An Algebraic Fault Attack on the LED Block Cipher

P. Jovanovic, M. Kreuzer and I. Polian

Fakultät für Informatik und Mathematik
Universität Passau
D-94030 Passau, Germany

Abstract. In this paper we propose an attack on block ciphers where we combine techniques derived from algebraic and fault based cryptanalysis. The recently introduced block cipher `LED` serves us as a target for our attack. We show how to construct an algebraic representation of the encryption map and how to cast the side channel information gained from a fault injection into polynomial form. The resulting polynomial system is converted into a logical formula in conjunctive normal form and handed over to a SAT solver for reconstruction of the secret key. Following this approach we were able to mount a new, successful attack on the version of `LED` that uses a 64-bit secret key, requiring only a single fault injection.

Key words: Cryptanalysis, algebraic attacks, differential fault analysis, fault based attack, LED block cipher, SAT solver

1 Introduction

Immunity to conventional cryptanalysis has been formally proven for a number of ciphers. Newly developed ciphers are expected to be resistant against known cryptanalytic methods. For this reason, *fault-based cryptanalysis* [5] is receiving increasing attention [9, 10, 13, 16, 19]. In fault-based cryptanalysis, the attacker targets the hardware implementation of a cryptographic algorithm rather than the algorithm itself. The attacker performs a *fault injection* into the electronic circuit and manipulates the logical values being processed by the circuit. A variety of fault-injection techniques has been discussed [2]. For instance, the attacker may reduce the power-supply voltage of the circuit, causing the logic gates within the circuit to switch slower; as a consequence, wrong values will be calculated. A different technique is irradiating a desired location in the circuit (a logic gate performing some calculation or a register holding an intermediate value) using a laser. The laser pulse will induce parasitic currents and ultimately flip the logical value of the targeted location from logic-0 to logic-1 or vice versa.

Typically, the attacker will run the cryptographic algorithms multiple times, with and without fault injection, and will perform differential cryptanalysis on the outcomes (see [3]). Obviously, fault-based attacks are easier if the attacker

can accurately control which logic structure is manipulated and what new value it assumes. In reality, the effectiveness of a fault-based attack may suffer if the attacker has only limited control over the location and/or the exact time (calculation step) of the fault injection. For example, the laser may have a precision that is sufficient to target a register but not sufficient to target individual memory cells within the register. In this case, the register's value will be modified, but to an unknown value. Therefore, a fault-based attack is always defined with respect to an assumption on the attacker's technical capabilities.

We recently introduced a fault-based attack [12] on the new `LED` block cipher [7]. The `LED` encryption scheme is conceptually similar to `AES` [17] but belongs to the family of lightweight block ciphers [4, 8], which are developed for usage in low-cost, power-constrained systems, and are typically employed in mobile, embedded and ubiquitous contexts. Those ciphers carefully balance cryptographic strength against resource requirements, most importantly power consumption. We were able to break `LED` using one fault injection under weak assumptions on the resolution of the equipment. Our attack yielded a reduced set of key candidates which was feasible for brute force enumeration.

Recently, a new idea originated in [18], namely to enhance algebraic attacks by information obtained through side-channel cryptanalysis. This idea was further developed in [6] and used in [15] to attack the stream cipher `Trivium`. In this paper, we exploit this idea by combining the previously mentioned fault-based attack on the `LED` block cipher with a more traditional algebraic attack. The paper is organized as follows.

In the next section we describe the 64-bit and 128-bit versions of the `LED` cipher and provide a complete algebraic description of the encryption map. After that we recall in Section 3 the fault attack from [12] and discuss the transformation of the fault equations to fault polynomials. The description of the actual attack and experimental results showing its practical feasibility are the subject of Section 4. Finally, Section 5 containing our conclusions and open questions finishes the paper.

Unless specifically stated otherwise, we will use the terminology and notation introduced in [14].

2 Algebraic Representation of the `LED` Block Cipher

In this section we recall the design of the `LED` cipher, as specified in [7]. In addition, we provide an algebraic representation of the encryption map using multivariate polynomials over $\mathbb{F}_2$. To accomplish this, we show for each step in the encryption algorithm of `LED` how one can represent it via polynomials.

Let us start with a brief overview of the main features of `LED`. Its structural layout shows several parallels to the block ciphers `AES` [17] and `PRESENT` [4]. The cipher `LED` has 64-bit blocks and one or two 64-bit keys. We denote these two versions by `LED-64` and `LED-128`, respectively. Other key lengths, e.g. the popular choice of 80 bits, are padded to 128 bits by appending zeros. Depending

on the key size, the encryption algorithm performs 32 rounds for `LED-64` and 48 rounds for `LED-128`.

Every 64-bit state s of the cipher is divided into 16 nibbles (4-bit tuples) $s = s_1 \parallel s_2 \parallel \cdots \parallel s_{16}$ and these are arranged in a matrix of size 4×4 of the form

$$s = \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ s_5 & s_6 & s_7 & s_8 \\ s_9 & s_{10} & s_{11} & s_{12} \\ s_{13} & s_{14} & s_{15} & s_{16} \end{pmatrix}$$

Each 4-bit sized entry $s_i = a_{4i-3} \parallel a_{4i-2} \parallel a_{4i-1} \parallel a_{4i}$ is identified with an element of the finite field $\mathbb{F}_{16} \cong \mathbb{F}_2[x]/\langle x^4 + x + 1\rangle$ as follows. Notice that the residue classes of $\{1, x, x^2, x^3\}$ form an $\mathbb{F}_2$-vector space basis of this field. Then s_i corresponds to the field element $a_{4i-3}x^3 + a_{4i-2}x^2 + a_{4i-1}x + a_{4i}$ where $a_j \in \mathbb{F}_2$. For instance, the 4-bit entry `1011` is identified with $x^3 + x + 1$. For convenience, we also write 4-bit strings in their hexadecimal form, e.g. `1011` = `B`.

