 in the first monomial of p.

2. If ν_0 or ν_1 are in the rest of the monomials in p continue with the corresponding variable(s).

3. For all linear variables ν_ℓ in p test whether the monomial $\nu_\ell \nu_i$ is in the system.

Algorithm 1 ElimLin for arbitrary systems of equations F.

function elimLin(F):
$L' \leftarrow \emptyset$
flag $\leftarrow$ **true**
while flag==**true do**
  flag $\leftarrow$ **false**
  $M_F = \texttt{macaulay}(F)$
  $M_F = \texttt{echelonize}(M_F)$
  $L, Q = \texttt{split}(M_F)$
  for each $\ell \in L$ in an arbitrary order **do**
    if ℓ is trivial **then**
      if ℓ is unsolvable **then**
        return $\emptyset$
      end if
    else
      flag $\leftarrow$ **true**
      Choose variable ν in ℓ
      Substitute ν in all $q \in Q$
      Substitute ν in all $p \in L$
      L'.append(ℓ)
    end if
  end for
  $F \leftarrow L' \cup Q$
end while
return F

4. If $\nu_\ell \nu_i$ is already in the system, add $\nu_i p$ to the system of equations.

This is a strong restriction and does not add many polynomials to the system, but it greatly improves the ElimLin algorithm. The following example shows how SL improves ElimLin.

Example 1. *We consider the system of equations* $a \cdot b + a = 1$ *over* $F_2[a,b]/\langle a^2 + a, b^2 + b\rangle$.
Since there is no linear equation, ElimLin does nothing.

SL starts with the variables a *and* b *out of the first monomial* $a \cdot b$.
Though there are no more quadratic monomials, we accept a *and* b *in the second step of the description above.*
Since $a \cdot a = a$ *and* $a \cdot b$ *is in the set of all monomials, we get the new polynomials* $a \cdot b + a + a = 0 \Leftrightarrow a \cdot b = 0$ *and* $a \cdot b + a \cdot b + b = 0 \Leftrightarrow b = 0$.
This directly solves the system of equations with $b = 0, a = 1$.

Our implementation of this solver is written in C++, Python and the open source computer algebra system SAGE which is described in [S+14]. For the linear algebra part we used the method of the four russian implementation (M4RI) out of [AAB+].

In section 6 we use the ability of the ElimLin algorithm 1 to determine whether a system is solvable or not. This ability is given in lines $10 - 13$ of the algorithm and is observed for many solvers for systems of equations (see for instance [BFP09, FJ03, QW14]).

4 A more realistic fault model

In this section we describe a fault attack from an engineering point of view and translate it to a more realistic fault model.

In a physical attack we inject a fault by shooting a laser at the circuit of the internal state of a cipher (LFI). We also measure the power consumption which leads to individual clock cycles of the encryption or decryption of a message. Further we can measure how long it takes until a round is over as described in [Lib]. That allows us to measure the exact time and therefore control the round where we inject the fault.

When we shoot the laser to target the hardware, we would like to hit exactly one bit of the internal state and flip it. But that is not the case, especially for the latest low-power technology nodes Katan is designed for. Here, transistor sizes smaller than the optical limit of LFI might prevent the injection of a single bit flip. It is more likely to influence three neighboring bits at a time. But we might not flip all three of them. It is possible that our laser does not flip any bit. We will use a random bit vector in $\mathbb{F}_2^3$ to model this behavior.

Once we adjusted the laser, we do not adjust its lateral position again. We simply change the time of the injection, i.e., the round in which we inject the fault. This allows us to know the position of the error vector for the whole attack once we have found it.

Overall we get the following fault model.

\# We can control the fault round R^*.

\# We induce an error $\vec{e} \in \mathbb{F}_2^3$ at a random unknown position e.

\# For each fault the error vector $\vec{e}$ might change, but the fault position e stays the same for the attack.

For this 3-bit fault model, we need a new characteristic. Therefore we use the solver described in section 3. We model the correct instance and one faulty instance of Katan. We do not set the output of the ciphers, but set the difference $\Delta = Z + Z'$ to symbolic values. After that we

reduce the system as far as we can and extract the values Δ. The characteristic for fault rounds $R^* = 242$ are given in Appendix A. This way we have found a characteristic for $R^* \geqslant 241$.

In our attacks we first need to determine the fault position e. Even with our characteristic, we might not know the exact position of the fault. Only the error vectors $(1,0,1)$ and $(1,1,1)$ allow us to determine e. The other possible vectors do not determine the position uniquely. Therefore we have a probability of $\mathcal{P}(e|\varepsilon = 1) = \frac{2}{7}$ to find e after a fault ($\varepsilon = 1$). The zero error vector leads to $\Delta = 0$ so we ignore it. Since there is no difference, the described countermeasures do not count the fault and neither do we. After $\varepsilon = 2$ faults we have 7^2 possibilities for the pairs of error vectors. Only if a pair combined leads to an error vector $(1,0,1)$ or $(1,1,1)$ we can determine the fault position e. If one of the error vectors is $\vec{e} = (1,1,1) \vee (1,0,1)$ we directly have the fault position. There are 14 possibilities to get one of them at first fault and $5 + 5 = 10$ possibilities at the second fault. When we get the fault vectors $(1,0,0)$ and $(0,0,1)$ we have a success because combined they lead to $(1,0,1)$. Altogether we have a probability of $\mathcal{P}(e|\varepsilon = 2) = \frac{32}{49} > \frac{1}{2}$ to find the position after two faults. Since this probability is greater as $1/2$, we assume in the following that we need 2 faults on average to get the exact fault position since by an iteration we can get a probability as close to 1 as we like.

