Related-Key Attack on Full-Round PICARO

Anne Canteaut(✉), Virginie Lallemand, and María Naya-Plasencia

Inria, project-team SECRET, Rocquencourt, France
{Anne.Canteaut,Virginie.Lallemand,Maria.Naya_Plasencia}@inria.fr

Abstract. Side-channel cryptanalysis is a very efficient class of attacks that recover secret information by exploiting the physical leakage of a device executing a cryptographic computation. To address this type of attacks, many countermeasures have been proposed, and some papers addressed the question of constructing an efficient masking scheme for existing ciphers. In their work, G. Piret, T. Roche and C. Carlet took the problem the other way around and specifically designed a cipher that would be easy to mask. Their careful analysis, that started with the design of an adapted Sbox, leads to the construction of a 12-round Feistel cipher named PICARO. In this paper, we present the first full-round cryptanalysis of this cipher and show how to recover the key in the related-key model. Our analysis takes advantage of the low diffusion of the key schedule together with the non-bijectivity of PICARO Sbox. Our best trade-off has a time complexity equivalent to $2^{107.4}$ encryptions, a data complexity of 2^{99} plaintexts and requires to store 2^{17} (plaintext, ciphertext) pairs.

Keywords: Related-key attack · Differential cryptanalysis · PICARO

1 Introduction

While performance and side-channel attacks resistance are most of the time considered separately and as distinct problems — new design papers focus on performance figures while countermeasure papers focus on how to implement a specific protection in order to reduce performance overheads —, some new cipher proposals tackle the two problems together by designing new primitives that fit given protections. Examples of such constructions are PICARO [8], Zorro [6], and the family of LS-design [7] (including Robin and Fantomas as concrete instantiations).

The countermeasure studied in these three designs is the *masking scheme* [10] for which the heavier parts to protect are the operations which are not linear with respect to the group operation used to share the sensitive variables. For these three ciphers as for most ciphers, these non-linear operations are concentrated in the Sboxes, and then the straightforward way to limit the masking cost is to

Partially supported by the French Agence Nationale de la Recherche through the BLOC project under Contract ANR-11-INS-011.

O. Dunkelman and L. Keliher (Eds.): SAC 2015, LNCS 9566, pp. 86–101, 2016.
DOI: 10.1007/978-3-319-31301-6_5

reduce the number of Sbox applications[1] as well as to choose Sboxes that are masking-friendly, for instance by reducing the number of field multiplications processed[2]. This later direction is the one followed by Piret, Roche and Carlet while devising PICARO [8,9]: their design relies on a non-bijective Sbox defined by two bivariate polynomials over $GF(2^4)$, reducing the number of non-linear multiplications to four. This Sbox is integrated to the round function of a Feistel scheme, composed of four operations which are an expansion, a key addition, an Sbox-layer and a compression.

In this paper we show that despite the fact that the authors of PICARO aim at resisting to related-key attacks, as pointed out in [9][3], we have been able to mount related-key attacks on the full cipher. Our attacks exploit some weaknesses in the key-schedule, the non-bijective properties of the Sbox and some properties of the linear layer. They provide different trade-offs between time and data complexities.

The paper is organized as follows. In Sect. 2, we give a brief description of the PICARO block cipher and of its design choices, in Sect. 3 we give some definitions and in Sect. 4, we present an analysis of PICARO key-schedule that leads to the related-key attack described in Sect. 5. The paper ends with a conclusion.

2 Description of PICARO

2.1 Round Function

One of the main objectives of G. Piret, T. Roche and C. Carlet when designing PICARO was to propose a 128-bit block cipher that would get an advantage over other ciphers regarding the ease to protect against *side-channel attacks*. To achieve this, they started from Rivain and Prouff's masking scheme [10] and determined which operations are difficult to mask to derive some new design criteria. Their analysis brought to light that efforts have to focus on the Sboxes since the masking scheme implies heavy overheads for the non-linear operations.

This condition must be added to the usual criteria coming from the (non-physical) usual attacks (including the prominent linear, differential, algebraic cryptanalyses and their variants) that require the Sbox be highly non-linear, have a high algebraic degree and a low differential-uniformity.

Their deep analysis resulted in the selection of an Sbox defined as the concatenation of two polynomials:

$$\begin{aligned} S : GF(2^4) \times GF(2^4) &\rightarrow GF(2^4) \times GF(2^4) \\ (x, y) &\mapsto (xy, (x^3 + \texttt{0x02})(y^3 + \texttt{0x04})) \end{aligned}$$

where `0x02` and `0x04` represent elements of $GF(2^4)$ defined as $GF(2)[x]/(X^4 + X^3 + 1)$. Its full look-up table is given in Appendix A.

[1] This direction has been followed in the design of Zorro.

[2] When considering binary masking, this criterion is equivalent to limiting the number of AND processed (see for instance the LS-design [7]).

[3] Section 7.2 of [9]: "We want our scheme to resist known attacks on a key schedule algorithm, in particular related-key attacks...".

This non-bijective Sbox has already been studied in [5] and possesses the following desirable characteristics: a non-linearity equal to 94, an algebraic degree equal to 4, a maximal differential probability equal to 4/256. Furthermore, to fit well the masking protection, it can be implemented with only 4 non-linear operations; moreover all these non-linear operations are defined in the small field $GF(2^4)$.