To construct the polynomial representation of `LED`, we use indeterminates $p_1, \ldots, p_{64}$ representing the bits of a plaintext unit, indeterminates $k_1, \ldots, k_{64}$ (or $k_1, \ldots, k_{128}$ for `LED-128`) representing the key bits, indeterminates $c_1, \ldots, c_{64}$ representing the bits of a ciphertext unit, and indeterminates $x_i^{(r)}, y_i^{(r)}, z_i^{(r)}$ representing various states of the cipher during encryption round r (as defined below). In particular, if we combine the input bits to field elements $m_i = p_{4i-3}x^3 + p_{4i-2}x^2 + p_{4i-1}x + p_{4i}$, the input state of the encryption map is represented by the matrix

$$M = \begin{pmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \\ m_{13} & m_{14} & m_{15} & m_{16} \end{pmatrix}$$

of size 4×4 over the field $\mathbb{F}_{16}$. Similarly, we can represent the key by a matrix K (or two matrices $K, \widetilde{K}$) of size 4×4 over $\mathbb{F}_{16}$.

The general layout of the encryption algorithm is illustrated by the following Figure 1. It exhibits a special feature of this cipher – there is no key schedule. Key additions are effected by the function `AddRoundKey (AK)`. It performs an

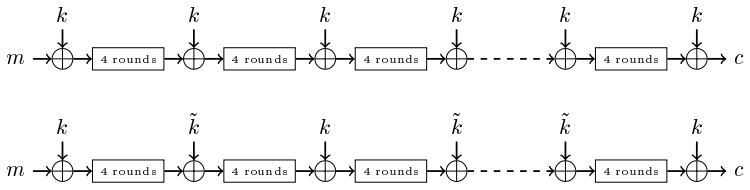

Fig. 1. `LED` key usage: 64-bit key (top) and 128-bit key (bottom).

addition of the state matrix and the matrix representing the key using bitwise

XOR. It is applied for input- and output-whitening as well as after every fourth round.

Next, Figure 2 provides an overview of the structure of one round of the LED encryption algorithm. All matrices are defined over the field $\mathbb{F}_{16}$.

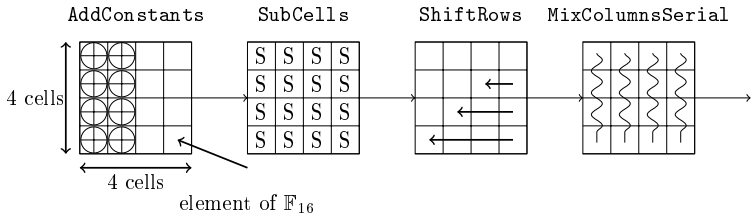

Fig. 2. An overview of a single round of LED

In the following, we construct the polynomial representation of each step. It will be contained in $\mathbb{F}_2[p_i, k_i, x_i^{(r)}, y_i^{(r)} z_i^{(r)}, c_i \mid i = 1, \ldots, 64; r = 1, \ldots, 32]$, a polynomial ring having no less than 6336 indeterminates.

AddConstants (AC). For every round, a round constant consisting of a tuple of six bits $(b_5, b_4, b_3, b_2, b_1, b_0)$ is defined as follows. Before the first round, we start with the zero tuple. In consecutive rounds, we start with the previous round constant. Then we shift the six bits one position to the left. The new value of b_0 is computed as $b_5 + b_4 + 1$. This results in the round constants whose hexadecimal values are given in Table 1.

Rounds	Constants
1-24	01,03,07,0F,1F,3E,3D,3B,37,2F,1E,3C,39,33,27,0E,1D,3A,35,2B,16,2C,18,30
25-48	21,02,05,0B,17,2E,1C,38,31,23,06,0D,1B,36,2D,1A,34,29,12,24,08,11,22,04

Table 1. The LED round constants.

Next, the round constant is divided into $\mathtt{x} = b_5 \parallel b_4 \parallel b_3$ and $\mathtt{y} = b_2 \parallel b_1 \parallel b_0$ where we interpret x and y as elements of $\mathbb{F}_{16}$. Then we form the matrix

$$\begin{pmatrix} 0 & \mathtt{x} & 0 & 0 \\ 1 & \mathtt{y} & 0 & 0 \\ 2 & \mathtt{x} & 0 & 0 \\ 3 & \mathtt{y} & 0 & 0 \end{pmatrix}$$

and add it to the state matrix. (In the current setting, matrix addition is nothing but bitwise XOR.)

To represent this operation by polynomials, we distinguish two cases: round number $r = 1$ and round numbers $r > 1$. In the first case we model the input whitening and the first application of AC in one step. Since the first round constants vector is $(b_5, b_4, b_3, b_2, b_1, b_0) = (0, 0, 0, 0, 0, 1)$, we get

$$\begin{aligned} x_i^{(1)} &= p_i + k_i + 1 \quad && \text{for } i \in \{20, 24, 35, 51, 52, 56\}, \\ x_i^{(1)} &= p_i + k_i && \text{otherwise.} \end{aligned}$$

Here the indeterminates $x_i^{(1)}$ describe the state after the first application of AC. Similarly, let $x_i^{(r)}$ describe the state after the r-th application of AC, for $r = 2, \ldots, 32$, and let $z_i^{(r)}$ denote the state of the cipher after the application of MSC in round r.

For the case $r > 1$, let $(b_5^{(r)}, b_4^{(r)}, b_3^{(r)}, b_2^{(r)}, b_1^{(r)}, b_0^{(r)})$ be the r-th round constants vector, as definied in Table 1. Then we get

$$\begin{aligned} x_i^{(r)} &= z_i^{(r-1)} + 1 && \text{for } i \in \{20, 35, 51, 52\} \\ x_i^{(r)} &= z_i^{(r-1)} + b_5^{(r)} && \text{for } i \in \{6, 38\} \\ x_i^{(r)} &= z_i^{(r-1)} + b_4^{(r)} && \text{for } i \in \{7, 39\} \\ x_i^{(r)} &= z_i^{(r-1)} + b_3^{(r)} && \text{for } i \in \{8, 40\} \\ x_i^{(r)} &= z_i^{(r-1)} + b_2^{(r)} && \text{for } i \in \{22, 54\} \\ x_i^{(r)} &= z_i^{(r-1)} + b_1^{(r)} && \text{for } i \in \{23, 55\} \\ x_i^{(r)} &= z_i^{(r-1)} + b_0^{(r)} && \text{for } i \in \{24, 56\} \\ x_i^{(r)} &= z_i^{(r-1)} && \text{otherwise} \end{aligned}$$

in rounds whose round number r is not divisible by four, and the same equations plus a keybit addition every fourth round.