5 Statistical fault analysis of Katan32

In this section we present our first fault attack on Katan. It builds up a filter to eliminate wrong key candidates while guessing.

Consider the deciphering process of Katan. It uses 2 round key variables per round. These round keys arise from a linear feedback shift register of length 80. When we have 80 consecutive round keys we can compute the register backwards and get the key this way. Our filter does the analysis part in security game in figure 2. We get the outputs Z_i and Z'_i and guess $2R_\Delta$ round keys $X = (x_0, \ldots, x_{2R_\Delta - 1})$ with the round delta $R_\Delta = 254 - R^*$ and calculate them back to the fault round R^* using the decipher function $\texttt{dec}_X(R_\Delta, Z)$. The decipher function gives us the internal states S_X and S'_X at the fault round. Afterwards we test if $S_\Delta = S_X + S'_X$ is a valid error vector.

We note that this can be done even without knowledge of the exact fault position e. If we do not use the fault position e and assume that we have a uniform distribution of S_Δ, we have 32 possibilities to find an error of Hamming Weight (HW) 1, $31 + 30$ possible error vectors of HW 2 and 30 possible error vectors of HW 3. All in all we have $32 + 31 + 30 + 30 = 123$ error vectors that our filter will accept. There are 2^{32} possible values of S_Δ. That leads to a so-called theoretical filter quality σ, which is defined as $\sigma := \mathcal{P}(\textit{accept wrong})$. In our case, we assume a uniform distribution, so we get $\sigma = \frac{123}{2^{32}} < \frac{2^7}{2^{32}} = 2^{-25}$. So if we test 2^{80} round keys our filter accepts 2^{55} round keys. So we reduce the round key space by 2^{25}. When we do not guess 80 key bits we can construct an attack on Katan.

1. Inject a fault in round R_0^* with the round delta $R_\Delta^0 = 254 - R_0^*$

2. Guess $2R_\Delta^0$ round keys and test them with the described filter

3. Choose a new fault round and guess the corresponding round keys.

4. Apply the filter for the new round keys and the accepted ones.

5. Repeat until 80 consecutive round key bits are known.

6. Calculate the key with the linear shift register.

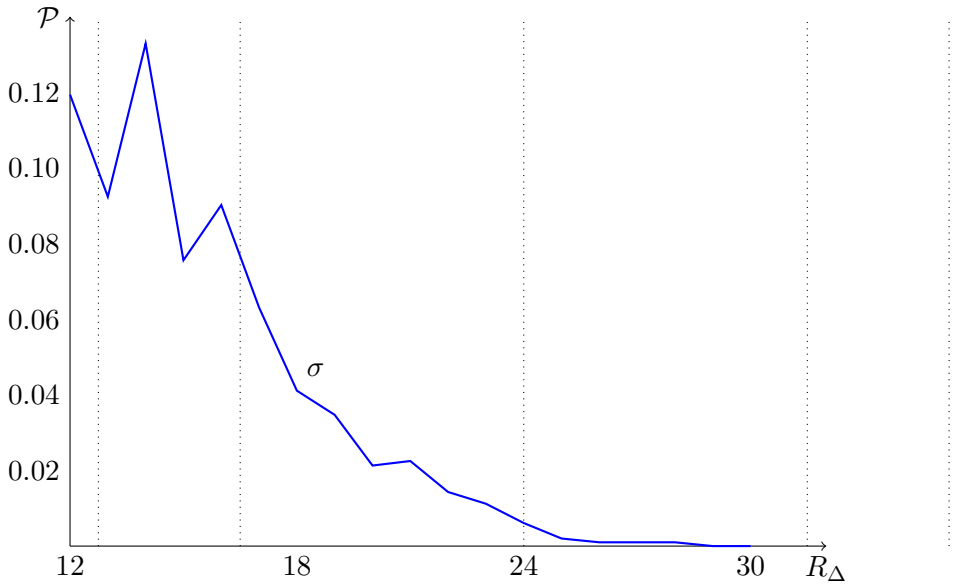

Figure 3: Filter quality depending on R_Δ; round delta R_Δ against probability $\mathcal{P}$ that a random key is accepted

With our characteristic we can further improve the filter. The exact error is not as important as the error position, since we only have a characteristic for $R^* \geqslant 241$. Therefore we can only guess 26 round key bits. After that we can guess a small number of round key bits and get the whole key. Also using the position the theoretical filter quality is $\sigma = \frac{2^3-1}{2^{32}} < 2^{-29}$.

