Differential Analysis of the LED Block Cipher *

Florian Mendel, Vincent Rijmen, Deniz Toz, Kerem Varıcı

KU Leuven, ESAT/COSIC and IBBT, Belgium
{florian.mendel,vincent.rijmen,deniz.toz,kerem.varici}@esat.kuleuven.be

Abstract. In this paper, we present a security analysis of the lightweight block cipher LED proposed by Guo et al. at CHES 2011. Since the design of LED is very similar to the Even-Mansour scheme, we first review existing attacks on this scheme and extend them to related-key and related-key-cipher settings before we apply them to LED. We obtain results for 12 and 16 rounds (out of 32) for LED-64 and 16 and 24 rounds (out of 48) for LED-128. Furthermore, we present an observation on LED in the related-key-cipher setting. For all these attacks we need to find good differentials for one step (4 rounds) of LED. Therefore, we extend the study of plateau characteristics for AES-like structures from two rounds to four rounds when the key addition is replaced with a constant addition. We introduce an algorithm that can be used to find good differentials and right pairs for one step of LED. To be more precise, we can find more than 2^{10} right pairs for one step of LED with complexity of 2^{16} and memory requirement of 5×2^{17}. Moreover, a similar algorithm can also be used to find iterative characteristics for LED.

1 Introduction

Security in embedded systems, such as RFID and sensor networks, where the area is restricted is getting more and more important since people started interacting with them in daily life more often. Improving the efficiency while preserving the security is one of the main challenges in this area and it has been an ongoing research problem. Recently, many algorithms have been developed to address this problem: hash functions like QUARK [1], PHOTON [13], SPONGENT [3] as well as block ciphers like Piccolo [22], LED [14], TWINE [23] and Klein [12]. Each of them uses the advantage of the improved knowledge on the design and analysis of symmetric key components.

LED [14] is a lightweight block cipher proposed by Guo et al. at CHES 2011. While being dedicated to compact hardware implementation with one of the smallest area consumptions (among block ciphers with comparable parameters), LED also offers reasonable performance in software. The design bears some resemblance with the (generalized) Even-Mansour construction [4] with the difference that the same key is used in each step for LED-64 or every second step in the case of the larger variant LED-128. The step function is based on AES-like design principles that provide good bounds against large classes of attacks including differential and linear cryptanalysis. Additionally, LED offers strong security arguments against attacks even in the related-key model.

To the best of our knowledge, no external analysis of LED with respect to differential cryptanalysis has been published so far. The best existing differential attacks are distinguishers for 15 (out of 32) rounds of LED-64 and 27 (out of 48) rounds of LED-128 in a hash setting, where the key is known to (or even chosen by) the attacker, described by the designers. Moreover, the security of LED against meet-in-the-middle attack has been investigated recently by Isobe et al. [16]. They describe attacks for 8 (out of 32) and 16 (out of 48) rounds for LED-64 and LED-128, respectively.

Our contribution. In this paper, we present the first external cryptanalysis of LED with respect to differential cryptanalysis. First, we show attacks for LED-64 reduced to 12 and 16 rounds. Furthermore, we present an observation on LED in the related-cipher setting [24]. All our attacks are based on the attack of Daemen [5] on Even-Mansour construction [11] that is extended in a straightforward way to the related-key setting.

*This work was sponsored he Research Fund KU Leuven, OT/08/027, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT Programme under Contract ICT-2007-216676 (ECRYPT II).

Secondly, we show how to improve the bound for the maximum expected differential probability (MEDP) for four rounds (one STEP) of LED from 2^{-32} to $2^{-41.75}$ using mega-boxes and the result of Park et al. [19].

Furthermore, we present algorithms to find differential characteristics with high probability that can be used in our attacks. By using the ideas of plateau characteristics [9] and extending the work with mega boxes [6], we are able to obtain characteristics for four rounds of LED. We find more than 2^{10} right pairs for a differential with a complexity less than 2^{16} time and 5×2^{17} memory and an iterative characteristic with six right pairs with the same complexities. We emphasize that our method is not specific to the block cipher LED and it can be used in the analysis of any AES-like construction where the key addition is replaced with a constant addition.

Outline. This paper is organized as follows. In Section 2 we give a brief description of LED and introduce the required definitions for our analysis. In Section 3, we describe the attacks on Even-Mansour construction and show how they can be extended to attack LED. We continue with differential analysis and give an algorithm to find the number of right pairs in a plateau characteristic in Section 4. We generalize this algorithm to find characteristics for four rounds of LED in Section 5 and we provide the results for characteristics with high probability and iterative characteristics that can be used in our attacks in Section 6.

2 Description of LED

LED [14] is a conservative lightweight block cipher whose design can be seen as a special case of the generalized Even and Mansour construction [11] depicted in Figure 1.

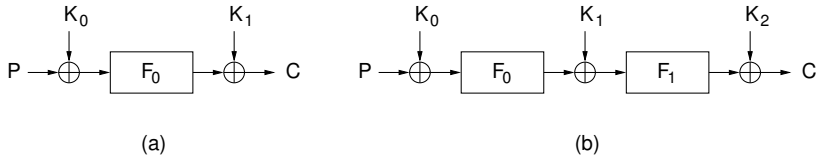

Fig. 1. Even-Mansour Construction with (a) $t = 1$ and (b) $t = 2$

LED accepts a 64-bit plaintext P, represented by a 4×4 array, and a 64-bit (or 128-bit) user key as inputs, and is composed of 8 (or 12) STEP functions preceded by a key addition. The STEP function is an AES-like design composed of four rounds. Each round is combination of Constant addition, S-boxes, ShiftRows, and (a variant of) MixColumns. LED uses the PRESENT S-box. In MixColumnsSerial, each column vector is multiplied by a matrix and replaced with the resulting vector. Note that the round constants for the second column are obtained from a linear shift register while the round constants for the remaining three columns do not change.

Key Schedule: LED has a simple key schedule where the 64-bit user key K is used as it is in each round whereas the 128-bit key is divided into two parts, $K = K_0||K_1$, and used alternately. For the remainder of this paper, we refer to these two versions as LED-64 and LED-128. For more detail, please check the specification of LED [14].