SubCells (SC). During this step, every entry x of the state matrix is replaced by the element $S(x)$ from the SBox given in Table 2. Let $x_1 \parallel x_2 \parallel x_3 \parallel x_4$ be the

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2. The LED SBox.

4-bit sized input and $y_1 \parallel y_2 \parallel y_3 \parallel y_4$ the 4-bit sized output of S. Then an easy interpolation computation shows that the SBox is represented by polynomials

as follows.

$$
\begin{aligned}
y_1 &= x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + x_2x_3 + x_1 + x_3 + x_4 + 1\\
y_2 &= x_1x_2x_4 + x_1x_3x_4 + x_1x_3 + x_1x_4 + x_3x_4 + x_1 + x_2 + 1\\
y_3 &= x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + x_1x_2 + x_1x_3 + x_1 + x_3\\
y_4 &= x_2x_3 + x_1 + x_2 + x_4
\end{aligned}
$$

In our polynomial representation of the `LED` encryption algorithm, this step will be combined with the next.

ShiftRows (SR). For $i = 1, 2, 3, 4$, the i-th row of the state matrix is shifted cyclically to the left by $i - 1$ positions. Equivalently, this permutation of the 64-bit state can be described by

$$
\begin{aligned}
\sigma = {} & (17\ 29\ 25\ 21)(18\ 30\ 26\ 22)(19\ 31\ 27\ 23)(20\ 32\ 28\ 24)\\
& (33\ 41)(34\ 42)(35\ 43)(36\ 44)(37\ 45)(38\ 46)(39\ 47)(40\ 48)\\
& (49\ 53\ 57\ 61)(50\ 54\ 58\ 62)(51\ 55\ 59\ 63)(52\ 56\ 60\ 64)
\end{aligned}
$$

Note that the first row of the state matrix stays fixed under the `ShiftRows` permutation. Thus the indices $1, \ldots, 16$ do not appear in the above representation of σ.

Now we model the combined effect of `SubCells` and `ShiftRows`. Let $i_1 = 4i - 3$, $i_2 = 4i - 2$, $i_3 = 4i - 1$ and $i_4 = 4i$ for $i = 1, \ldots, 16$. Then, in round r, we get the following four equations.

$$
\begin{aligned}
y^{(r)}_{\sigma(i_1)} &= x^{(r)}_{i_1}x^{(r)}_{i_2}x^{(r)}_{i_4} + x^{(r)}_{i_1}x^{(r)}_{i_3}x^{(r)}_{i_4} + x^{(r)}_{i_2}x^{(r)}_{i_3}x^{(r)}_{i_4} +\\
&\qquad x^{(r)}_{i_2}x^{(r)}_{i_3} + x^{(r)}_{i_1} + x^{(r)}_{i_3} + x^{(r)}_{i_4} + 1\\
y^{(r)}_{\sigma(i_2)} &= x^{(r)}_{i_1}x^{(r)}_{i_2}x^{(r)}_{i_4} + x^{(r)}_{i_1}x^{(r)}_{i_3}x^{(r)}_{i_4} + x^{(r)}_{i_1}x^{(r)}_{i_3} +\\
&\qquad x^{(r)}_{i_1}x^{(r)}_{i_4} + x^{(r)}_{i_3}x^{(r)}_{i_4} + x^{(r)}_{i_1} + x^{(r)}_{i_2} + 1\\
y^{(r)}_{\sigma(i_3)} &= x^{(r)}_{i_1}x^{(r)}_{i_2}x^{(r)}_{i_4} + x^{(r)}_{i_1}x^{(r)}_{i_3}x^{(r)}_{i_4} + x^{(r)}_{i_2}x^{(r)}_{i_3}x^{(r)}_{i_4} +\\
&\qquad x^{(r)}_{i_1}x^{(r)}_{i_2} + x^{(r)}_{i_1}x^{(r)}_{i_3} + x^{(r)}_{i_1} + x^{(r)}_{i_3}\\
y^{(r)}_{\sigma(i_4)} &= x^{(r)}_{i_2}x^{(r)}_{i_3} + x^{(r)}_{i_1} + x^{(r)}_{i_2} + x^{(r)}_{i_4}
\end{aligned}
$$

MixColumnsSerial (MCS). Every column v of the state matrix is replaced by the product $M \cdot v$, where M is the matrix

$$
M = \begin{pmatrix}
\mathtt{4} & \mathtt{1} & \mathtt{2} & \mathtt{2}\\
\mathtt{8} & \mathtt{6} & \mathtt{5} & \mathtt{6}\\
\mathtt{B} & \mathtt{E} & \mathtt{A} & \mathtt{9}\\
\mathtt{2} & \mathtt{2} & \mathtt{F} & \mathtt{B}
\end{pmatrix}
$$

Let $y_1^{(r)} \parallel \cdots \parallel y_{64}^{(r)}$ be the state of the cipher after `ShiftRows` has been executed in round r, and let $z_1^{(r)} \parallel \cdots \parallel z_{64}^{(r)}$ be its state after `MCS`. The entries of the state matrix are the field elements $y_{4i-3}^{(r)}x^3 + y_{4i-2}^{(r)}x^2 + y_{4i-1}^{(r)}x + y_{4i}^{(r)}$ of $\mathbb{F}_{16}$. Plugging these into the above matrix multiplication yields, for instance, the following first entry of the resulting state matrix.

$$\begin{aligned} z_1^{(r)}x^3 + z_2^{(r)}x^2 + z_3^{(r)}x + z_4^{(r)} = \; & x^2 \cdot (y_1^{(r)}x^3 + y_2^{(r)}x^2 + y_3^{(r)}x + y_4^{(r)}) \\ + \; & 1 \cdot (y_{17}^{(r)}x^3 + y_{18}^{(r)}x^2 + y_{19}^{(r)}x + y_{20}^{(r)}) \\ + \; & x \cdot (y_{33}^{(r)}x^3 + y_{34}^{(r)}x^2 + y_{35}^{(r)}x + y_{36}^{(r)}) \\ + \; & x \cdot (y_{49}^{(r)}x^3 + y_{50}^{(r)}x^2 + y_{51}^{(r)}x + y_{52}^{(r)}). \end{aligned}$$