An obvious but quite important remark that has to be made is that since the Sbox is not bijective, there exist sets of values that all have the same image through the Sbox. In other words, it is possible to cancel a difference entering the Sbox: a byte that is active before applying the Sbox can lead to an inactive output. Since the cube function is 3-to-1 over $GF(2^4)^*$, we deduce from the definition of the Sbox that it is a 3-to-1 function over $GF(2^4)^2 \backslash \{(0,0)\}$. Moreover, it is very easy to prove that, for any non-zero input difference Δ, the transition $\Delta \rightarrow 0$ holds with probability 2^{-7}. In other words, the equation $S(x+\Delta)+S(x)=0$ has exactly two solutions x.

The choice of a non-bijective Sbox was motivated by the fact that finding a good Sbox is easier in this case. However, it requires to find a way to include it in a construction that makes the cipher invertible, and also to take the resulting differential properties into account. Therefore, this non-bijective Sbox is used within the Feistel construction. But, as noticed by the designers, the use of a basic Sbox layer as an inner function would make the cipher vulnerable to a quite simple but very efficient differential attack with only one active Sbox every 2 rounds.

To thwart this sort of attacks, PICARO designers choose to use an expansion and a compression function that have good diffusion, more precisely that ensure that a minimum of 7 Sboxes are active in each active round. This is achieved by using linear operations deduced from linear codes with minimum distance 7. The expansion G takes as input a 64-bit word and outputs an extended word of 112 bits. The corresponding matrix is depicted below (its entries are elements in $GF(2^8)$ defined as $GF(2)[X]/(1+X^2+X^3+X^4+X^8)$). Then, the inner state must be compressed back at the end of the round function. To this end, another linear operation is used, defined by the matrix $H = G^T$, which converts 112-bit words into 64-bit ones.

$$G = \begin{pmatrix}
01 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01 & 01 & 0A & 01 & 09 & 0C \\
00 & 01 & 00 & 00 & 00 & 00 & 00 & 00 & 05 & 01 & 01 & 0A & 01 & 09 \\
00 & 00 & 01 & 00 & 00 & 00 & 00 & 00 & 06 & 05 & 01 & 01 & 0A & 01 \\
00 & 00 & 00 & 01 & 00 & 00 & 00 & 00 & 0C & 06 & 05 & 01 & 01 & 0A \\
00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 & 09 & 0C & 06 & 05 & 01 & 01 \\
00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 & 01 & 09 & 0C & 06 & 05 & 01 \\
00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 & 0A & 01 & 09 & 0C & 06 & 05 \\
00 & 00 & 00 & 00 & 00 & 00 & 00 & 01 & 01 & 0A & 01 & 09 & 0C & 06
\end{pmatrix}$$

To sum up, the round function is made of four operations, as depicted on Fig. 1: first, the 64-bit left part is extended by the expansion function defined by G, then the round-key is added. This is followed by the Sbox application (14 parallel applications of S) and finally the state is compressed back to 64 bits, and is XORed to the right part of the internal state.

PICARO encryption routine is made of 12 iterations of this round function.

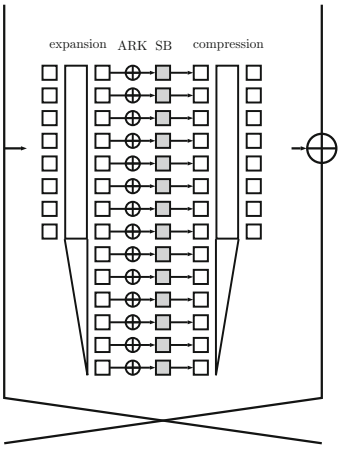

Fig. 1. One round of PICARO block cipher.

2.2 Key-Schedule

The previous algorithm requires twelve 112-bit subkeys that are derived from the 128-bit master key K. The designers wanted a simple and efficient key-schedule, together with resistance to the two main attacks exploiting the key-schedule, namely related-key attacks [1] and slide attacks [3]. In addition to that, Piret *et al.* looked for round-keys that could be computed on the fly, i.e. for which the computation of the subkey of round i can be done from the knowledge of the subkey at round $(i-1)$ (or from the knowledge of the subkey at round $(i+1)$, in decryption mode).

Their analysis has resulted in a linear key-schedule composed of rotations, bitwise additions and truncations. Namely, K denotes the 128-bit master key and $K^{(1)}, K^{(2)}, K^{(3)}$and $K^{(4)}$ are the four 32-bit chunks composing K: $K = (K^{(1)}, K^{(2)}, K^{(3)}, K^{(4)})$. Let $T : (GF(2)^{32})^4 \rightarrow (GF(2)^{32})^4$ be the linear transformation defined as follows:

$$\begin{pmatrix} T(K)^{(1)} \\ T(K)^{(2)} \\ T(K)^{(3)} \\ T(K)^{(4)} \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix} \times \begin{pmatrix} K^{(1)} \\ K^{(2)} \\ K^{(3)} \\ K^{(4)} \end{pmatrix}$$

Then, to compute the subkeys from the master key K, we first compute the extended key $(\kappa^1, \kappa^2, \cdots \kappa^{12})$ (where each κ^i is 128-bit long) with the following formulas:

$$\begin{cases} \kappa^1 = K \\ \kappa^i = T(K) \ggg \Theta(i) & \text{for } i = 2, 4, 6, 8, 10, 12 \\ \kappa^i = K \ggg \Theta(i) & \text{for } i = 3, 5, 7, 9, 11 \end{cases}$$

where $\Theta(i)$ is:

i	2	3	4	5	6	7	8	9	10	11	12
$\Theta(i)$	1	16	17	32	33	85	86	101	102	117	118

and then we obtain the round keys k^i by extracting the first 112 bits from κ^i. We denote by $skw = (k^1||k^2||...k^{12})$ the 'subkey word' made of the concatenation of the k^i.