One observation is that the S-boxes and linear transformations in the round function of the cipher can be described by structure of a *super box*:

Definition 1 (Super box [9]). *A* super box *maps an array a of m elements a_i to an array e of m elements e_i. Each of the elements has size n. A super box takes a key k of size $m \times n = n_b$ where n_b is the block size. It consists of the sequence of four transformations (layers): Substitution, Mixing, Round Key Addition, Substitution.*

Similar to AES [7], two rounds of LED can also be described alternatively as four parallel instances of the LED *super box* where the key addition is replaced by the constant addition. So, instead of dealing with the

classical 4-bit S-boxes, one can consider 16-bit super boxes each composed of two S-box layers surrounding one `MixColumnsSerial` (MC) and one `AddConstants` (AC) function.

Four rounds of `LED` can be described as a mega-box, where the elements are 16-bit words and the `LED` super boxes defined above are seen as S-boxes. The linear transformation in the middle is a combination of `ShiftRows`, `MixColumnsSerial` and `ShiftRows` respectively. We will refer to this linear transformation as `SMS`.

3 Attacks on the Even-Mansour Construction and Application to `LED`

The Even-Mansour construction is a simple and yet provably secure block cipher construction. The designers have shown that the number of queries needed to break the scheme is bounded by $2^{n/2}$, where n is the blocklength ($n = 64$ for LED). A generic key recovery attack with chosen plaintexts showing that this bound is tight was introduced by Daemen [5]. Twenty years later, the construction was revisited. It was shown that the same bound applies to the known plaintext setting by using the slidex attack, an extended version of the slide attack [10].

Simultaneously, Bogdanov et al. generalized the construction in [4] to more steps and discussed its security. They even provided a security proof for the construction in the single-key setting. However, as pointed out by the authors, the scheme is insecure in the related-key setting. In this section, we focus on the attack of Daemen on the Even-Mansour construction, since it is the basis for all our attacks on `LED`. First we show how it can be extended to a related key attack on the generalized Even-Mansour construction. Then, we will use it to attack reduced versions of the `LED` block cipher.

3.1 Daemen's Attack

At Asiacrypt 1991 Daemen presented a generic key-recovery attack with complexity of $2^{n/2}$ [5]. It can be summarized as follows.

1. Choose a difference Δ.
2. For ℓ values of a compute $\Delta F_0 = F_0(a) \oplus F_0(a \oplus \Delta)$ and save the pair $(\Delta F_0, a)$ in a list L.
3. Choose an arbitrary plaintext P with $P' = P \oplus \Delta$ and ask for the ciphertexts C and C'
4. Compute $\Delta C = C \oplus C'$ and check if ΔC is in the list L to get a.
   - If ΔC is in the list L then a candidate for the key is found. Compute $K_0 = a \oplus P$ and $K_1 = F_0(a) \oplus C$.
   - Else go back to Step 3.

After repeating steps $3-4$ about $2^n/\ell$ times one expects to find the correct key with complexity of about $2^n/\ell + \ell$. Obviously the attack has the best complexity by choosing $\ell = 2^{n/2}$ resulting in a final attack complexity of about $2^{n/2}$ and similar memory requirements.

Note that, the attack can be applied in an iterative way to attack the Even-Mansour construction with $t > 1$ with complexity of $2^{t \cdot n/2}$ and similar memory requirements. For instance, if $t = 2$ then we get a complexity of 2^n.

3.2 Using Daemen's Attack in a Related-Key Setting

In certain scenarios one considers also related-key attacks where the adversary is allowed to get encryptions under several related keys. In this setting Daemen's attack can be adapted to attack t steps of the Even-Mansour construction with complexity of $t \cdot 2^{n/2}$ and similar memory requirements. For the sake of simplicity we first describe the attack for $t = 2$.

Related Key Attack with $t = 2$. Let K, K' be two related keys, where $K = K_0 \| K_1 \| K_2$ and $K' = K_0 \oplus \Delta_0 \| K_1 \oplus \Delta_1 \| K_2 \oplus \Delta_2$, with arbitrary (but known) $\Delta_0, \Delta_1, \Delta_2$ and $\Delta_1 \neq 0$. Then we can do a key recovery attack on the Even-Mansour construction with $t = 2$ with complexity of roughly $2^{n/2}$ and similar memory requirements using the attack of Daemen [5]. It can be summarized as follows.

1. For ℓ values of a compute $\Delta F_1 = F_1(a) \oplus F_1(a \oplus \Delta_1)$ and save the pair $(\Delta F_1, a)$ in a list L.
2. Choose an arbitrary P and $P' = P \oplus \Delta_0$ and ask for the ciphertexts C and C'
3. Compute $\Delta C = C \oplus (C' \oplus \Delta_2)$ and check if ΔC is in the list L to get a.
   - If ΔC is in the list L then a candidate for K_2 is found, $K_2 = F_1(a) \oplus C$.
   - Else go back to Step 2.

After repeating steps $2-3$ about $2^n/\ell$ times, the expected number of matches in the list L (i.e., candidates for K_2) is at least one. Note that, if we have more than one candidate for K_2 then we have to repeat the attack to get new candidates for K_2. The intersection of both sets of candidates gives us the correct key. Note that it is very unlikely that this intersection will have more than one solution.

Once K_2 is known one can apply the attack of Daemen to find K_0 and K_1. This results in a final attack complexity of about $2 \cdot 2^n/\ell + 2\ell$ and memory requirements of ℓ. Again, the attack has the best complexity by choosing $\ell \approx 2^{n/2}$ resulting in a final attack complexity of about $2 \cdot 2^{n/2}$ and memory requirements of $2^{n/2}$.

Related Key Attack with $t > 2$. The related key attack can be extended to more steps by applying the attack for $t = 2$ iteratively using more related keys with certain properties. Assume $t = 3$ and there are two related keys $K = K_0 \| K_1 \| K_2 \| K_3$ and $K' = K_0 \oplus \Delta_0 \| K_1 \| K_2 \oplus \Delta_2 \| K_3 \oplus \Delta_3$, with arbitrary (but known) $\Delta_0, \Delta_2, \Delta_3$ and $\Delta_2 \neq 0$. Then one can find K_3 similar as in the attack on the Even-Mansour construction with $t = 2$ with a complexity of roughly $2^{n/2}$. Once K_3 is found one can apply the attack for $t = 2$ with another pair of related keys to recover K_0, K_1 and K_2. In general, one can find the key for $t = i$ using i related keys with certain properties.

3.3 Attacks on Reduced LED

In this section, we will discuss the application of the attacks described in the previous section to the LED block cipher. Due to the fact that in LED the same key is used more than once the number of steps that can be attacked is significantly reduced. However, the attack can still be used in a straightforward way to break one and two steps of LED-64 in a single-key and related-key setting, respectively. Both attacks have a complexity of about $2^{n/2}$ and similar memory requirements. Note that a similar related-attack was described recently in [4].