After expanding, simplifying, and comparing the coefficients of $1, x, x^2, x^3$, we finally get 64 equations

$$\begin{aligned}
z_{j1}^{(r)} &= y_{j3}^{(r)} + y_{j5}^{(r)} + y_{j10}^{(r)} + y_{j14}^{(r)} \\
z_{j2}^{(r)} &= y_{j1}^{(r)} + y_{j4}^{(r)} + y_{j6}^{(r)} + y_{j11}^{(r)} + y_{j15}^{(r)} \\
z_{j3}^{(r)} &= y_{j1}^{(r)} + y_{j2}^{(r)} + y_{j7}^{(r)} + y_{j9}^{(r)} + y_{j12}^{(r)} + y_{j13}^{(r)} + y_{j16}^{(r)} \\
z_{j4}^{(r)} &= y_{j2}^{(r)} + y_{j8}^{(r)} + y_{j9}^{(r)} + y_{j13}^{(r)} \\
z_{j5}^{(r)} &= y_{j1}^{(r)} + y_{j4}^{(r)} + y_{j6}^{(r)} + y_{j7}^{(r)} + y_{j9}^{(r)} + y_{j11}^{(r)} + y_{j14}^{(r)} + y_{j15}^{(r)} \\
z_{j6}^{(r)} &= y_{j1}^{(r)} + y_{j2}^{(r)} + y_{j5}^{(r)} + y_{j7}^{(r)} + y_{j8}^{(r)} + y_{j9}^{(r)} + y_{j10}^{(r)} + y_{j12}^{(r)} + y_{j13}^{(r)} + y_{j15}^{(r)} + y_{j16}^{(r)} \\
z_{j7}^{(r)} &= y_{j2}^{(r)} + y_{j3}^{(r)} + y_{j6}^{(r)} + y_{j8}^{(r)} + y_{j9}^{(r)} + y_{j10}^{(r)} + y_{j11}^{(r)} + y_{j14}^{(r)} + y_{j16}^{(r)} \\
z_{j8}^{(r)} &= y_{j3}^{(r)} + y_{j5}^{(r)} + y_{j6}^{(r)} + y_{j10}^{(r)} + y_{j12}^{(r)} + y_{j13}^{(r)} + y_{j14}^{(r)} \\
z_{j9}^{(r)} &= y_{j2}^{(r)} + y_{j4}^{(r)} + y_{j5}^{(r)} + y_{j6}^{(r)} + y_{j7}^{(r)} + y_{j8}^{(r)} + y_{j9}^{(r)} + y_{j10}^{(r)} + y_{j12}^{(r)} + y_{j16}^{(r)} \\
z_{j10}^{(r)} &= y_{j1}^{(r)} + y_{j3}^{(r)} + y_{j6}^{(r)} + y_{j7}^{(r)} + y_{j8}^{(r)} + y_{j9}^{(r)} + y_{j10}^{(r)} + y_{j11}^{(r)} + y_{j13}^{(r)} \\
z_{j11}^{(r)} &= y_{j1}^{(r)} + y_{j2}^{(r)} + y_{j4}^{(r)} + y_{j7}^{(r)} + y_{j8}^{(r)} + y_{j9}^{(r)} + y_{j10}^{(r)} + y_{j11}^{(r)} + y_{j12}^{(r)} + y_{j14}^{(r)} \\
z_{j12}^{(r)} &= y_{j1}^{(r)} + y_{j3}^{(r)} + y_{j4}^{(r)} + y_{j5}^{(r)} + y_{j6}^{(r)} + y_{j7}^{(r)} + y_{j9}^{(r)} + y_{j11}^{(r)} + y_{j15}^{(r)} + Y_{j16}^{(r)} \\
z_{j13}^{(r)} &= y_{j2}^{(r)} + y_{j6}^{(r)} + y_{j10}^{(r)} + y_{j11}^{(r)} + y_{j12}^{(r)} + y_{j14}^{(r)} + y_{j16}^{(r)} \\
z_{j14}^{(r)} &= y_{j3}^{(r)} + y_{j7}^{(r)} + y_{j11}^{(r)} + y_{j12}^{(r)} + y_{j13}^{(r)} + y_{j15}^{(r)} \\
z_{j15}^{(r)} &= y_{j1}^{(r)} + y_{j4}^{(r)} + y_{j5}^{(r)} + y_{j8}^{(r)} + y_{j12}^{(r)} + y_{j13}^{(r)} + y_{j14}^{(r)} + y_{j16}^{(r)} \\
z_{j16}^{(r)} &= y_{j1}^{(r)} + y_{j5}^{(r)} + y_{j9}^{(r)} + y_{j10}^{(r)} + y_{j11}^{(r)} + y_{j12}^{(r)} + y_{j13}^{(r)} + y_{j15}^{(r)} + y_{j16}^{(r)}
\end{aligned}$$

where $i \in \{1, 2, 3, 4\}$ and $j_k = 4i - 4 + k$, $j_{4+k} = 4i + 12 + k$, $j_{8+k} = 4i + 28 + k$, and $j_{12+k} = 4i + 44 + k$ for $k = 1, 2, 3, 4$.

Final Key Addition. For $i = 1, \ldots, 64$, the equations $c_i = z_i^{(32)} + k_i$ describe the final key addition and finish the algebraic representation of the `LED-64` block cipher. It is clear that `LED-128` has a similar description, using additional indeterminates for the second key and the extra rounds.

3 Algebraic Representation of the Fault Equations

The algebraic representation of LED-64 constructed above is not suitable to launch a successful algebraic attack. It involves too many non-linear equations in too many indeterminates. To reconstruct the secret key from given (correct or faulty) plaintext – ciphertext pairs requires additional information. This information will be furnished by a fault attack. In [12] we discussed a method for injecting fault and using it to break LED-64 by exhaustive search. In the following, we construct a polynomial version of the fault equations which were generated there.