The fact that T is an involution allows to easily deduce the 'on-the-fly' expressions, which are:

$$\begin{cases} \kappa^1 = K \\ \kappa^i = T(\kappa^{i-1}) \ggg \theta(i) \qquad \text{for } i = 2, \cdots, 12 \end{cases}$$

where θ is defined by:

i	2	3	4	5	6	7	8	9	10	11	12
$\theta(i)$	1	15	1	15	1	52	1	15	1	15	1

Since we do not need it here, we refer to the design document [8] for the formulas of the round-key derivation in decryption mode.

3 Definitions and Notation

In most block cipher constructions, the Sboxes are bijective, implying that in a differential attack a non-zero difference at the input of an Sbox is equivalent to a non-zero difference at its output. Here, the situation is different since the Sboxes are not bijective. Therefore, in order to avoid any ambiguity, we give a more precise definition of the notion of *active Sbox*:

Definition 1 (Active Sbox). *An active Sbox is an Sbox with a nonzero input difference (and a possibly zero output difference).*

Definition 2 (Data, Time and Memory Complexities).

- *The data complexity is defined as the number of plaintext/ciphertext pairs necessary to conduct the attack;*
- *The time complexity, which is expressed as a number of full 12-round encryptions, incorporates all the operations performed by the attacker to recover the key, and includes the encryptions needed to compute the necessary data;*
- *The memory complexity, which is expressed as a number of* 128*-bit blocks, measures the memory needed during the attack.*

Definition 3 (nR-attack [2]). *A nR-attack is a differential attack that covers R rounds with a differential characteristic and attacks $R+n$ rounds in total. We extend this definition to the related-key setting and call the additional rounds the key-recovery rounds.*

4 Key-Schedule Analysis

In this section we focus on PICARO key-schedule. We first focus on keys under which a given plaintext leads to the same ciphertext and then extend our analysis to a related-key attack.

4.1 Keys Leading to Colliding Ciphertexts

In this section we are interested in sets of keys that with high probability encrypt a given plaintext into the same ciphertext after the 12 rounds of PICARO. Note that in the ideal case, if one plaintext is fixed and encrypted under different keys, assuming that the resulting function is random, a ciphertext collision is expected to occur with probability 2^{-128}.

In the case of PICARO, we can remark that the round structure enables us to cancel a key difference quite easily: indeed, since the key addition is immediately followed by the Sbox layer, composed of non-bijective Sboxes, a key difference can be immediately canceled by going through the Sbox, resulting in an internal-state collision at the input of the compression function.

Obtaining colliding ciphertexts becomes then possible if we can construct keys differing in as few positions as possible, in order to make the probability of the event "all the subkey differences are canceled by the Sboxes" higher than 2^{-128}. This means that we can cancel a maximum of s Sboxes, with s satisfying $2^{-7\times s} > 2^{-128}$, so we have to find keys such that the corresponding subkey words differ in at most 18 bytes.

To find such keys, we first remark that the key-schedule algorithm is linear over $GF(2)$. This implies that the words corresponding to the concatenation of the subkeys (skw) belong to a linear code C of dimension $k = 128$ and length $n = 112 \times 12 = 1344$. Our search then boils down to the search for codewords with a low Hamming weight. We first focus on the Hamming weights of the codewords over $GF(2)$ and we will then move to the Hamming weight over $GF(2^8)$, i.e., the number of non-zero bytes, which is the relevant parameter in our attack.

To determine if it is possible to obtain keys differing in less than 18 bytes, i.e., at the inputs of less than 18 Sboxes, a first idea is to compute the minimum distance of the binary code, hereafter denoted by d. A straightforward computation of d would be too complicated due to the large size of the code. Some algorithms for finding low-weight codewords, e.g [4], could be used to determine d. But we now show that it can be easily deduced from the structure of the code with very simple arguments. We first make some observations coming from the structure of the generator matrix depicted on Fig. 2.

- If we consider all possible master keys of weight 1, the minimum weight among all corresponding codewords is 20, implying that $d \leq 20$
- According to the key-schedule description, 6 subkeys, namely k^1, k^3, k^5, k^7, k^9 and k^{11} consist of a selection of bits from the master key K. Following this, if we consider a master key of weight 1, then the word made by the

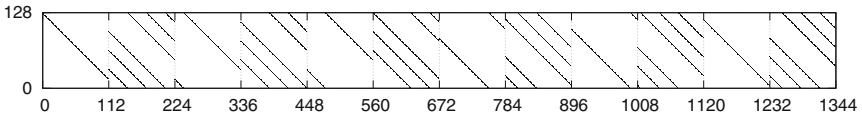

Fig. 2. Graphical representation of the generator matrix of linear code corresponding to the key-schedule.

concatenation of the subkeys *skw* contains at least 4 ones. Indeed, we have a 1 for sure in κ^1, κ^3, κ^5, κ^7, κ^9 and κ^{11} but after the truncation, a maximum of two of these ones may disappear. Accordingly, every time we add a one in the master key, the weights of the odd subkey words increase by at least 4.