However, both attacks can be extended to more steps in the case of LED-128. In more detail, we can attack four and six steps of LED-128 in the single-key and related-key setting, respectively. First, we describe an attack on four steps of LED-128 based on Daemen's attack. It is based on the following simple observation (cf. Figure 2).

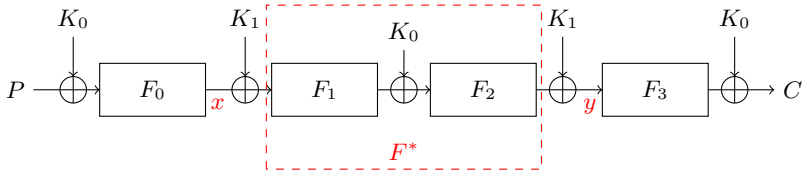

Fig. 2. Structure of LED-128 with $t = 4$.

Assume K_0 is known, then one can peel off the first and last key addition. Thus, the attacker can remove one iteration at each side of the cipher with a complexity of about 2^{64} tries on K_0. Moreover, assuming that K_0 is known two steps of LED-128 can be viewed as one big iteration using only K_1. In other words, we get a 'new' Even-Mansour construction with $t = 1$ and one key K_1 where we can apply Daemen's attack to recover the key. Using this, one can find K_0 and K_1 for four steps of LED-128 with a complexity of about $2^{3n/2}$. It can be summarized as follows.

1. Guess the key K_0.

2. For $2^{n/2}$ values a and a fixed Δ compute $\Delta F^* = F^*(a) \oplus F^*(a \oplus \Delta)$ with $F^*(a) = F_2(F_1(a) \oplus K_0)$ and save the pair $(\Delta F^*, a)$ in a list L.
3. Choose an arbitrary P and compute $P' = F_0^{-1}(x \oplus \Delta) \oplus K_0$ with $x = F_0(P \oplus K_0)$. Ask for the ciphertexts C and C'.
4. Compute $\Delta y = y \oplus y'$ with $y = F_3^{-1}(C \oplus K_0)$ and $y' = F_3^{-1}(C' \oplus K_0)$. Check if Δy is in the list L to get a.
   - If Δy is in the list L then a candidate for the key is found. Compute $K_1 = a \oplus x$.
   - Else go back to Step 3.
5. Once K_1 is found check if the key $K = K_0 \| K_1$ is correct.

Since the expected number of K_0 guesses that we need to make to find the correct key is 2^n, we need to repeat the attack 2^n times. Since for each guess of K_0 we need about $2^{n/2}$ computations to find K_1, the complexity of the attack is roughly $2^{3n/2}$. Note that the above attack needs the whole codebook. However, at the cost of a higher attack complexity, the data complexity of the attack can be reduced. To be more precise, in step 3 of the attack we can always choose P from a predefined subset and when computing P' we check if it is also in this subset, if not then we repeat this step. Thus, the data complexity of the attack can be reduced by simultaneously increasing the time complexity.

The attack can be extended to six steps of LED-128 using related keys as in the attack on the Even-Mansour construction with $t = 2$. The attack is very similar as the attack on four steps. Basically only steps $2 - 4$ (Daemen's attack) are replaced by the related key attack described in the previous section. The result is a key-recovery attack on six steps (24 rounds) of LED-128 with complexity of about $2^{3n/2}$. Again, as in the attack on 4 steps the data complexity of the attack can be reduced on the cost of a higher attack complexity.

3.4 Extending the Attack to more Steps

In this section, we discuss how the attacks can be extended to more steps of LED. First, we show that by exploiting differential properties of the STEP-function F, it might be possible to extend the attacks on LED-64 by one or two steps. Moreover, the attack on 4 steps can also be used in related-cipher attack [24] with related key setting on full LED-128. We call this a related-key-cipher attack. It is described in Appendix B.

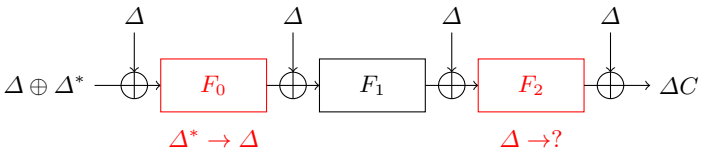

Fig. 3. Attack on LED-64 with $t = 3$.

In the following, we show how the attack can be extended to t steps of LED-64. The attack is based on the assumption that one can find a good related-key differential for the first $t - 2$ steps such that one gets a zero difference after the key addition of step $t - 2$. Then one can use Daemen's attack on the last 2 steps to recover the key. For In the attack on 3 steps we a differential with good probability in F_0 is used, see Figure 3. The attack can be summarized as follows.

1. Assume we have given two related keys K_0 and $K_0' = K_0 \oplus \Delta$ and furthermore the differential $\Delta^* \to \Delta$ for F_0 holds with probability $p \gg 2^{-64}$.
2. For $2^{n/2} \cdot (1/p)^{1/2}$ values a compute $\Delta F_2 = F_2(a) \oplus F_2(a \oplus \Delta)$ and save the pair $(\Delta F_2, a)$ in a list L.
3. Choose an arbitrary P and $P' = P \oplus \Delta^* \oplus \Delta$ and ask for the ciphertexts C and C'
4. Compute $\Delta C = C \oplus (C' \oplus \Delta)$ and check if ΔC is in the list L to get a.
   - If ΔC is in the list L then a candidate for K_0 is found, $K_0 = F_2(a) \oplus C$.
   - Else go back to step 3.

Table 1. Summary of the attacks on LED

algorithm	# STEP functions	time complexity	memory complexity	attack type	reference
LED-64	3	$2^{n/2} \cdot (1/p)^{1/2}$	$2^{n/2} \cdot (1/p)^{1/2}$	related-key	Section 3.4
	4	$2^{n/2} \cdot (1/p)^{1/2}$	$2^{n/2} \cdot (1/p)^{1/2}$	related-key	Section 3.4
LED-128	4	$2^{3n/2}$	$2^{n/2}$	single-key	Section 3.3
	6	$2^{3n/2}$	$2^{n/2}$	related-key	Section 3.3
	12	$2^{n/2} \cdot (1/p)^{1/2}$	$2^{n/2} \cdot (1/p)^{1/2}$	related-key-cipher	Appendix B

After repeating steps $3-4$ about $2^{n/2} \cdot (1/p)^{1/2}$ times, the expected number of matches in the list L (and hence candidates for the key K_0) is $1/p$. Since the differential in F_0 will hold with probability p, for one of these matches we will have $\Delta F_1 = 0$. Hence, one will find the right key after testing all candidates for K_0 resulting from the $1/p$ matches in the list L. The complexity and memory requirements of the attack depends on p, i.e. $2^{n/2} \cdot (1/p)^{1/2}$.