Let us recall the description of the attack. We assume the following fault model. The attacker is supposed to be able to encrypt the same plain text unit twice using the same secret key k. The first encryption takes place correctly, and during the second encryption a fault is introduced. The fault is a random change in the value of the first (4-bit sized) entry of the state matrix at the beginning of round 30. As a consequence, we obtain a correct ciphertext c and a faulty ciphertext c'.

The propagation of the fault is observed. It leads to an incorrect first column of the state matrix after the SBox has been applied in round 31 whose 4-bit entries we denote by a, b, c, d. In [12] we derived the following 16 fault equations for a, b, c, d.

$$\begin{aligned}
a = \mathtt{D} \cdot (S^{-1}(\mathtt{C} \cdot (\bar{c}_1 + \bar{k}_1) + \mathtt{C} \cdot (\bar{c}_5 + \bar{k}_5) + \mathtt{D} \cdot (\bar{c}_9 + \bar{k}_9) + \mathtt{4} \cdot (\bar{c}_{13} + \bar{k}_{13})) + \\
S^{-1}(\mathtt{C} \cdot (\bar{c}'_1 + \bar{k}_1) + \mathtt{C} \cdot (\bar{c}'_5 + \bar{k}_5) + \mathtt{D} \cdot (\bar{c}'_9 + \bar{k}_9) + \mathtt{4} \cdot (\bar{c}'_{13} + \bar{k}_{13}))) \qquad (E_{a,0}) \\
a = \mathtt{F} \cdot (S^{-1}(\mathtt{3} \cdot (\bar{c}_4 + \bar{k}_4) + \mathtt{8} \cdot (\bar{c}_8 + \bar{k}_8) + \mathtt{4} \cdot (\bar{c}_{12} + \bar{k}_{12}) + \mathtt{5} \cdot (\bar{c}_{16} + \bar{k}_{16})) + \\
S^{-1}(\mathtt{3} \cdot (\bar{c}'_4 + \bar{k}_4) + \mathtt{8} \cdot (\bar{c}'_8 + \bar{k}_8) + \mathtt{4} \cdot (\bar{c}'_{12} + \bar{k}_{12}) + \mathtt{5} \cdot (\bar{c}'_{16} + \bar{k}_{16}))) \qquad (E_{a,1}) \\
a = \mathtt{5} \cdot (S^{-1}(\mathtt{7} \cdot (\bar{c}_3 + \bar{k}_3) + \mathtt{6} \cdot (\bar{c}_7 + \bar{k}_7) + \mathtt{2} \cdot (\bar{c}_{11} + \bar{k}_{11}) + \mathtt{E} \cdot (\bar{c}_{15} + \bar{k}_{15})) + \\
S^{-1}(\mathtt{7} \cdot (\bar{c}'_3 + \bar{k}_3) + \mathtt{6} \cdot (\bar{c}'_7 + \bar{k}_7) + \mathtt{2} \cdot (\bar{c}'_{11} + \bar{k}_{11}) + \mathtt{E} \cdot (\bar{c}'_{15} + \bar{k}_{15}))) \qquad (E_{a,2}) \\
a = \mathtt{9} \cdot (S^{-1}(\mathtt{D} \cdot (\bar{c}_2 + \bar{k}_2) + \mathtt{9} \cdot (\bar{c}_6 + \bar{k}_6) + \mathtt{9} \cdot (\bar{c}_{10} + \bar{k}_{10}) + \mathtt{D} \cdot (\bar{c}_{14} + \bar{k}_{14})) + \\
S^{-1}(\mathtt{D} \cdot (\bar{c}'_2 + \bar{k}_2) + \mathtt{9} \cdot (\bar{c}'_6 + \bar{k}_6) + \mathtt{9} \cdot (\bar{c}'_{10} + \bar{k}_{10}) + \mathtt{D} \cdot (\bar{c}'_{14} + \bar{k}_{14}))) \qquad (E_{a,3})
\end{aligned}$$

$$\begin{aligned}
b = \mathtt{1} \cdot (S^{-1}(\mathtt{C} \cdot (\bar{c}_4 + \bar{k}_4) + \mathtt{C} \cdot (\bar{c}_8 + \bar{k}_8) + \mathtt{D} \cdot (\bar{c}_{12} + \bar{k}_{12}) + \mathtt{4} \cdot (\bar{c}_{16} + \bar{k}_{16})) + \\
S^{-1}(\mathtt{C} \cdot (\bar{c}'_4 + \bar{k}_4) + \mathtt{C} \cdot (\bar{c}'_8 + \bar{k}_8) + \mathtt{D} \cdot (\bar{c}'_{12} + \bar{k}_{12}) + \mathtt{4} \cdot (\bar{c}'_{16} + \bar{k}_{16}))) \qquad (E_{b,0}) \\
b = \mathtt{7} \cdot (S^{-1}(\mathtt{3} \cdot (\bar{c}_3 + \bar{k}_3) + \mathtt{8} \cdot (\bar{c}_7 + \bar{k}_7) + \mathtt{4} \cdot (\bar{c}_{11} + \bar{k}_{11}) + \mathtt{5} \cdot (\bar{c}_{15} + \bar{k}_{15})) + \\
S^{-1}(\mathtt{3} \cdot (\bar{c}'_3 + \bar{k}_3) + \mathtt{8} \cdot (\bar{c}'_7 + \bar{k}_7) + \mathtt{4} \cdot (\bar{c}'_{11} + \bar{k}_{11}) + \mathtt{5} \cdot (\bar{c}'_{15} + \bar{k}_{15}))) \qquad (E_{b,1}) \\
b = \mathtt{3} \cdot (S^{-1}(\mathtt{7} \cdot (\bar{c}_2 + \bar{k}_2) + \mathtt{6} \cdot (\bar{c}_6 + \bar{k}_6) + \mathtt{2} \cdot (\bar{c}_{10} + \bar{k}_{10}) + \mathtt{E} \cdot (\bar{c}_{14} + \bar{k}_{14})) + \\
S^{-1}(\mathtt{7} \cdot (\bar{c}'_2 + \bar{k}_2) + \mathtt{6} \cdot (\bar{c}'_6 + \bar{k}_6) + \mathtt{2} \cdot (\bar{c}'_{10} + \bar{k}_{10}) + \mathtt{E} \cdot (\bar{c}'_{14} + \bar{k}_{14}))) \qquad (E_{b,2}) \\
b = \mathtt{9} \cdot (S^{-1}(\mathtt{D} \cdot (\bar{c}_1 + \bar{k}_1) + \mathtt{9} \cdot (\bar{c}_5 + \bar{k}_5) + \mathtt{9} \cdot (\bar{c}_9 + \bar{k}_9) + \mathtt{D} \cdot (\bar{c}_{13} + \bar{k}_{13})) + \\
S^{-1}(\mathtt{D} \cdot (\bar{c}'_1 + \bar{k}_1) + \mathtt{9} \cdot (\bar{c}'_5 + \bar{k}_5) + \mathtt{9} \cdot (\bar{c}'_9 + \bar{k}_9) + \mathtt{D} \cdot (\bar{c}'_{13} + \bar{k}_{13}))) \qquad (E_{b,3})
\end{aligned}$$