All in one, those remarks show that, if they exist, the codewords of weight strictly less than 20 are obtained from master keys of weight at most 4. An exhaustive search over all these master keys shows that the minimum distance of the code is $d = 18$. The active positions of all master keys that reach this minimum are given in Table 1.

Table 1. Positions of the active bits in the round-keys that correspond to a minimum-weight subkey word (the bits entering the first Sbox are indexed from 0 to 7).

set	K	k^1	k^2	k^3	k^4	k^5	k^6	k^7	k^8	k^9	k^{10}	k^{11}	k^{12}
1	27, 123	27	28	11, 43	12, 44	27, 59	28, 60	80	81	0, 96	1, 97	16	17
2	28, 124	28	29	12, 44	13, 45	28, 60	29, 61	81	82	1, 97	2, 98	17	18
3	29, 125	29	30	13, 45	14, 46	29, 61	30, 62	82	83	2, 98	3, 99	18	19
4	30, 126	30	31	14, 46	15, 47	30, 62	31, 63	83	84	3, 99	4, 100	19	20
5	91, 123	91	92	11, 107	12, 108	27	28	48, 80	49, 81	64, 96	65, 97	80	81
6	92, 124	92	93	12, 108	13, 109	28	29	49, 81	50, 82	65, 97	66, 98	81	82
7	93, 125	93	94	13, 109	14, 110	29	30	50, 82	51, 83	66, 98	67, 99	82	83
8	94, 126	94	95	14, 110	15, 111	30	31	51, 83	52, 84	67, 99	68, 100	83	84

Note that these configurations might have been expected: the best scenarios are the ones for which the differences do not propagate a lot, so for keys differing in more than 2 bits we are looking for differences that cancel each other by the T map, i.e. that are at the same relative position in the 32-bit chunks of the master key. Moreover, the best configurations correspond to differences that end up as often as possible in the truncated part of the subkeys.

We note that the previous analysis determines the subkey words having a low Hamming weight counted in bits, while the relevant quantity is the number of nonzero bytes since it corresponds to the number of active Sboxes. However, the structure of the previous result indicates that the words with minimal weight over $GF(2^8)$ are derived from master keys having at most two nonzero bytes. An exhaustive search over this set enables us to determine a set W of 30 minimum-weight codewords over $GF(2^8)$, corresponding to the $(2^4 - 1)$ non-zero linear

combinations of the first four rows in Table 1 and to the $(2^4 - 1)$ non-zero linear combinations of the last four rows. This analysis exhibits the following bias in PICARO:

The probability that any given plaintext is encrypted to the same ciphertext under two keys whose difference belongs to the set W is $(2^{-7})^{18} = 2^{-126}$.

5 Related-Key Attack on the Full-Round PICARO

We have shown in the previous section that we can ensure that the differences introduced by the two keys are integrally canceled with probability greater than 2^{-128}. In this section, we show how to use this property in order to build a distinguisher and recover the encryption key with a related-key attack. We consider a scenario in which the attacker is given the right to ask for the encryption of plaintexts under the secret encryption key and under a second key which is related to the first one by a fixed difference.

In the following, we denote by a_i, $i = 1 \cdots 12$ the number of active Sboxes in each round (including the rounds not covered by the characteristic) and by $a_{i \to j}$ the number of active Sboxes in rounds i to j. We provide in Fig. 3 a concrete example providing the best trade-off.

5.1 A First 2R-Attack

We describe here a simple related-key attack on the full cipher and give some optimizations in the following sections. Our attack is based on a differential characteristic (in which all the differences are canceled by the Sboxes) that covers the first 10 rounds and ends with a 2-round key-recovery.

The two points that make the 2-round key-recovery holds are the following: first, the characteristic - that cancels all the differences - allows us to obtain a good filter on the ciphertexts. Second, a property of the compression function allows us to invert the compression step with a limited complexity.

Ciphertext Filter. For a plaintext and a pair of keys following the characteristic, the state entering round 11 is free of differences, and we know the value of the round-key differences entering the Sboxes at round 11, so we can determine a set of possible differences at the output of round 11. In the worst case scenario, there are 2^7 possible output differences for each active Sbox, so $2^{7 \times a_{11}}$ differences are possible out of 2^{64} for the corresponding half of the ciphertext. This remark implies that we can filter out some pairs that for sure do not follow the characteristic.

Property of the Compression Function. The following proposition will be extensively used in the attack. It shows that the knowledge of the output of the compression function and of any 6 bytes of the input uniquely determines all input bytes.