The attack on three steps can be extended to four steps of LED-64. Assume we can find a good iterative differential for F_1 that holds with probability p. Then this differential can be easily extended to a differential for the first 2 steps with the same probability (see Figure 4), resulting in an attack on 4 steps of LED-64 with complexity of $2^{n/2} \cdot (1/p)^{1/2}$ and similar memory requirements.

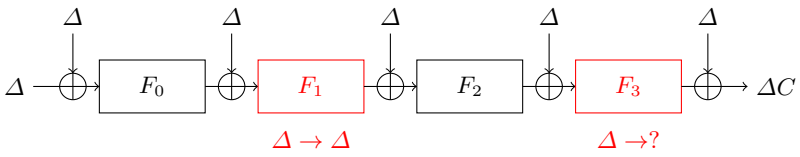

Fig. 4. Attack on LED-64 with $t = 4$.

In the Table 1, we summarize the attacks on LED that are given in Section 3 and Appendix B. We will discuss in the following sections how to find good (iterative) differential characteristics for one step of LED that can be used in the attacks on three and four steps.

4 Differential Analysis and Plateau Characteristics

In this section, we start with some definitions that will be helpful to understand the rest of the paper. We then give an introduction of the previous work on AES [9] and describe how we can use this method to find two/four round characteristics efficiently (and the corresponding right pairs).

4.1 Characteristics and Differentials

Differential cryptanalysis [2] is one of the most powerful techniques used in analysis of block ciphers, hash functions, stream ciphers, etc. It investigates how an input difference (generally XOR) propagates through the target function. The concept of differential cryptanalysis starts with analyzing the components of the function, mostly focusing on S-boxes since they are the smallest nonlinear building block. In the analysis, we call an S-box *active* if it has a non-zero input difference, otherwise we call it *passive*.

A *differential characteristic* $Q = (\Delta_0, \Delta_1, \cdots, \Delta_m)$ is a sequence of differences through various stages of the encryption. The sequence consists of an input difference Δ_0, followed by the output differences of all the steps $(\Delta_1, \Delta_2, \cdots, \Delta_m)$.

A *differential* [17] over a map is denoted by (Δ_0, Δ_m) where Δ_0 is the input difference and Δ_m is the output difference. The *differential probability* $\mathrm{DP}(\Delta_0, \Delta_m)$ of a differential over a map f is the fraction of pairs with input difference Δ_0 that have output difference Δ_m.

For a keyed map, we can define differential probabilities $\mathrm{DP}[k](\Delta_0, \Delta_m)$ and $\mathrm{DP}[k](Q)$ for each value k of the key. Then, the *expected differential probability* (EDP) is the average of the differential probability over all keys. The *weight* of a differential or a characteristic is minus the binary logarithm of their EDP. Moreover, we define the *height* of a possible differential or a characteristic as the binary logarithm of the number of their right pairs satisfying (Δ_0, Δ_m) for a fixed key.

A differential characteristic through the AES-like (including `LED`) super boxes consist of a sequence of four differences: the input difference a, the difference after the first substitution b, the difference after the mixing step which is equal to the difference after the round (key) constant addition d, and the output difference after the second substitution e. These characteristics are denoted by $Q = (a, b, d, e)$.

It can be shown that `SMS` is a map whose branch number is 5. Therefore, a characteristic over a mega-box consists of 5 to 8 sub-characteristics, each over an `LED` super box. We denote the characteristics over the first and the second layer of super boxes by (a, b, d, e) and (f, g, i, j), respectively.

4.2 The Maximum Expected Differential Probability of `LED`

Differential cryptanalysis plays a crucial role in the analysis of symmetric key components since most of the cryptanalysis techniques are based on it. Therefore, giving bounds for resistance against differential cryptanalysis is one of the first steps in the evaluation of a design. In `LED`, the AES-like structure in the `STEP` function makes it possible to apply the previous work of [19] to bound the MEDP. By a straightforward computation of the formula stated in [19, Theorem 4], the designers compute the bound for the MEDP as 2^{-32}. This bound can be improved by considering the `STEP` function as a mega-box and then using [19, Theorem 1] to bound the MEDP of `LED` as

$$\max\left\{\max_{\substack{1\le i\le 8\\ 1\le x\le 2^{16}-1}}\sum_{y=1}^{2^{16}-1}\left\{\mathrm{DP}^{\mathrm{sb}_i}(x,y)\right\}^5, \max_{\substack{1\le i\le 8\\ 1\le x\le 2^{16}-1}}\sum_{y=1}^{2^{16}-1}\left\{\mathrm{DP}^{\mathrm{sb}_i}(y,x)\right\}^5\right\} = 2^{-41.75}.$$

Here $\mathrm{DP}^{\mathrm{sb}_i}(x, y)$ is the probability of the characteristic (x, y) for the i-th super box obtained from the Difference Distribution Table (DDT). This result improves the approximations used in [14, Table 1]. We provided the bound for the first `STEP` function; the results for the other super boxes are similar.

4.3 Planar differentials

Let γ be a map and let $F_{(a,b)}$, $G_{(a,b)}$ be the sets that contain the input values, respectively the output values, for the right pairs of the differential (a, b). i.e., $F_{(a,b)} = \{x|\gamma(x) + \gamma(x + a) = b\}$ and $G_{(a,b)} = \gamma(F_{(a,b)})$. A differential (a, b) is called a *planar differential*, if $F_{(a,b)}$ and $G_{(a,b)}$ form affine subspaces [9]. In that case, we can write:

$$F_{(a,b)} = p + U_{(a,b)}$$
$$G_{(a,b)} = q + V_{(a,b)},$$

where $U_{(a,b)}$ and $V_{(a,b)}$ are uniquely defined vector spaces, p any element in $F_{(a,b)}$ and q any element in $G_{(a,b)}$. Note that, if a differential (a, b) has exactly two or four right pairs, then it is always planar [9].

Plateau characteristics [9] are a special type of characteristics whose probability for each value k of the key, $DP[k](Q)$, depends on the key and can have only two values. For a fraction $2^{n_b-(weight(Q)+height(Q))}$ of the keys $DP[k](Q) = 2^{height(Q)-n_b}$ and for all other keys the it is zero. Note that the height is independent of the key.