In figure 3, the filter quality is shown without the exact position. We performed 1001 or 10001 experiments on random round keys for different round deltas R_Δ. We only run 10001 experiments when $\mathcal{P}(x \text{ accepted}) \leqslant 2/1001$.

With a low round delta, $R_\Delta < 17$, the measurement is inaccurate with a high chance of a key acceptance. So we do not have a uniform distribution for the entries of the internal state of Katan. We can see this in the characteristic in appendix A as well. The error just repositions before we have a diffusion due to the quadratic terms.

If we know the fault position figure 3 is a line at zero. We get $\sigma = 2^{-14}$ for $R_\Delta = 14$. We do not test the theoretical quality because it would require 2^{32} experiments.

Our attack proceeds as follows. We get the position of the fault by injecting two faults in round $R^* = 242$ as described in section 4. Therefore we guess 24 round key bits. As we have $R_\Delta = 12$, our filter quality is at least $\sigma \leqslant 2^{-14}$. Thus we have 2^{10} possible states at round $R = 242$. Now we inject another fault with round delta $R_\Delta = 12$ while guessing another 24 round key bits. Our filter now eliminates 2^{14} keys and we get 2^{20} possible states for round $R = 230$. For these 2^{20} states we repeat the procedure for another 4 faults and obtain 80 consecutive round key bits. We can now compute the key.

Overall we need 7 faults for the attack and $2^{24} + 2^{24} \cdot (2^{10} + 2^{20} + 2^{30} + 2^{40} + 2^{50}) \approx 2^{74}$ Katan computations.

We want to stress that we have not taken advantage of the theoretical quality, which we did not prove, as it would take 2^{32} for each R_Δ. When we do so, we get the correct round key bits at every step in the attack. We find the position with 2 faults on average and get 24 round key bits. After that we set $R_\Delta = 14$ and guess 28 round key bits per fault. Since our theoretical filter quality is 2^{-29} we can eliminate all wrong keys. Thus we just need $2^{24} + 2 \cdot 2^{28} = 2^{29.04}$ steps and $\varepsilon = 4$ faults to get the full key.

6 Algebraic fault analysis of Katan32

In this section we introduce an algebraic fault attack. We use the solver introduced in section 3 to find errors in the system of equations fast.

For the fault attack we model Katan as described in section 3. Since we know the fault round R^*, we can model the faulty instance the same way, if we have a characteristic. It gives us the error which we can add to the correct position in the internal state.

With faults in rounds $R^* > 241$, we cannot calculate a single key bit since we do not have enough information. In our experiments we needed at least a fault in round $R^* = 204$ to solve the system, but our solver can determine if we made a mistake while initializing the system of equations. Thus we are able to try all vectors with the given fault position e. We have $2^3 - 1 = 7$ possible error vectors if we do not count the zero vector. Let $\mathcal{P}(\vec{e_w}\ accepted)$ be the probability that a wrong error vector is accepted. Table 2 shows this probability for chosen fault rounds R^*. We generate the system for all previous faults with the exact error and test then a random vector $\vec{e}$. We did 1001 experiments in each case.

R_w^*	230	222	214	206	198	190
$\mathcal{P}(\vec{e_w}\ accepted)$	0,01	0,01	0,01	0,03	0,01	0,02

Table 2: Probability that we accept a random error vector $\vec{e}$ in a given round, based on 1001 trials each.

Table 2 shows that our solver is able to detect errors. We just have small systems around 800 variables, 700 equations and 3000 monomials so an error is found quickly when we insert $\varepsilon = 5$ faults.

Further we only used $R_\Delta = 8$. Consider the update function of Katan

$$\begin{aligned} A_i &= B_{i-18} + B_{i-7} + B_{i-12} \cdot B_{i-10} + B_{i-8} \cdot B_{i-3} + K_{2R+1} \\ B_i &= A_{i-12} + A_{i-7} + A_{i-8} \cdot A_{i-5} + A_{i-3} \cdot f_r + K_{2R}\,. \end{aligned}$$

We wish to inject as few faults as possible. When we choose $R_\Delta \geqslant 12$, we skip the update of register B. This leads to more independent systems in which we do not find errors. This phenomenon begins for $R_\Delta > 8$. Furthermore, at $R_\Delta = 8$, we skip two of the three quadratic monomials in the update function.

After the position is found, we can skip $R_\Delta = 16$ rounds for the first fault without loss in accuracy. For the first fault we take round $R_0^* = 246$ because we have a characteristic and round delta $R_\Delta = 8$.

Our attack is illustrated in figure 4. We inject two faults in round $R_0^* = 246$ to determine the fault position. Afterwards, we skip $R_\Delta = 16$ rounds and test all 7 possible error vectors. If we take a wrong vector, we will test all 7 wrong vectors at round $R^* = 222$ and go back recursively. The chance that we accept a wrong error vector twice in a row is overwhelmingly low ($< 1/10001$) throughout our experiments. If we choose a correct error vector we choose $R_\Delta = 8$ and inject a fault in a new round. When we do not have a solution for $R^* \geqslant 190$, we inject a fault in round 250 and proceed with the attack. In round 250 we have a characteristic so we can detect all previous errors instantly. In our experiments this second loop is only necessary in few cases.