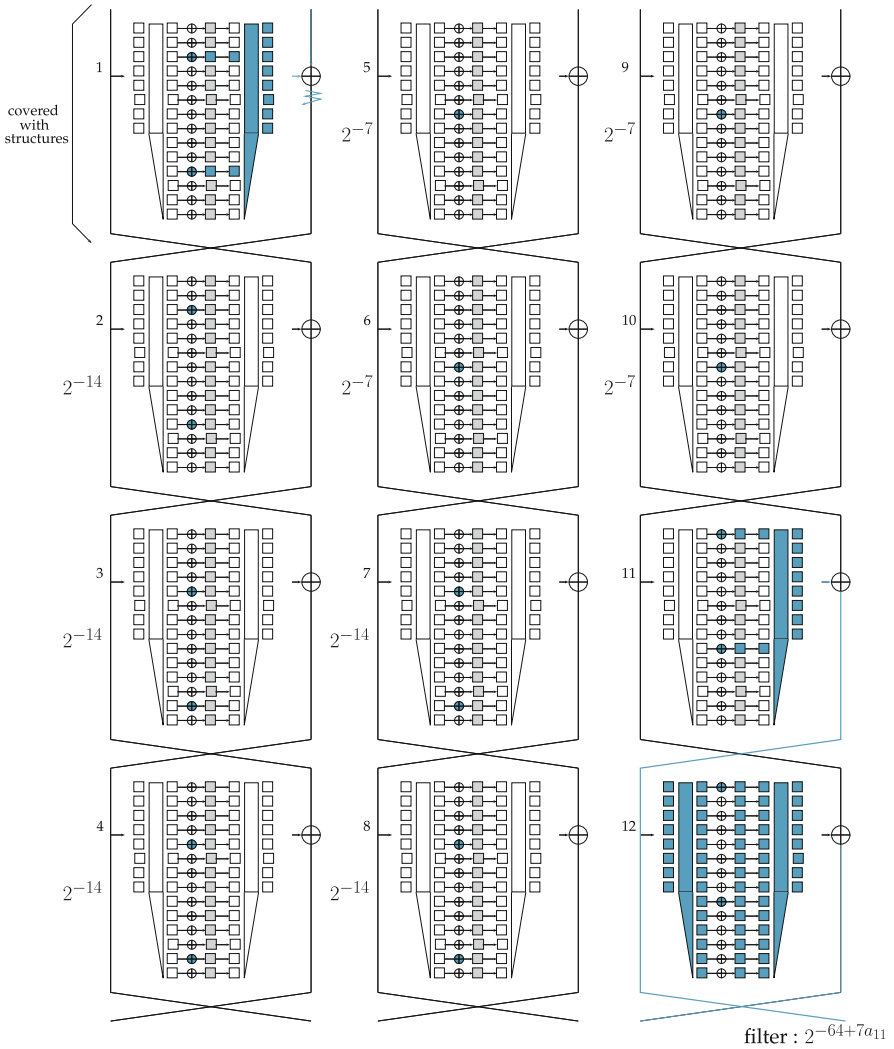

Fig. 3. Related-key attack on full-round PICARO. This figure represents the variant providing the best data complexity: 2^{99}, with a time complexity of $2^{107.4}$. The master key differences are positioned in bits 16 and 80.

Proposition 1. *Two distinct* 14*-byte words having the same image under the compression function coincide on at most five bytes.*

Proof. The compression function is defined by

$$\begin{aligned} \mathsf{Compr} : GF(2^8)^{14} &\to GF(2^8)^8 \\ x &\mapsto xH \end{aligned}$$

where H is a 14×8 matrix over $GF(2^8)$ defined in Sect. 2.1. Then, the set of all inputs which have the same image under Compr is an affine subspace of $GF(2^8)^{14}$ of dimension 6, which is a coset of the kernel

$$\mathcal{C} = \ker(\mathsf{Compr}) = \{x : xH = 0\}.$$

Therefore, any two elements x and x' having the same output under Comp are such that $(x + x')$ belongs to $\mathcal{C}$. Moreover, $\mathcal{C}$ can also be seen as the linear code of length 14 and dimension 6 over $GF(2^8)$ with parity-check matrix H^T. Since H^T has been chosen by the designers of PICARO as the generator matrix of an MDS code, it also corresponds to the parity-check matrix of an MDS code, implying that the minimum distance of $\mathcal{C}$ is $14 - 6 + 1 = 9$. Now, assume that there exist two distinct inputs x and x' which have the same output under Compr and which coincide on 6 input bytes. Then, $(x + x')$ is a non-zero element of $\mathcal{C}$ and it has at most $14 - 6 = 8$ nonzero bytes, which contradicts the fact that the minimum distance of $\mathcal{C}$ equals 9. □

Moreover, for any fixed output y of Compr and any set $I \subset \{1, \ldots, 14\}$ of 6 input positions, the unique 14-byte word equal to a given α on I can be determined by elementary algebra. We first observe that the matrix H defining the compression function is equal to

$$\begin{pmatrix} \mathsf{Id}_8 \\ Z^T \end{pmatrix}$$

where Z denotes the 8×6-right part of matrix G defined in Sect. 2.1. Then, the set of all words whose image equals y is equal to $\mathcal{C} + \widetilde{y}$ where $\widetilde{y}$ is the 14-byte word equal to y on the first 8 bytes and which vanishes on the other 6 bytes, and $\mathcal{C} = \ker(\mathsf{Compr})$ is the linear space spanned by the rows of matrix $M = (Z, \mathsf{Id}_6)$. Let M_I (resp. $M_{\overline{I}}$) denote the columns of M corresponding to I (resp. to $\overline{I} = \{1, \ldots, 14\} \backslash I$). The previous proposition shows that M_I is non-singular. Therefore, the unique element x in $\mathsf{Compr}^{-1}(y)$ which is equal to α on I corresponds to the sum of $\widetilde{y}$ and of the element in $\mathcal{C}$ which is equal to $\alpha + \widetilde{y}_{|I}$ on I. This implies that the value of x on $\overline{I}$ is equal to $\widetilde{y}_{|\overline{I}} + (\alpha + \widetilde{y}_{|I}) M_I^{-1} M_{\overline{I}}$.