Two-Round Plateau Characteristic Theorem states that a characteristic $Q = (a, b, c)$ over a map consisting of two steps with a key addition in between, in which the differentials (a, b) and (b, c) are planar, is a plateau characteristic with $height(Q) = dim(V_{(a,b)} \cap U_{(b,c)})$.

4.4 Algorithm for number of right pairs in a plateau characteristic

Here, we describe the algorithm to find the number of right pairs of a given characteristic $Q = (a, b, d, e)$ through a super box. If the sub-characteristics (a, d) and (d, e) are planar then we can use the *Two-Round Plateau Characteristic Theorem* to compute the right pairs. Our aim in the algorithm is to build the matrix B containing the basis vectors of $(M(V_{(a,b)}))$ and $U_{(d,e)}$ where M is the mixing operation and $M(V) = \{M(v)|v \in V\}$. We denote vectors by *rows* of n_b bits.

The first step of our algorithm is to determine $V_{(a,b)}$ and $U_{(d,e)}$. Since, the super box is a set of m parallel maps, $V_{(a,b)}$ and $U_{(d,e)}$ can be written as:

$$V_{(a,b)} = V_{(a_1,b_1)} \times V_{(a_2,b_2)} \times \cdots \times V_{(a_m,b_m)}$$
$$U_{(d,e)} = U_{(d_1,e_1)} \times U_{(d_2,e_2)} \times \cdots \times U_{(d_m,e_m)}$$

by using the Lemma 4 in [9]. Now, if $|G_{(a_i,b_i)}| > 0$, we are interested in the output values of the right pairs.

- If $|G_{(a_i,b_i)}| = 2$, then the right pairs have input values in the set $\{q + \{0, b_i\}\}$ for some q in $G_{(a_i,b_i)}$, the basis vector for $V_{(a_i,b_i)}$ being b_i.
- If $|G_{(a_i,b_i)}| = 2^k$ where $2 \leq k < n$, then $V_{(a_i,b_i)} =< b_i, \beta_i^1, \ldots, \beta_i^k >$ and hence $V_{(a_i,b_i)}$ is said to be spanned by b_i and β_i^j's.
- If $(a_i, b_i) = (0, 0)$ then $G_{(a_i,b_i)}$ covers the whole space and $V_{(a_i,b_i)} =< w_0, w_1, \cdots, w_{n-1} >$ where w_j is a coordinate vector (i.e. a vector with 1 at position j and zero at all other positions) and V is the standard basis.

Similarly, if $|F_{(d_i,e_i)}| > 0$, we are interested in the input values of the right pairs. When we find the right pairs for each parallel map we can compute the height by using Algorithm 1. The number of dependent rows in B gives $dim(M(V_{(a,b)}) \cap U_{(d,e)})$ which is equal to the height.

Algorithm 1 calls the following subroutines. `Add`(v) adds the vector v as a new row to the matrix B. `RowReduce` is the Gaussian Elimination and `RowCount` gives the number of nonzero rows of a matrix.

The algorithm also gives us an insight on how to find the right pairs which can be determined by intersecting the affine spaces $F_{(a,b)} \cap (G_{(b,c)} \oplus k)$. This can be efficiently done by preparing the set of linear equations to solve. We would like to emphasize that, for a fraction of the keys the right pairs exists and their values differ depending on the key. On the other hand, if the constant operation is used instead of the key addition operation in the cipher, then it is not guaranteed always to have a solution.

5 Non-plateau Characteristics: `LED` Mega-box

As we mentioned in Section 2, two rounds of `LED` can be considered as a super box and four rounds is defined as a mega-box. Let (a, b, d, e) and (f, g, i, j) denote the characteristic through the super boxes at the input and the output of `SMS` respectively. Since the super boxes are key independent, we consider them as 16-bit S-boxes. This allows us to omit the middle values (b, d) and (g, i) and use the differentials (a, e) and (f, j) in our analysis.

In order to use the two-round plateau characteristic theorem, it is required that the set of output values $G_{(a,e)}$ and the set of input values $F_{(f,j)}$ for the right pairs must be affine spaces/planar. However, this is not always guaranteed when the number of right pairs is greater than 4. Although the difference between the values of each pair is known and constant, some extra conditions between the pairs are also required for a set to become affine/planar. Therefore, we have to work with a union of affine spaces in order to compute the number of right pairs of a given characteristic. In the following, we will denote by height* the binary logarithm of the maximum number of right pairs of a given characteristic, over all values of the key. For a plateau characteristic, height* equals the height.

The details of our algorithm are given below. An algorithmic description can be found in Algorithm 2.

Algorithm 1 Algorithm to compute the height of a given plateau characteristic

Input: Characteristic $Q = (a, b, d, e)$ with $EDP(Q) > 0$
Output: height(Q)

procedure PRECOMPUTE
for $i = 1 \rightarrow m$ **do**
  Compute $V_{(a_i,b_i)} =< b_i, \beta_i^1, \ldots, \beta_i^{k_i^v} >$ and $U_{(d_i,e_i)} =< d_i, \delta_i^1, \ldots, \delta_i^{k_i^u} >$
end for
end procedure

procedure HEIGHT
//at the input of Mixing
for $i = 0 \rightarrow m$ **do**
  if $b_i = 0$ **then**
    for $j = 0 \rightarrow n$ **do**
      Add($M(w_{4i+j})$)
    end for
  else if $b_i > 0$ **then**
    Add($M(b_i)$)
    if $|V_{(a_i,b_i)}| > 2$ **then**
      for $j = 1 \rightarrow k_i^v$ **do**
        Add($M(\beta_i^j)$)
      end for
    end if
  end if
end for

//at the output of Mixing
for $i = 0 \rightarrow m$ **do**
  if $d_i = 0$ **then**
    for $j = 0 \rightarrow n$ **do**
      Add(w_{4i+j})
    end for
  else if $d_i > 0$ **then**
    Add(d_i)
    if $|U_{(d_i,e_i)}| > 2$ **then**
      for $j = 1 \rightarrow k_i^u$ **do**
        Add(δ_i^j)
      end for
    end if
  end if
end for

B' = RowReduce (B).
return height(Q) = RowCount(B) - RowCount(B')
end procedure

Precomputation: The first step of our algorithm is finding $G_{(a,e)}$ and $F_{(f,j)}$ for the given path, and the next step is obtaining the subspace decompositions of $V_{(a,e)}$ and $U_{(f,j)}$. If $V_{(a_i,e_i)}$ is affine then $V_{(a_i,e_i)} =< e, \varepsilon_1, \ldots, \varepsilon_n >$, otherwise it is a union of smaller vector spaces, i.e. $V_{(a_i,e_i)} = V^1_{(a_i,e_i)} \cup V^2_{(a_i,e_i)} \cup \ldots V^m_{(a_i,e_i)}$ where $m \geq 2$. Therefore, we have to find the corresponding basis vectors (ε_i's) for each subspace. The results are then stored in a list, L_i, for each active super box.