As previously pointed out, we find a incorrect error vector directly or by testing all 7 error vectors for the new fault. The next lemma gives us a prediction of how many solver iterations on average we need to get to a solution.

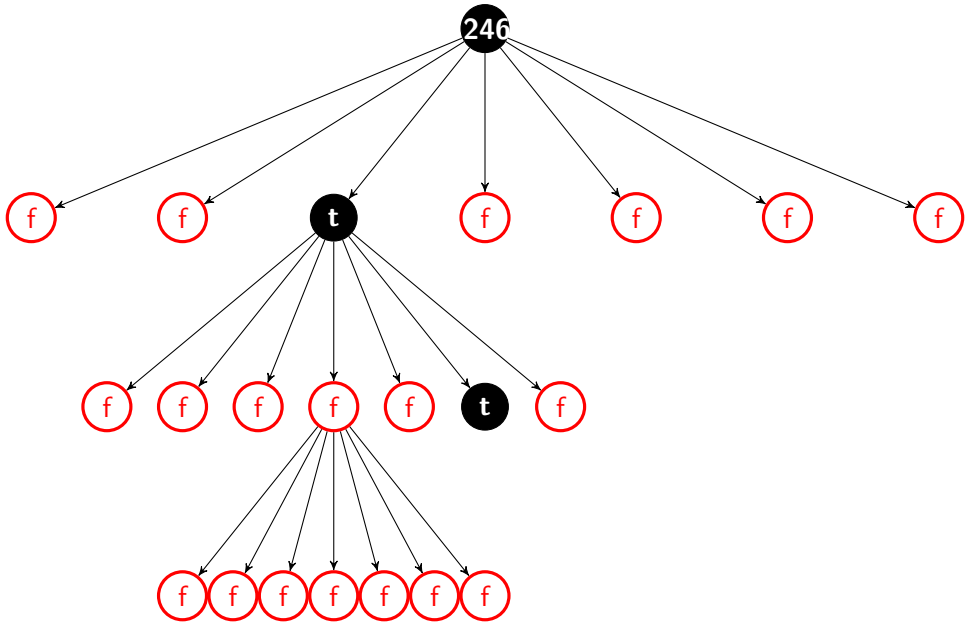

Figure 4: Structure of our algebraic fault attack on Katan; a wrong error vector is marked f and a correct one as t

Lemma 1. *On average we need*

$$2 + d + 42 \cdot \sum_{i=1}^{d} \mathcal{P}_i(\vec{e_w} \text{ accepted})$$

solver iterations for a successful attack on Katan with d faults.

In particular, we need 11.78 *solver iterations if we only inject faults until fault round* $R^* = 190$.

Proof. The tree structure of our problem is shown in figure 4. Let d be the depth of the tree for a successful attack.

We need two faults to determine the fault position. We have a probability of $\mathcal{P}_i(\vec{e_w} \text{ accepted})$ that we choose a wrong leaf of the tree. Overall there are 6 wrong and one correct leaf. To have a successful attack we need to traverse the tree down for d leaves.

If we accept a wrong leaf for the i-th fault, we recognize that with overwhelming probability ($< 1/10001$) after 7 solver iterations. All together we have

$$2 + d + 6 \cdot \sum_{i=1}^{d} 7 \cdot \mathcal{P}_d(\vec{e_w} \text{ accepted})$$

solver iterations for a successful attack.
With the values from table 2 the given number of solver iterations follows directly. □

The "jump" to $R^* = 250$ is handled as another fault with $\mathcal{P}(\vec{e_w} \text{ accepted}) = 0$. After that we choose the round delta $R_\Delta = 8$ again. The values for the probability $\mathcal{P}(\vec{e_w} \text{ accepted})$ for faults after this "jump" are similar to those shown in table 2.

For our attack, we performed 1001 experiments and the results are shown in table 3. We decided to compare the results with our statistical fault analysis.

We needed 7.19 faults on average. The variance of our attack was 1.87 faults and it took $2^{27.2}$ Katan computations to perform our attack. The time consumption was calculated as prior

Attack	ε	τ
Statistical	7	2^{74}
Statistical + theoretic quality	**4**	$2^{29.04}$
Algebraic	$7,19 \pm 1,87$	$\mathbf{2^{27,2}}$
Algebraic + Guess	$4,52 \pm 0,6$	$2^{64,8}$

Table 3: Results for the algebraic fault attack on Katan; ε is the total number of faults and τ is the time complexity needed.

stated in section 3. The average fault number of $\varepsilon = 7.19$ and variance of 1.87 faults show that we do not need the fault round 250 very often. Overall, we did the "jump" 65 times which is below 1% of the experiments done. If we choose a round delta $R_\Delta = 10$, the systems become less dense. We needed overall less faults $\varepsilon = 7.05$ but the variance increased to 2.62 faults. Again, we performed this experiment 1001 times. So we did the "jump" more often.