$$
\begin{aligned}
c = \mathtt{9} \cdot (S^{-1}(\mathtt{C} \cdot (\bar{c}_3 + \bar{k}_3) + \mathtt{C} \cdot (\bar{c}_7 + \bar{k}_7) + \mathtt{D} \cdot (\bar{c}_{11} + \bar{k}_{11}) + \mathtt{4} \cdot (\bar{c}_{15} + \bar{k}_{15})) + \\
S^{-1}(\mathtt{C} \cdot (\bar{c}'_3 + \bar{k}_3) + \mathtt{C} \cdot (\bar{c}'_7 + \bar{k}_7) + \mathtt{D} \cdot (\bar{c}'_{11} + \bar{k}_{11}) + \mathtt{4} \cdot (\bar{c}'_{15} + \bar{k}_{15}))) \qquad (E_{c,0}) \\
c = \mathtt{B} \cdot (S^{-1}(\mathtt{3} \cdot (\bar{c}_2 + \bar{k}_2) + \mathtt{8} \cdot (\bar{c}_6 + \bar{k}_6) + \mathtt{4} \cdot (\bar{c}_{10} + \bar{k}_{10}) + \mathtt{5} \cdot (\bar{c}_{14} + \bar{k}_{14})) + \\
S^{-1}(\mathtt{3} \cdot (\bar{c}'_2 + \bar{k}_2) + \mathtt{8} \cdot (\bar{c}'_6 + \bar{k}_6) + \mathtt{4} \cdot (\bar{c}'_{10} + \bar{k}_{10}) + \mathtt{5} \cdot (\bar{c}'_{14} + \bar{k}_{14}))) \qquad (E_{c,1}) \\
c = \mathtt{C} \cdot (S^{-1}(\mathtt{7} \cdot (\bar{c}_1 + \bar{k}_1) + \mathtt{6} \cdot (\bar{c}_5 + \bar{k}_5) + \mathtt{2} \cdot (\bar{c}_9 + \bar{k}_9) + \mathtt{E} \cdot (\bar{c}_{13} + \bar{k}_{13})) + \\
S^{-1}(\mathtt{7} \cdot (\bar{c}'_1 + \bar{k}_1) + \mathtt{6} \cdot (\bar{c}'_5 + \bar{k}_5) + \mathtt{2} \cdot (\bar{c}'_9 + \bar{k}_9) + \mathtt{E} \cdot (\bar{c}'_{13} + \bar{k}_{13}))) \qquad (E_{c,2}) \\
c = \mathtt{8} \cdot (S^{-1}(\mathtt{D} \cdot (\bar{c}_4 + \bar{k}_4) + \mathtt{9} \cdot (\bar{c}_8 + \bar{k}_8) + \mathtt{9} \cdot (\bar{c}_{12} + \bar{k}_{12}) + \mathtt{D} \cdot (\bar{c}_{16} + \bar{k}_{16})) + \\
S^{-1}(\mathtt{D} \cdot (\bar{c}'_4 + \bar{k}_4) + \mathtt{9} \cdot (\bar{c}'_8 + \bar{k}_8) + \mathtt{9} \cdot (\bar{c}'_{12} + \bar{k}_{12}) + \mathtt{D} \cdot (\bar{c}'_{16} + \bar{k}_{16}))) \qquad (E_{c,3})
\end{aligned}
$$

$$
\begin{aligned}
d = \mathtt{9} \cdot (S^{-1}(\mathtt{C} \cdot (\bar{c}_2 + \bar{k}_2) + \mathtt{C} \cdot (\bar{c}_6 + \bar{k}_6) + \mathtt{D} \cdot (\bar{c}_{10} + \bar{k}_{10}) + \mathtt{4} \cdot (\bar{c}_{14} + \bar{k}_{14})) + \\
S^{-1}(\mathtt{C} \cdot (\bar{c}'_2 + \bar{k}_2) + \mathtt{C} \cdot (\bar{c}'_6 + \bar{k}_6) + \mathtt{D} \cdot (\bar{c}'_{10} + \bar{k}_{10}) + \mathtt{4} \cdot (\bar{c}'_{14} + \bar{k}_{14}))) \qquad (E_{d,0}) \\
d = \mathtt{7} \cdot (S^{-1}(\mathtt{3} \cdot (\bar{c}_1 + \bar{k}_1) + \mathtt{8} \cdot (\bar{c}_5 + \bar{k}_5) + \mathtt{4} \cdot (\bar{c}_9 + \bar{k}_9) + \mathtt{5} \cdot (\bar{c}_{13} + \bar{k}_{13})) + \\
S^{-1}(\mathtt{3} \cdot (\bar{c}'_1 + \bar{k}_1) + \mathtt{8} \cdot (\bar{c}'_5 + \bar{k}_5) + \mathtt{4} \cdot (\bar{c}'_9 + \bar{k}_9) + \mathtt{5} \cdot (\bar{c}'_{13} + \bar{k}_{13}))) \qquad (E_{d,1}) \\
d = \mathtt{2} \cdot (S^{-1}(\mathtt{7} \cdot (\bar{c}_4 + \bar{k}_4) + \mathtt{6} \cdot (\bar{c}_8 + \bar{k}_8) + \mathtt{2} \cdot (\bar{c}_{12} + \bar{k}_{12}) + \mathtt{E} \cdot (\bar{c}_{16} + \bar{k}_{16})) + \\
S^{-1}(\mathtt{7} \cdot (\bar{c}'_4 + \bar{k}_4) + \mathtt{6} \cdot (\bar{c}'_8 + \bar{k}_8) + \mathtt{2} \cdot (\bar{c}'_{12} + \bar{k}_{12}) + \mathtt{E} \cdot (\bar{c}'_{16} + \bar{k}_{16}))) \qquad (E_{d,2}) \\
d = \mathtt{5} \cdot (S^{-1}(\mathtt{D} \cdot (\bar{c}_3 + \bar{k}_3) + \mathtt{9} \cdot (\bar{c}_7 + \bar{k}_7) + \mathtt{9} \cdot (\bar{c}_{11} + \bar{k}_{11}) + \mathtt{D} \cdot (\bar{c}_{15} + \bar{k}_{15})) + \\
S^{-1}(\mathtt{D} \cdot (\bar{c}'_3 + \bar{k}_3) + \mathtt{9} \cdot (\bar{c}'_7 + \bar{k}_7) + \mathtt{9} \cdot (\bar{c}'_{11} + \bar{k}_{11}) + \mathtt{D} \cdot (\bar{c}'_{15} + \bar{k}_{15}))) \qquad (E_{d,3})
\end{aligned}
$$