Analysis: We then use the Two-Round Plateau Characteristic Theorem to compute the height using the basis vectors obtained in the precomputation phase. Since the solution exist only for a fraction of the constant values, we check whether the given round constant is in the solution set or not. This step can also be done by solving a system of linear equations as in two-rounds, but this time the equations are obtained from the `SMS` layer and the basis vectors of the super boxes.

Here, we would like to emphasize that the solution does not always exist for the round constant of `LED`. Denote by K_q, the set of values, k, such that DP$[k](q) > 0$. Since constants are used in the round function of `LED`, it is not guaranteed that the round constant, $c_r \in K_q$ for all q. Therefore, the algorithm gives an upper bound for height$^*(Q)$. If the key addition was used in the round function rather than constant addition, it could be possible to find a key value $k \in K_q$ for all q satisfying the upper bound.

On the other hand, if the key addition was used, the Algorithm 2 could not be applied immediately, since the lists L_i would depend on the key values and would not be unique. This would require recomputation of the lists for each key value increasing the complexity of the algorithm.

Note that, since height* for four rounds is the summation over all possible decompositions $q \in L_0 \times L_1 \times \ldots \times L_7$ of the characteristic Q, height$^*(Q)$ is not guaranteed to be an integer, although height(q) is integer for all q.

Algorithm 2 Algorithm to compute the height* of a four-round characteristic

Input: Characteristic $Q = (a, e, f, j)$
Output: Upper bound for height*(Q)

procedure PRECOMPUTE
$L_0 = L_1 = \ldots = L_7 = \emptyset$
for $i = 0 \rightarrow 3$ **do**
  Compute $G_{(a_i,e_i)}$ and $F_{(f_i,j_i)}$

$$\bigcup_m V^m_{(a_i,e_i)} = \text{Decompose}(V_{(a_i,e_i)}) \text{ and } < \varepsilon^m_1, \varepsilon^m_2, \ldots \varepsilon^m_{d_m} >= V^m_{(a_i,e_i)},\ d_m = |V^m_{(a_i,e_i)}|$$

$$\bigcup_n U^n_{(f_i,j_i)} = \text{Decompose}(U_{(f_i,j_i)}) \text{ and } < \varepsilon^n_1, \varepsilon^n_2, \ldots \varepsilon^n_{d_n} >= V^n_{(f_i,j_i)},\ d_n = |V^n_{(f_i,j_i)}|$$

  Store$(L_i, \{(a_i, e_i), \varepsilon^m_1, \varepsilon^m_2, \ldots \varepsilon^m_{d_m}\})$ and Store$(L_{4+i}, \{(f_i, j_i), \varepsilon^n_1, \varepsilon^n_2, \ldots \varepsilon^n_{d_n}\})$
end for
end procedure

procedure ANALYZE
count = 0
for all $q \in L_0 \times L_1 \times \ldots \times L_7$ **do**
  $h =$ HEIGHT(q);
  count = count + 2^h
end for
return $\log_2$(count)
end procedure

In Algorithm 2, `Store` adds input/output differences and the basis vectors $\{\varepsilon_1, \varepsilon_2, \ldots\}$ to the list L. HEIGHT is given in Algorithm 1 used with parameters $m = 4$ and $n = 16$.

6 Application of the Algorithms 1 and 2

In this section, we give two examples to demonstrate how Algorithm 1 and Algorithm 2 work. These examples can directly be used with attacks described in Section 3.4. We do not claim that these are the best characteristics in terms of probability for the `STEP` function of `LED` that one can find. For both examples, we fix the number of active S-boxes to 25 for four rounds of `LED`. Since, we know from previous work [9] that all the characteristics with high probability are expected to have a low weight and a low number of active S-boxes. This also allows us to reduce the time and memory complexities of our algorithm and make the computation feasible.

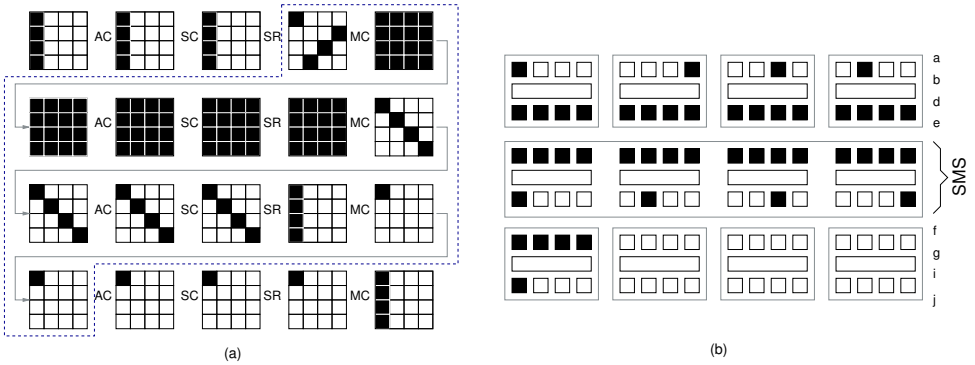

Fig. 5. (a)Path for iterative characteristics of the `LED` cipher (b)Mega-box representation of the same path

Algorithm 3 Compute iterative characteristics

Input: Precomputed tables L_i where $i \in \{1, \cdots, 5\}$
Output: All the iterative characteristics with their height*

1: **for all** $(e, f) \in$ S **do**
2: **if** $(f_0, j_0) \in L_4$ **then**
3: $\Delta = MC \circ SR \circ AC(j)$
4: $a = SR \circ AC(\Delta)$
5: **if** $(a_i, e_i) \in L_i$ for $1 \leq i \leq 4$ **then**
6: $h =$HEIGHT$^*(Q)$
7: **Output** $Q = (a, e, f, j)$ and h
8: **end if**
9: **end if**
10: **end for**

6.1 Iterative Characteristics

Our aim is to find iterative characteristics (i.e., characteristics that have the same input and output difference) for the `STEP` function of the `LED` block cipher. We show that it is possible to obtain multiple iterative characteristics by using the 16-bit boxes and the two round plateau characteristic theorem in 2^{16} time and around 5×2^{17} memory. In terms of efficiency, this computation can be compared with the inbound technique of the rebound attack [18]. The main advantage of our computation is that many characteristics can be found whereas with the rebound attack, the expected number of characteristics that we find, equals one, using the same time complexity and slightly less data complexity.