If we wish to use less faults, we can guess some key variables. In our experiments we found 38 suited variables for this purpose. We can lower the number of needed faults to $\varepsilon = 4.52$ with a variance of 0.6 if we know these variables. As we used our solver to detect the wrong error vectors, we cannot detect a wrong key guess that easily. On average, we need 28.1 seconds to identify a wrong guess and compute the solution in 6 seconds for a right guess of these key variables. This leads to $2^{64.8}$ Katan computations for this attack. We note, that this attack is theoretical and is no longer practical.

7 Conclusions

In this paper, we successfully attacked the Katan32 block cipher. We improved existing results in convincing fashion using new techniques and a mixture of classical fault attacks and algebraic attacks to do so. In a nutshell, we used the ability of algebraic solvers to detect inconsistent systems quickly. It is not a specialized solver but the usage as a filter that makes the attack successful. The results are summed up in table 4.

We also introduced a new fault model which arises from an electrical engineering point of view. It states that we cannot flip exactly one bit in the state of a cipher but must consider neighboring bits. Also we take into account laser positioning in real life circumstances.

Origin	Method	ε	τ	Practical?
[ALRSS12]	Cube	115	2^{59}	No
[SH13]	SAT	140	–	Yes
this paper	Statistical	7	2^{74}	No
this paper	Statistical + theoretic quality	**4**	$2^{29.04}$	Yes
this paper	Algebraic	7.19 ± 1.87	$\mathbf{2^{27.2}}$	Yes
this paper	Algebraic + Guess	4.52 ± 0.6	$2^{64.8}$	No

Table 4: Results of fault attacks on Katan; ε is the total number of faults.

We are confident that the attack can be extended to the whole Katan family and further to any symmetric primitive since we just use the specifics of Katan32 / the Katan family to adjust the attack.

We consider using our technique to improve fault attacks and algebraic fault attacks in particular as an interesting line of future research.

Acknowledgements

The author wants to thank Wolfram Koepf (University of Kassel) for fruitful discussions and guidance, Falk Schellenberg (Ruhr-University Bochum) for great help with fault models, Gregor Leander (Ruhr-University of Bochum) for discussions about the statistical attack and Christopher Wolf (Ruhr-University Bochum) for the great collaboration while implementing the solver and overall guidance. Further he gratefully acknowledges an Emmy Noether Grant of the Deutsche Forschungsgemeinschaft (DFG).

References

[AAB+] Tim Abbott, Martin Albrecht, Gregory Bard, Marco Bodrato, Michael Brickenstein, Alexander Dreyer, Jean-Guillaume Dumas, William Hart, David Harvey, Jerry James, David Kirkby, Clément Pernet, Wael Said, and Carlo Wood. M4RI(e)—Linear Algebra over F_2 (and F_2^e). http://m4ri.sagemath.org/.

[AFI+04] Gwnol Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and Gröbner Basis Algorithms. In *ASIACRYPT 2004, LECTURE*, pages 338–353. Springer-Verlag, 2004.

[Alb10] Martin Albrecht. *Algorithmic Algebraic Techniques and their Application to Block Cipher Cryptanalysis*. PhD thesis, Royal Holloway, University of London, 2010.

[ALRSS12] Shekh Faisal Abdul-Latip, Reza Reyhanitabar, Willy Susilo, and Jennifer Seberry. Fault Analysis of the KATAN Family of Block Ciphers. In *ISPEC*, pages 319–336. Springer, 2012.

[BDL97] Dan Boneh, Richard A. Demillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Advances in Cryptology – Eurocrypt '97*, volume 1233, pages 37–51. Springer-Verlag, 1997.

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *In Journal of Mathematical Cryptology*, 3:177–197, 2009.

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 513–525, London, UK, UK, 1997. Springer-Verlag.

[CB07] Nicolas T. Courtois and Gregory V. Bard. Algebraic Cryptanalysis of the Data Encryption Standard. In *Cryptography and Coding*, volume 4887 of *Lecture Notes in Computer Science*, pages 152–169. Springer Berlin Heidelberg, 2007.

[CDK] Christophe Cannière, Orr Dunkelman, and Miroslav Knežević. The KATAN/KTANTAN Family of Block Ciphers. http://www.cs.technion.ac.il/~orrd/KATAN/index.html. Accessed: 18.09.2014.

[CDK09] Christophe Cannière, Orr Dunkelman, and Miroslav Knežević. KATAN and KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 272–288, Berlin, Heidelberg, 2009. Springer-Verlag.

[CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Bart Preneel, editor, Springer, 2000. Extended Version: http://www.minrank.org/xlfull.pdf.

[CP02] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of equations. In *ASIACRYPT*, pages 267–287. Springer, 2002.

[CSSV12] Nicolas T. Courtois, Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. Elimlin Algorithm Revisited. In *Fast Software Encryption—FSE 2012*, pages 306–325. Springer, 2012.

[Die04] Claus Diem. The XL-Algorithm and a Conjecture from Commutative Algebra. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 323–337. Springer, 2004.