Obtaining Suggestions for the Values of the Subkey at Round 12.

1. Choose a master-key difference Δ with a minimum number of active Sboxes in the first 10 rounds and ask for $2^{7 \times a_{1 \to 10}}$ triples (P_i, C_i, C'_i) generated from different plaintexts P_i, where C_i and C'_i are respectively the corresponding ciphertext under the secret master key K and the corresponding ciphertext under the key $K' = K \oplus \Delta$.
2. Filter out the triples by looking at the ciphertext differences: the number of remaining triples is $2^{7 \times a_{1 \to 10} - (64 - 7 \times a_{11})}$.
3. Since the internal state entering round 11 is free of difference, the left part of the ciphertext difference is equal to the difference at the output of the compression function at round 12. Given that, guess 6 bytes of the difference

entering the compression function and deduce the value of the full 112-bit difference entering the compression function at round 12 (this value is uniquely determined as shown by Proposition 1). There are 2^{48} possibilities for each triple.

4. Starting from the right half of the ciphertext difference, compute the expansion function and add the difference coming from the key difference to deduce the value of the difference entering the Sboxes at round 12.
5. With the help of the difference distribution table, check the difference transitions of the 14 Sboxes at round 12. If all the transitions are valid, compute the possible intermediate state values that permit these transitions. In order to estimate the average number of states returned by this procedure, we use that, for any Sbox, the product between the probability that a transition is valid and the average number of values following this transition is equal to 1. For PICARO Sbox S (note that the reasoning would be similar for any Sbox), we consider the number $\delta(a,b)$ of inputs x such that $S(x+a)+S(x)=b$, and the number τ of valid transitions, i.e., τ is the number of pairs (a,b) such that $\delta(a,b)>0$. The sum of all (non-zero) entries in the difference table of S equals $2^{2\times 8}$, i.e., the sum over all valid (a,b) of $\delta(a,b)$ equals $2^{2\times 8}$. It then follows that
$$\left(\frac{1}{\tau}\sum_{\text{valid }(a,b)}\delta(a,b)\right)\times\left(\frac{\tau}{2^{2\times 8}}\right)=1.$$
Therefore, for each Sbox transition considered in the attack, we obtain in average one intermediate state which satisfies this transition. So, we get a total of
$$2^{7\times a_{1\to 10}-(64-7\times a_{11})}\times 2^{48}=2^{7\times a_{1\to 11}-16}\text{ candidates.}$$
6. From the ciphertext value, deduce the intermediate state value before the key addition in round 12. From that, we obtain $2^{7\times a_{1\to 11}-16}$ candidate values for the subkey of round 12.

This algorithm requires $2^{7\times a_{1\to 10}+1}$ full encryptions for obtaining the initial triples, then $2^{7\times a_{1\to 11}-16+1}\times\frac{1}{12}$ encryptions (since only round 12 is computed) to obtain candidates for k^{12}, which leads to an overall time complexity corresponding to the cost of
$$2^{7\times a_{1\to 10}+1}+2^{7\times a_{1\to 11}-18.58}\text{ encryptions.}$$

5.2 Optimizations

Reducing the Initial Number of Encryptions. A potential bottleneck in the previous algorithm comes from the data complexity and then from the cost of the generation of the initial data. In this section, we detail a technique (close to the notion of structures which is commonly used in differential attacks) that reduces it by a factor of $2^{7\times a_1}$. The idea here is to let the first round-key difference spread freely and to cancel this difference by introducing a difference in another plaintext.

For each plaintext P_i encrypted under the master key K, we ask for the encryption of a set of plaintexts P_i^{set} encrypted under the second key. These plaintexts are made by adding to P_i all the possible differences coming from the subkey difference. To compute these differences, we exhaust all the $(2^{8\times a_1})$ possibilities for the difference at the output of the active Sboxes at round 1 and apply the (linear) compression function.

Since this set contains all the possible differences, one of them corresponds to the actual difference and will cancel the first round difference. Now, we extend that remark by asking the oracle for the encryption of the same set P_i^{set} under the master key. We then possesses $2^{8\times a_1+1}$ plaintexts and each plaintext in the first set is such that the difference introduced by the first round-key is canceled by exactly one plaintext in the second set. To sum up, we have exactly $2^{8\times a_1}$ pairs that lead to a zero difference at the end of round 1 from $2^{8\times a_1+1}$ encryptions only. The other parts in the algorithm remain the same. So to obtain one pair that leads to a zero difference at round 10 we need

$$2^{7\times a_{2\to 10}-8\times a_1+8\times a_1+1} = 2^{7\times a_{2\to 10}+1} \text{ encryptions.}$$

If we apply this technique to the previous algorithm, the number of initial encryptions is reduced from $2^{7\times a_{1\to 10}+1}$ to $2^{7\times a_{2\to 10}+1}$.