In our analysis we used the differential path given in Figure 5. It is possible to adopt the algorithm for the other possible differential paths. The algorithm is summarized as follows:

Precomputation: For each of the active super boxes, obtain the differentials (a_i, e_i) (or (f_i, j_i)) for the given path and find the corresponding right pairs $G_{(a_i,e_i)}$ (or $F_{(f_i,j_i)}$). Then compute their affine subspace decomposition and the corresponding basis vectors. Store the input/output differences together with the basis vectors in a list. We denote these lists as L_0, L_1, L_2, L_3 for the super boxes at the input and L_4 for the super box at the output of the `SMS` layer. Note that this calculation is done for all possible differentials. Each list has around 2^{17} elements, therefore the total memory requirement of this step is 5×2^{17}.

Analysis: We start from the four `MixColumnsSerial` operations in the `SMS` layer. Each of them has only one 4-bit word active at the output, hence we have $15^4 \approx 2^{16}$ possibilities for the differences at f (call the set of possibilities S). For each of these differences, we obtain the possible differences at j by using the precomputed list L_4. Then, we compute $(MC \circ SR \circ AC)(j) = \Delta$ which is the output difference after four rounds of the `STEP` function and is also equal to the input difference of the `STEP` function since we are interested in iterative characteristics. We make one more computation $(SR \circ AC)(\Delta)$ to obtain the difference at a. Note that by choosing a difference for f, we have already fixed the difference at e. We then check whether (a_i, e_i) is in the list L_i for $0 \leq i \leq 3$. If it does for all i, we use the Algorithm 2 to compute the height* and find the right pairs.

Results: In our analysis we found 240 iterative characteristics for the pattern given in Figure 5 but not all of them have a solution for the round constants of the `LED` block cipher. One of these characteristics is given below. It has 6 right pairs and the corresponding right pairs are given in Appendix A.

a	0x6000	0x0003	0x0070	0x0C00
e	0x6962	0x5848	0x46A3	0x5CBF
f	0x943C	0x0000	0x0000	0x0000
j	0x8000	0x0000	0x0000	0x0000

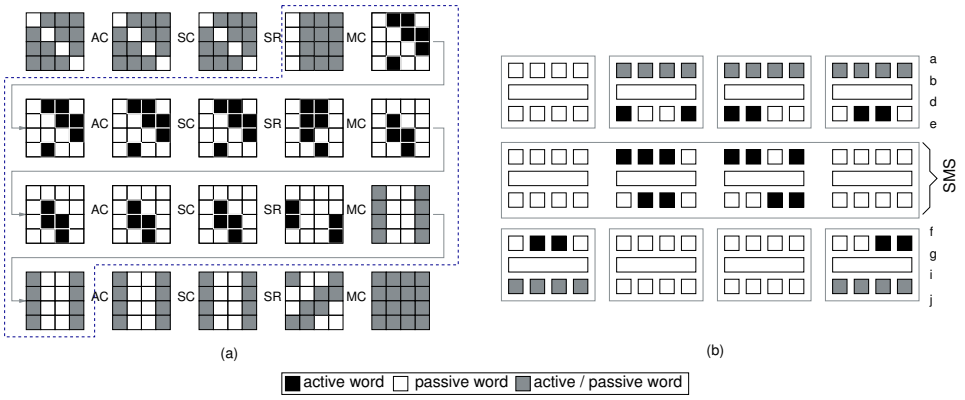

Fig. 6. (a)Characteristics of the LED cipher with high height* (b)Mega-box representation of the same characteristics

6.2 Characteristics with high height*

In this section, our aim is to find characteristics with high height* for the STEP function of the LED block cipher. We show that it is possible to obtain such characteristics by using a similar algorithm to Algorithm 3 with 2^{16} time complexity and 5×2^{17} memory complexity. In our analysis we focused on the differential path given in Figure 6 and searched for characteristics whose height* is greater than 5.

Precomputation: All possible differentials together with the basis vectors of their affine space decomposition are stored in the lists L_1, L_2, L_3 for each of the the super boxes at the input and in the lists L_4, L_7 for the super boxes at the output of the SMS layer. Again, each list has around 2^{17} elements, and the total the memory requirement of this step is 5×2^{17}.

Analysis: We start from the two active MixColumnsSerial operations in the SMS layer. Each of them has two 4-bit words active at the output, hence we have $(15^2)^2 \approx 2^{16}$ possibilities for the differences at f. For each of these possibilities, we obtain the possible differences at a by using the precomputed lists L_1, L_2 and L_3. Similarly, the possible differences at j are obtained by using the lists L_4 and L_7. We then use Algorithm 2 to compute the height and find the right pairs.

Results: Assume that $\dim(V_{(a,e)}) > 0$ and $\dim(U_{(f,j)}) > 0$, then we can write $V_{(a,e)} = \bigcup_m V^m_{(a,e)}$ and $U_{(f,j)} = \bigcup_n U^n_{(f,j)}$. We define a partition by Q_{mn} where $Q_{mn} = SMS(V^m_{(a,e)}) \cap U^n_{(f,j)}$. Then we know that height*$(Q) \leq log_2(\sum_{m,n} 2^{\dim(Q_{mn})})$ (see Algorithm 2). In our analysis we observed that it is not easy to find a partition whose height is greater than six, but by combining all partitions, we were able to find characteristics which have height* greater than eleven or twelve. One example of such characteristics is provided below.

a	0x0000	0x0F91	0x2F0B	0x2803
e	0x0000	0xC00D	0x8F00	0x0F50
f	0x0CD0	0x0000	0x0000	0x00C8
j	0x8C07	0x0000	0x0000	0x50BF

The upper bound for height* is computed as 12.16 by using the formula. However, not all partitions have a solution for the given round constant, and we obtain only 1026 right pairs for the round constants used in LED. We also computed the number of right pairs by changing the round constant used in round three of the STEP function. The number of right pairs is computed as $1024 \pm \epsilon$ where $\epsilon \leq 116$ for all constants.

To sum up, we introduced not only a new method that can be useful in the security evaluation of AES-like structures but we also showed that by using this method it is possible to obtain characteristics that can be used to attack LED (see Section 3.4).