In these equations, the indeterminates $\bar{k}_1, \ldots, \bar{k}_{16}$ represent the 4-bit parts of the secret key, the indeterminates $\bar{c}_1, \ldots, \bar{c}_{16}$ represent the parts of the correct ciphertext, and $\bar{c}'_1, \ldots, \bar{c}'_{16}$ the parts of the faulty ciphertext. Since these equations involve the map $S^{-1} : \mathbb{F}_{16} \longrightarrow \mathbb{F}_{16}$, we need to find a polynomial representation of this map. Using the values of this map given in Table 3 and

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S^{-1}(x)$	5	E	F	8	C	1	2	D	B	4	6	3	0	7	9	A

Table 3. The inverse LED SBox.

univariate interpolation, we construct the following polynomial representation of S^{-1}.

$$
\begin{aligned}
S^{-1}(y) = (x^2+1) + (x^2+1)y + (x^3+x)y^2 + (x^3+x^2+1)y^3 + xy^4 + \\
(x^3+1)y^5 + (x^3+1)y^7 + (x+1)y^9 + (x^2+1)y^{10} + (x^3+1)y^{11} + \\
(x^3+x)y^{12} + (x+1)y^{13} + (x^3+x^2+1)y^{14}
\end{aligned}
$$

Next, we plug the right-hand sides of the fault equations into this polynomial. We get 16 polynomial fault equations which are defined over the polynomial ring $\mathbb{F}_{16}[a, b, c, d, \bar{k}_1, \ldots, \bar{k}_{16}, \bar{c}_1, \ldots, \bar{c}_{16}, \bar{c}'_1, \ldots, \bar{c}'_{16}]$. For every group of equations $E_{t,0}, E_{t,1}, E_{t,2}, E_{t,3}$ having the same left-hand side $t \in \{a, b, c, d\}$, we can form three differences $E_{t,0} - E_{t,i} = 0$ with $i = 1, 2, 3$. Now, comparing coefficients for $\{1, x, x^2, x^3\}$ yields 48 equations in the bits $k_1, \ldots, k_{64}$ of the secret key, the bits $c_1, \ldots, c_{64}$ of the correct ciphertext, and the bits $c'_1, \ldots, c'_{64}$ of the faulty ciphertext. Notice that we can use the field equations $k_i^2 + k_i = 0$, $c_i^2 + c_i = 0$, and $(c'_i)^2 + c'_i = 0$ for simplification here.

Altogether, we find 48 polynomials in $\mathbb{F}_2[k_1, \ldots, k_{64}, c_1, \ldots, c_{64}, c'_1, \ldots, c'_{64}]$. They all have degree 3 and consist of approximately 3400-8800 terms. These polynomials will be called the **fault polynomials**.

4 An Algebraic Fault Attack on LED-64

4.1 Description of the Attack

In the preceding two sections we derived polynomials describing the encryption map of LED-64 and additional information gained from a fault attack. All in all, we found 6208 polynomials in 6336 indeterminates describing the encryption map, 6336 field equations, and 48 fault polynomials in 192 indeterminates.

As mentioned previously, we assume that we are able to mount a known-plaintext-attack and a repeat encryption involving the same key and the fault injection described previously. For every concrete instance of this attack, we can therefore substitute the plaintext bits, correct ciphertext bits, and faulty ciphertext bits into our polynomials. After this substitution, we have 6208 polynomials in 6208 indeterminates for the encryption map, 6208 field equations, and 48 fault polynomials in the 64 indeterminates of the secret key.

The resulting fault polynomials consist typically of 40-150 terms. Some of them (usually no more than 5) drop their degree and become linear. Of course, these linear polynomials are particularly valuable, since they decrease the complexity of the problem by one dimension. In the experiments reported below it turned out to be beneficial to interreduce the fault polynomials after substitution in order to generate more linear ones.

The polynomial systems can be solved using various techniques. For our experiments, we applied the algorithms for conversion to a SAT-solving problem explained in [11].

4.2 Experimental Results

All experiments were performed on a workstation having eight 3.5 GHz Xeon cores and 50 GB of RAM. We used the SAT-solvers **Minisat 2.2** (MS) and **CryptoMiniSat 2.9.4** (CMS). All timings are averages over ten LED-64 instances with random plaintext, key and fault values. Table 4 contains the timings for the straightforward application of the SAT-solving technique to the given polynomial systems.

SAT solver	MS (1 thread)	CMS (1 thread)	CMS (4 threads)
time (in sec)	90852	71656	22639
time (in h)	25.23	19.90	6.28

Table 4. Average SAT Solver Timings.