Speeding up the Master-Key Recovery. The previous algorithm returns candidates for the 112 bits of the subkey at round 12. From that point, we can naively perform an exhaustive search for the remaining 16 bits of the master key, which would lead to an attack with data complexity $2^{7\times a_{2\to 10}+1}$, time complexity $2^{7\times a_{2\to 10}+1} + 2^{7\times a_{1\to 11}-16} \times 2^{16}$ and with memory complexity $2^{8\times a_1+1}$ (necessary to store the initial 'structures').

We can further reduce the second term in the time complexity by using the previous rounds to filter out wrong candidates faster than with a trial encryption. Indeed, we can check if the candidate key gives the right Sbox transitions round after round, and discard a candidate as soon as the Sbox transition is wrong.

The optimal configuration would be the one for which we can deduce from the candidate value for k^{12} the key bits of k^1 and k^{11} that enter the active Sboxes at rounds 1 and 11. In such a case, the attacker can directly (i.e. without any additional key guess) check if the candidate value for k^{12} leads to the right Sbox transitions at rounds 1 and 11. This corresponds to a filter of about 2^{-7a_1} and $2^{-7a_{11}}$. The remaining operations (checking the Sbox transitions at the other rounds and guessing the remaining bits) are comparatively of negligible time complexity.

In case only a few key bits are known, the attacker can also reduce the time complexity by doing this gradual check. For the $2^{7a_{1\to 11}-16}$ pairs and candidate values for k^{12}, she can compute the expansion function at the round in which the largest number of information bits is known. At this round, she needs to guess the unknown bits in order to check the active Sbox transitions and then, for the right guesses, she can compute the entire round function in order to access another round. The full round function is then computed only for the guesses

Table 2. List of all master-key differences (of weight less than 4) minimizing $a_{2\to 10}$ ($a_{2\to 10} = 14$) with the smallest value for $a_{1\to 11}$ ($a_{1\to 11} = 18$). The last 2 columns indicate the number of key bytes (and bits) deduced from k^{12} at the positions corresponding to the active Sboxes at rounds 1 and 11.

Diff. positions		a_1	$a_{2\to 10}$	a_{11}	$a_{1\to 11}$	Known bytes (bits) in k_1	Known bytes (bits) in k_{11}
11	107	2	14	2	18	1 (14)	2 (16)
12	108	2	14	2	18	1 (14)	2 (16)
13	109	2	14	2	18	1 (14)	2 (16)
14	110	2	14	2	18	1 (14)	2 (16)
15	111	2	14	2	18	1 (14)	2 (16)
16	80	2	14	2	18	2 (16)	2 (16)
17	81	2	14	2	18	2 (16)	2 (16)
18	82	2	14	2	18	2 (16)	2 (16)
19	83	2	14	2	18	2 (16)	0 (14)
20	84	2	14	2	18	2 (16)	0 (14)
21	85	2	14	2	18	2 (16)	0 (14)
22	86	2	14	2	18	2 (16)	0 (14)
23	87	2	14	2	18	2 (16)	0 (14)
24	88	2	14	2	18	0 (4)	0 (14)
25	89	2	14	2	18	0 (4)	0 (14)
26	90	2	14	2	18	0 (4)	0 (14)
27	91	2	14	2	18	0 (4)	0 (0)
28	92	2	14	2	18	0 (4)	0 (0)
29	93	2	14	2	18	0 (4)	0 (0)
30	94	2	14	2	18	0 (4)	0 (0)
31	95	2	14	2	18	0 (4)	0 (0)
32	96	2	14	2	18	0 (0)	0 (0)
33	97	2	14	2	18	0 (0)	0 (0)
34	98	2	14	2	18	0 (0)	0 (0)
35	99	2	14	2	18	0 (0)	0 (2)
36	100	2	14	2	18	0 (0)	0 (2)
37	101	2	14	2	18	0 (0)	0 (2)
38	102	2	14	2	18	0 (0)	0 (2)
39	103	2	14	2	18	0 (0)	0 (2)
40	104	2	14	2	18	0 (12)	0 (2)
41	105	2	14	2	18	0 (12)	0 (2)

leading to the valid Sbox transitions. This occurs with probability 2^{-7} while at most 8 bits must be guessed. Therefore, we can consider that this step has negligible time complexity compared to $2^{7\times a_{1\to 11}-18.58}$.

Hence, the total data complexity is $2^{7\times a_{2\to 10}+1}$, the time complexity is reduced to $2^{7\times a_{2\to 10}+1}+2^{7\times a_{1\to 11}-18.58}$, and the memory is unchanged ($2^{8\times a_1+1}$).

Choosing $a_{2\to 10}$ and $a_{1\to 11}$. According to the previous analysis the bottleneck in the time complexity is:

$$2^{7\times a_{2\to 10}+1} + 2^{7\times a_{1\to 11}-18.58}.$$

It is then parametrized by $a_{2\to 10}$ (the amount of active Sboxes from round 2 to 10) and by $a_{1\to 11}$ (the amount of active Sboxes from round 1 to 11).