[Fau02a] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *International Symposium on Symbolic and Algebraic Computation — ISSAC 2002*, pages 75–83. ACM Press, July 2002.

[Fau02b] Jean-Charles Faugère. A New Efficient Algorithm for Computing Grbner Bases (F4). In *IN: ISSAC 02: PROCEEDINGS OF THE 2002 INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND ALGEBRAIC COMPUTATION*, pages 75–83. Springer, 2002.

[FJ03] Jean-Charles Faugère and Antoine Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In *In Advances in Cryptology CRYPTO 2003*, pages 44–60. Springer, 2003.

[HR08a] Michal Hojsk and Bohuslav Rudolf. Differential Fault Analysis of Trivium. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 158–172. Springer Berlin Heidelberg, 2008.

[HR08b] Michal Hojsk and Bohuslav Rudolf. Floating Fault Analysis of Trivium. In *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin Heidelberg, 2008.

[JT12] Marc Joye and Michael Tunstall. *Fault Analysis in Cryptography*. Springer Publishing Company, Incorporated, 2012.

[Lib] Joint Interpretation Library. Application of Attack Potential to Smartcards. http://www.sogisportal.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v2-9.pdf. Accessed: 30.10.2014.

[MBB11] MohamedSaiedEmam Mohamed, Stanislav Bulygin, and Johannes Buchmann. Using SAT Solving to Improve Differential Fault Analysis of Trivium. In *Information Security and Assurance*, volume 200 of *Communications in Computer and Information Science*, pages 62–71. Springer Berlin Heidelberg, 2011.

[MSGR10] Marcel Medwed, Franois-Xavier Standaert, Johann Groschdl, and Francesco Regazzoni. Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In DanielJ. Bernstein and Tanja Lange, editors, *Progress in Cryptology AFRICACRYPT 2010*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer Berlin Heidelberg, 2010.

[QW14] Frank Quedenfeld and Christopher Wolf. Advanced Algebraic Attack on Trivium. Cryptology ePrint Archive, Report 2014/893, 2014. http://eprint.iacr.org/.

[S+14] W. A. Stein et al. *Sage Mathematics Software (Version 6.1)*. The Sage Development Team, 2014. http://www.sagemath.org.

[SH13] Ling Song and Lei Hu. Improved Algebraic and Differential Fault Attacks on the KATAN Block Cipher. In *ISPEC*, pages 372–386. Springer, 2013.

[YC04a] Bo-Yin Yang and Jiun-Ming Chen. All in the XL family: Theory and practice. In *ICISC 2004*, pages 67–86. Springer, 2004.

[YC04b] Bo-Yin Yang and Jiun-Ming Chen. Theoretical analysis of XL over small fields. In *ACISP 2004*, volume 3108 of *LNCS*, pages 277–288. Springer, 2004.

A Characteristic of Katan

The tables in this section give the characteristic we have calculated with the model from section 3. To calculate the characteristic, we have used the solver in section 3 with variables for the difference of the output which we do not set to actual values. Then we reduced the equation system as far as possible and extract the characteristic. We show the characteristic for the fault round $R^* = 242$.

In a characteristic a row is the output delta $\Delta = (z_0 + z'_0, \ldots, z_{31} + z'_{31})$ against the fault position e of the fault. - means a value which is not fixed or which depends on the output of the cipher.

$e\backslash z_i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	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	1
1	1	0	0	0	-	1	0	-	0	0	0	0	0	0	0	0	0	0	0	-	-	0	-	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	-	1	0	-	0	0	0	0	0	0	0	0	0	0	-	-	-	0	-	0	0	0	0	0	0	0	0
3	0	-	1	0	0	0	-	1	0	-	0	0	0	0	0	0	0	0	0	-	-	-	-	0	-	0	0	0	0	0	0	0
4	-	0	0	1	0	0	0	-	1	0	-	0	0	0	0	0	0	0	0	1	-	-	-	-	0	-	0	0	0	0	0	0
5	0	-	0	0	1	0	0	0	-	1	0	-	0	0	0	0	0	0	0	-	1	-	-	-	-	0	-	0	0	0	0	0
6	-	-	0	0	0	1	0	0	0	-	1	0	0	0	0	0	0	0	0	-	-	1	0	0	-	-	0	0	0	0	0	0
7	-	-	0	0	0	0	1	0	0	0	-	1	0	0	0	0	0	0	0	-	-	-	1	0	0	-	-	0	0	0	0	0
8	0	-	0	0	0	0	0	1	0	0	0	-	0	0	0	0	0	0	0	-	0	-	-	0	0	0	-	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-	0	0	0	0	0	0	0	0
10	0	-	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-	1	0	0	0	-	0	0	0	0	0	0	0
11	-	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-	1	0	0	0	-	0	0	0	0	0	0
12	0	-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-	0	-	1	0	0	0	-	0	0	0	0	0
13	-	0	-	0	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0	-	-	0	-	1	0	0	0	-	0	0	0	0
14	-	-	0	-	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	-	0	-	0	-	1	0	0	0	-	0	0	0
15	-	-	-	0	-	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	-	0	-	1	0	0	0	-	0	0
16	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	-	0	-	1	0	0	0	-	0
17	1	-	-	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-	0	0	-	0	-	0	-	1	0	0	0	0
18	-	1	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-	0	0	0	-	0	-	0	-	1	0	0	0
19	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0	0	0	-	0	-	0	-	1	0	0
20	-	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	-	0	-	0	-	1	0
21	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0	-	0	-	0	0
22	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	0	0	0	0	0	-	0	-	0	0
23	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	0	0	0	0	0	-	0	-	0
24	0	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0	-	0	0
25	0	1	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0	-	0
26	-	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	0	0	0	0
27	0	-	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	0	0	0
28	1	0	-	0	1	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	0
29	-	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	0	1	0	0	0
30	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
31	0	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