For the second set of experiments, we first interreduced the fault polynomials using the computer algebra system **ApCoCoA** (see [1]) and then appended the linear polynomials to the system. In this way we were sometimes able to find more linear dependencies between the key indeterminates, thereby reducing the dimension even further. Moreover, the SAT-solvers appear to benefit from this simplification, because it is typically the number of terms in a polynomial that complicates its logical representation. This seemingly minor modification results in a meaningful speed-up, as we can see in Table 5.

SAT solver	MS (1 thread)	CMS (1 thread)	CMS (4 threads)
time (in sec)	36665	52835	11829
time (in h)	10.18	14.67	3.28

Table 5. Average SAT Solver Timings with Additional Linear Equations.

In summary, it is clear that the proposed attack is able to break the `LED-64` encryption scheme. While it is slower than the direct fault attack presented in [12], it does not rely on the specific properties underlying the key filtering steps there, and it offers numerous possibilities for optimization.

5 Conclusions and Future Work

After providing a complete algebraic description of the `LED` block cipher and showing how to convert the previously introduced fault equations into fault polynomials, it turned out that the combined polynomial system was solvable by state-of-the-art SAT solvers. Therefore the idea of combining an algebraic attack on a block cipher with fault-injection cryptanalysis is able to break the `LED-64` encryption algorithm in practice. This demonstrates that analysing the results of fault injection by algebraic methods is a promising approach. It may make it possible to attack ciphers for which the polynomial system resulting from a purely algebraic attack could not be solved in a reasonable time previously.

For the future, we plan to extend this attack technique in several ways. By optimizing the solving step further along the lines of Section 4, we expect to gain two major benefits. First, an improvement of the SAT solver running time, e.g. through modeling the *key set filtering* process (see [12]) by polynomials and adding those to the overall system. Second, the ability to adapt the presented

methods to more challenging encryption schemes such as `LED-128` or `PRESENT`. Furthermore, we plan to generalize the attack to other kinds of block ciphers, e.g. to Feistel network block ciphers such as `DES`, and to public key encryption schemes.

Acknowledgements. The authors are grateful to Mate Soos for numerous discussions and valuable information about the program CryptoMiniSat, about SAT-solving in general, and for being always ready to help with solver-related problems.

References

1. ApCoCoA: Applied Computations in Commutative Algebra, available for download at `http://www.apcocoa.org`.
2. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, C. Whelan, The Sorcerer's Apprentice Guide to Fault Attacks, Proceedings of the IEEE, vol. **94**, IEEE Computer Society, 2006, pp. 370–382.
3. E. Biham ans O. Dunkelman, Techniques for cryptanalysis of block ciphers, Springer, Heidelberg 2011.
4. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe, PRESENT: An Ultra-Lightweight Block Cipher, In: P. Paillier and I. Verbauwhede (eds.) *CHES* 2007, LNCS, vol. **4727**, Springer, Heidelberg 2007, pp. 450–466.
5. D. Boneh, R.A. DeMillo and R.J. Lipton, On the Importance of Elimination Errors in Cryptographic Computations, J. Cryptology **14** (2001), 101–119.
6. C. Carlet, J-C. Faugere, C. Goyet, G. Renault, Analysis of the algebraic side channel attack, Journal of Cryptographic Engineering, vol. **2** nr. 1, Springer Heidelberg 2012, pp. 45–62.
7. J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher, In: B. Preneel and T. Takagi (eds.) *CHES* 2011, LNCS, vol. **6917**, Springer, Heidelberg 2011, pp. 326–341.
8. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, S. Chee, HIGHT: A New Block Cipher Suitable for Low-Resource Device, In: L. Goubin and M. Matsui (eds.) *CHES* 2006, LNCS, vol. **4249**, Springer, Heidelberg 2006, pp. 46–59.
9. M. Hojsík and B. Rudolf, Differential Fault Analysis of Trivium, In: K. Nyberg (ed.) *FSE* 2008, LNCS, vol. **5086**, Springer, Heidelberg 2008, pp. 158–172.
10. M. Hojsík and B. Rudolf, Floating Fault Analysis of Trivium, In: D.R. Chowdhury, V. Rijmen and A. Das (eds.) *INDOCRYPT* 2008, LNCS, vol. **5365**, Springer, Heidelberg 2008, pp. 239–250.
11. P. Jovanovic and M. Kreuzer, Algebraic Attacks using SAT-Solvers, Groups – Complexity – Cryptology **2** (2010), pp. 247–259.
12. P. Jovanovic, M. Kreuzer, I. Polian, A Fault Attack on the LED Block Cipher, In: W. Schindler and S. Huss (eds.) *COSADE* 2012, LNCS, vol. **7275**, Springer Heidelberg 2012, pp. 120–134.
13. C.H. Kim and J-J. Quisquater, Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures, In: D. Sauveron, C. Markantonakis, A. Bilas and J-J. Quisquater (eds.) *WISTP* 2007, LNCS, vol. **4462**, Springer, Heidelberg 2007, pp. 215–228.

14. M. Kreuzer and L. Robbiano, Computational Commutative Algebra 1, Springer Verlag, Heidelberg 2000.
15. M.S.E. Mohamed, S. Bulygin and J. Buchmann, Using SAT Solving to Improve Differential Fault Analysis of Trivium, In: T-H. Kim, H. Adeli, R.J. Robles and M.O. Balitanas (eds.) *ISA* 2011, CCIS, vol. **200**, Springer, Heidelberg 2011, pp. 62–71.
16. D. Mukhopadhyay, An Improved Fault Based Attack of the Advanced Encryption Standard, In: B. Preneel (ed.) *AFRICACRYPT* 2009, LNCS, vol. **5580**, Springer, Heidelberg 2009, pp. 421–434.
17. National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES). FIPS Publication 197, available for download at `http://www.itl.nist.gov/fipsbups/`, 2001.
18. M. Renauld, F-X. Standaert and N. Veyrat-Charvillon, Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA, In: C. Clavier and K. Gaj (eds.) *CHES* 2009, LNCS, vol. **5747**, Springer Heidelberg 2009, pp. 97–111.
19. M. Tunstall, D. Mukhopadhyay and S. Ali, Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault, In: C.A. Ardagna and J. Zhou (eds.) *WISTP* 2011, LNCS, vol. **6633**, Springer Heidelberg 2011, pp. 224–233.