7 Future Work and Open Problems

The analysis of super boxes and mega-boxes play an important role in the cryptanalysis of AES-like ciphers. In this paper, we focused on characteristics for the block cipher LED with 25 active S-boxes. Since it is not feasible to compute the whole distribution of the characteristics for four rounds of LED, we focus only on characteristics that may have many right pairs. Therefore, our results cover characteristics with high height* and iterative characteristics with a fixed pattern. The examples given in this paper are the best ones that we computed. But still, it is possible to cover other patterns with 25 active S-boxes and they might give better results and at the same time result in improvements of our attacks.

We want to note that the algorithms given in this paper can also be used to compute the differentials for constructions using four rounds of AES as internal building block such as Pelican [8] giving new insights on these designs. Moreover, these algorithms might also be used in the computation of the inbound phase of the rebound attack.

References

1. Aumasson, J.P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: A Lightweight Hash. In: Mangard, S., Standaert, F.X. (eds.) CHES. Lecture Notes in Computer Science, vol. 6225, pp. 1–15. Springer (2010)
2. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990)
3. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: SPONGENT: A Lightweight Hash Function. In: Preneel and Takagi [21], pp. 312–325
4. Bogdanov, A., Knudsen, L.R., Leander, G., Standaert, F.X., Steinberger, J.P., Tischhauser, E.: Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations - (Extended Abstract). In: Pointcheval and Johansson [20], pp. 45–62
5. Daemen, J.: Limitations of the Even-Mansour Construction. In: Imai et al. [15], pp. 495–498
6. Daemen, J., Lamberger, M., Pramstaller, N., Rijmen, V., Vercauteren, F.: Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. Computing 85(1-2), 85–104 (2009)
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
8. Daemen, J., Rijmen, V.: The Pelican MAC Function. IACR Cryptology ePrint Archive 2005, 88 (2005)
9. Daemen, J., Rijmen, V.: Plateau characteristics. Information Security, IET 1(1), 11 –17 (March 2007)
10. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In: Pointcheval and Johansson [20], pp. 336–354
11. Even, S., Mansour, Y.: A Construction of a Cipher From a Single Pseudorandom Permutation. In: Imai et al. [15], pp. 210–224
12. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A New Family of Lightweight Block Ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec. Lecture Notes in Computer Science, vol. 7055, pp. 1–18. Springer (2011)
13. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011)
14. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: Preneel and Takagi [21], pp. 326–341
15. Imai, H., Rivest, R.L., Matsumoto, T. (eds.): Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings, Lecture Notes in Computer Science, vol. 739. Springer (1993)
16. Isobe, T., Shibutani, K.: Security Analysis of the Lightweight Block Ciphers XTEA, LED and Piccolo. to appear in ACISP (2012)
17. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differentail cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer (1991)
18. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE. Lecture Notes in Computer Science, vol. 5665, pp. 260–276. Springer (2009)
19. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 247–260. Springer (2003)

20. Pointcheval, D., Johansson, T. (eds.): Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7237. Springer (2012)
21. Preneel, B., Takagi, T. (eds.): Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings, Lecture Notes in Computer Science, vol. 6917. Springer (2011)
22. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: Preneel and Takagi [21], pp. 342–357
23. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A Lightweight, Versatile Blockcipher. ECRYPT Workshop on Lightweight Cryptography (2011), http://www.uclouvain.be/crypto/ecrypt_lc11/static/post_proceedings.pdf
24. Wu, H.: Related-cipher attacks. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS. Lecture Notes in Computer Science, vol. 2513, pp. 447–455. Springer (2002)

A Right pairs for iterative characteristic of the first STEP function

```
{0x2CD65C01406D989B, 0x4CD69C01306DA89B}, {0x4CD69C01306DA89B, 0x2CD65C01406D989B}
{0x2DD65F71428EA8EF, 0x4DD69F71328E98EF}, {0x4DD69F71328E98EF, 0x2DD65F71428EA8EF}
{0x2FDE5CF1406098FB, 0x4FDE9CF13060A8FB}, {0x4FDE9CF13060A8FB, 0x2FDE5CF1406098FB}
```

B An Observation on the LED-128

In this section, we discuss the extension of attack on four steps for LED-64 to LED-128. We observe that if the key value for LED-128 is of the form $K = k \,||\, k$ for some k, then the first 8 STEPs of LED-128 is identical to LED-64 with the key value k and two ciphers differ only in four STEP functions. Therefore, in addition to previous attack settings, we need two additional assumptions to perform an attack. First, we assume that we have access to LED-64 decryption oracle under a key value k. Secondly, the key value for LED-128 must be set to $K = k \,||\, k$ as described above.

Attack on LED-128 in the related-cipher model. The attack can be summarized as follows:

1. Find an iterative differential in F_9 with high probability.
2. Assume we have given two related keys k, $k' = k \oplus \Delta$ and furthermore the differential $\Delta \rightarrow \Delta$ for F_9 holds with probability $p \gg 2^{-64}$.
3. For $2^{n/2} \cdot (1/p)^{1/2}$ values a compute $\Delta F_{11} = F_{11}(a) \oplus F_{11}(a \oplus \Delta)$ and save the pair $(\Delta F_{11}, a)$ in a list L.
4. Choose an arbitrary C_{64}, set $C'_{64} = C_{64}$ and ask for the plaintexts P and P' by using LED-64. Then, ask for corresponding ciphertexts C_{128}, C'_{128} by using LED-128.
5. Compute $\Delta C_{128} = C_{128} \oplus (C'_{128} \oplus \Delta)$ and check if ΔC_{128} is in the list L to get a.
   - If ΔC_{128} is in the list L then a candidate for k is found, $k = F_{11}(a) \oplus C_{128}$.
   - Else go back to Step 3.

The expected number of matches in the list L is $1/p$ and hence candidates for the key $K = k \,||\, k$ after repeating steps $3 - 4$ about $2^{n/2} \cdot (1/p)^{1/2}$ times. The complexity and memory requirements of the attack only depend on p and equal to $2^{n/2} \cdot (1/p)^{1/2}$.

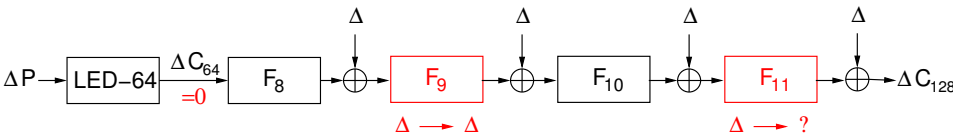

Fig. 7. Attack on LED-128 in the related-cipher model