Table 5: Characteristic for one error after fault round $R^* = 242$ of Katan. The fault position e means a flip at position e.

$e\backslash z_i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	0	0	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0	1
1	1	1	0	0	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	0	0	0	0	0	0	0	0
2	0	-	1	0	0	-	-	1	-	-	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	0	0	0	0	0	0	0
3	-	-	1	1	0	0	-	-	1	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	0	0	0	0	0	0
4	-	-	0	1	1	0	0	-	-	1	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0
5	-	-	0	0	1	1	0	0	-	-	1	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0
6	-	-	0	0	0	1	1	0	0	-	-	1	0	0	0	0	0	0	0	-	-	-	1	0	-	-	-	0	0	0	0	0
7	-	-	0	0	0	0	1	1	0	0	-	-	0	0	0	0	0	0	0	-	-	-	-	0	0	-	-	0	0	0	0	0
8	0	-	0	0	0	0	0	1	1	0	0	-	0	0	0	0	0	0	0	-	0	-	-	-	0	0	-	0	0	0	0	0
9	0	-	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	-	1	0	0	-	-	0	0	0	0	0	0	0
10	-	-	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	-	-	1	0	0	-	-	0	0	0	0	0	0
11	-	-	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	-	-	-	1	0	0	-	-	0	0	0	0	0
12	-	-	-	0	-	0	0	0	0	0	0	1	1	0	0	0	0	0	0	-	-	-	-	1	0	0	-	-	0	0	0	0
13	-	-	-	-	-	0	0	0	0	0	0	0	1	1	0	0	0	0	0	-	-	-	-	-	1	0	0	-	-	0	0	0
14	-	-	-	-	-	0	0	0	0	0	0	0	0	1	1	0	0	0	0	-	-	-	-	-	-	1	0	0	-	-	0	0
15	-	-	-	-	-	-	0	0	0	0	0	0	0	0	1	1	0	0	0	-	-	-	-	-	-	-	1	0	0	-	-	0
16	-	-	-	-	1	-	0	0	0	0	0	0	0	0	0	1	1	0	0	-	-	-	-	-	-	-	-	1	0	0	-	0
17	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	1	1	0	-	0	0	-	-	-	-	-	-	1	0	0	0
18	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	1	1	-	0	0	0	-	-	-	-	-	-	1	0	0
19	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	1	-	-	0	0	0	-	-	-	-	-	-	1	0
20	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	-	-	-	-	-	-	0
21	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	1	0	0	0	0	-	-	-	-	-	0
22	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	1	1	0	0	0	-	-	-	-	-	0
23	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	1	1	0	0	0	0	-	-	-	0
24	0	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	1	1	0	0	0	0	-	-	0
25	-	1	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1	1	0	0	0	0	-	0
26	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
27	1	-	-	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	1	1	0	0	0	0
28	-	1	-	-	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	1	1	0	0	0
29	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	1	1	0	0
30	0	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	1	1	0

Table 6: Characteristic for a double error after fault round $R^* = 242$ of Katan. The fault position e means a flip at position e and $e + 1$.

$e\backslash z_i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	1	0	-	1	-	-	0	-	0	0	0	0	0	0	0	0	0	0	-	-	-	0	-	0	0	0	0	0	0	0	1
1	1	-	1	0	-	1	-	-	0	-	0	0	0	0	0	0	0	0	0	-	-	-	-	0	-	0	0	0	0	0	0	0
2	-	1	0	1	0	-	1	-	-	0	-	0	0	0	0	0	0	0	0	-	-	-	-	-	0	-	0	0	0	0	0	0
3	0	-	1	0	1	0	-	1	-	-	0	-	0	0	0	0	0	0	0	-	-	-	-	-	-	0	-	0	0	0	0	0
4	-	-	0	1	0	1	0	-	1	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	0	0	0	0	0	0
5	-	-	0	0	1	0	1	0	-	1	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0
6	-	-	0	0	0	1	0	1	0	-	1	-	0	0	0	0	0	0	0	-	-	-	-	0	-	-	-	0	0	0	0	0
7	-	-	0	0	0	0	1	0	1	0	-	1	0	0	0	0	0	0	0	-	-	-	1	-	0	-	-	0	0	0	0	0
8	0	-	0	0	0	0	0	1	0	1	0	-	0	0	0	0	0	0	0	-	1	-	-	0	-	0	-	0	0	0	0	0
9	-	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	-	1	0	-	0	-	0	0	0	0	0	0
10	0	-	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	-	1	-	1	0	-	0	-	0	0	0	0	0
11	-	0	-	0	-	0	0	0	0	0	1	0	1	0	0	0	0	0	0	-	-	1	-	1	0	-	0	-	0	0	0	0
12	-	-	0	-	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	-	0	-	1	-	1	0	-	0	-	0	0	0
13	-	-	-	0	-	0	0	0	0	0	0	0	1	0	1	0	0	0	0	-	-	0	-	1	-	1	0	-	0	-	0	0
14	-	-	-	-	0	-	0	0	0	0	0	0	0	1	0	1	0	0	0	-	-	-	0	-	1	-	1	0	-	0	-	0
15	-	-	-	0	-	0	0	0	0	0	0	0	0	0	1	0	1	0	0	-	-	0	-	0	-	1	-	1	0	-	0	0
16	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	1	0	1	0	-	-	-	0	-	0	-	1	-	1	0	-	0
17	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	1	0	1	-	0	0	-	0	-	0	-	1	-	1	0	0
18	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	1	0	-	-	0	0	-	0	-	0	-	1	-	1	0
19	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	1	-	-	0	0	0	-	0	-	0	-	1	-	0
20	-	-	-	1	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	1	0	0	0	-	0	-	0	-	1	0
21	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	0	0	0	-	0	-	0	-	0
22	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	0	1	0	0	0	-	0	-	0	0
23	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	0	1	0	0	0	-	0	-	0
24	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1	0	1	0	0	0	-	0	0
25	0	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1	0	1	0	0	0	-	0
26	-	0	-	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	1	0	1	0	0	0	0
27	-	-	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	1	0	0	0
28	1	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	1	0	1	0	0
29	-	1	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	1	0	1	0

Table 7: Characteristic for a double error after fault round $R^* = 242$ of Katan. The fault position e means a flip at position e and $e + 2$.

$e\backslash z_i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	1	0	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	0	0	0	0	0	0	0	1
1	1	-	1	0	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	0	0	0	0	0	0	0
2	-	-	1	1	0	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	0	0	0	0	0	0
3	-	-	1	1	1	0	-	-	-	-	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0
4	-	-	0	1	1	1	0	-	-	-	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0
5	-	-	0	0	1	1	1	0	-	-	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0
6	-	-	0	0	0	1	1	1	0	-	-	-	0	0	0	0	0	0	0	-	-	-	-	0	-	-	-	0	0	0	0	0
7	-	-	0	0	0	0	1	1	1	0	-	-	0	0	0	0	0	0	0	-	-	-	-	-	0	-	-	0	0	0	0	0
8	0	-	0	0	0	0	0	1	1	1	0	-	0	0	0	0	0	0	0	-	1	-	-	-	-	0	-	0	0	0	0	0
9	-	-	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	-	-	1	0	-	-	-	0	0	0	0	0	0
10	-	-	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	-	-	-	1	0	-	-	-	0	0	0	0	0
11	-	-	-	0	-	0	0	0	0	0	1	1	1	0	0	0	0	0	0	-	-	-	-	1	0	-	-	-	0	0	0	0
12	-	-	-	-	-	0	0	0	0	0	0	1	1	1	0	0	0	0	0	-	-	-	-	-	1	0	-	-	-	0	0	0
13	-	-	-	-	-	0	0	0	0	0	0	0	1	1	1	0	0	0	0	-	-	-	-	-	-	1	0	-	-	-	0	0
14	-	-	-	-	-	-	0	0	0	0	0	0	0	1	1	1	0	0	0	-	-	-	-	-	-	-	1	0	-	-	-	0
15	-	-	-	-	-	-	0	0	0	0	0	0	0	0	1	1	1	0	0	-	-	-	-	-	-	-	-	1	0	-	-	0
16	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	1	1	1	0	-	-	-	-	-	-	-	-	-	1	0	-	0
17	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	1	1	1	-	0	0	-	-	-	-	-	-	-	1	0	0
18	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	1	1	-	-	0	0	-	-	-	-	-	-	-	1	0
19	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	1	-	-	0	0	0	-	-	-	-	-	-	-	0
20	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	1	0	0	0	-	-	-	-	-	-	0
21	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	1	1	0	0	0	-	-	-	-	-	0
22	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	1	1	1	0	0	0	-	-	-	-	0
23	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	1	1	1	0	0	0	-	-	-	0
24	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	1	1	1	0	0	0	-	-	0
25	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1	1	1	0	0	0	-	0
26	-	-	-	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	1	1	1	0	0	0	0
27	-	-	-	-	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	1	1	1	0	0	0
28	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	1	1	1	0	0
29	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	1	1	1	0

Table 8: Characteristic for a triple error after fault round $R^* = 242$ of Katan. The fault position e means a flip at position $e, e+1$ and $e+2$.