A simple search finds that the minimum for $a_{2\to 10}$ is 14 and that the minimum for $a_{1\to 11}$ is 17. Unfortunately, the two minima cannot be reached simultaneously, which gives raise to the following two possible options.

Variant 1: minimizing $a_{2\to 10}$. We consider here situations for which $a_{2\to 10} = 14$, the minimum for $a_{1\to 11}$ in these cases is 18 (see Table 2 for the difference positions that reach these minima and Fig. 3 for an example of a related-key attack using this variant).

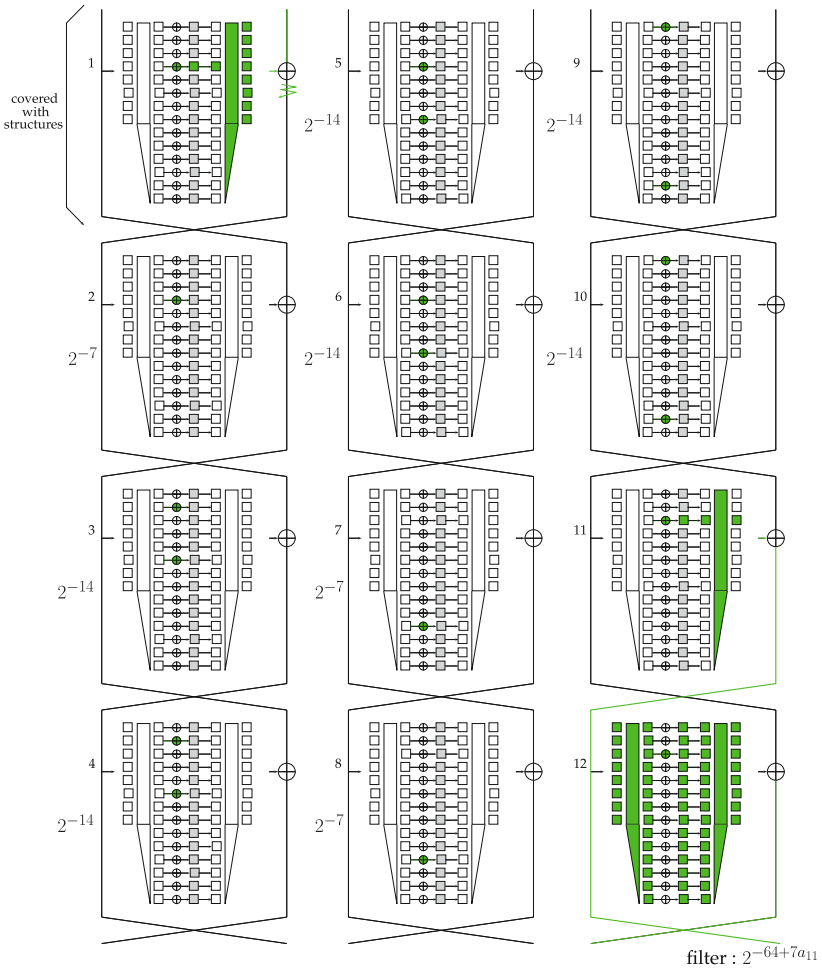

Fig. 4. Related-key attack on full-round PICARO . This figure represents the attack with the best time complexity: 2^{106}, with a data complexity of 2^{106}. The master-key differences are located at bits 27 and 123.

Table 3. List of all master-key differences (of weight less than 4) minimizing $a_{1\to 11}$ ($a_{1\to 11} = 17$) with the smallest value for $a_{2\to 10}$ ($a_{2\to 10} = 15$). The last 2 columns indicate the number of key bytes (and bits) deduced from k^{12} at the positions corresponding to the active Sboxes at rounds 1 and 11.

diff. positions	a_1	$a_{2\to 10}$	a_{11}	$a_{1\to 11}$	known bytes (bits) in k_1	known bytes (bits) in k_{11}
27 123	1	15	1	17	0(2)	0(0)
28 124	1	15	1	17	0(2)	0(0)
29 125	1	15	1	17	0(2)	0(0)
30 126	1	15	1	17	0(2)	0(0)
91 123	1	15	1	17	0(2)	0(0)
92 124	1	15	1	17	0(2)	0(0)
93 125	1	15	1	17	0(2)	0(0)
94 126	1	15	1	17	0(2)	0(0)

One of the advantages of this variant is that some of the options we can choose for the master-key difference allow us to speed up the search for the master-key. Indeed, two active Sbox transitions at round 1 and two other ones at round 11 can be checked without any additional key guess, as previously explained. However, this speed-up is imperceptible since it does not decrease the time complexity bottleneck. The final time complexity is

$$2^{7\times a_{2\to 10}+1} + 2^{7\times a_{1\to 11}-18.58} = 2^{7\times 14+1} + 2^{7\times 18-18.58} = 2^{99} + 2^{107.4} = 2^{107.4}.$$

The total data complexity is $2^{7\times a_{2\to 10}+1} = 2^{99}$, and the memory complexity is $2^{8\times a_1+1} = 2^{17}$.

Variant 2: minimizing $a_{1\to 11}$. If we choose to minimize $a_{1\to 11}$, the minimum value of $a_{2\to 10}$ is 15, and the time necessary to generate the data becomes $2^{7\times a_{2\to 10}+1} = 2^{106}$.

An exhaustive search among these configurations, as presented in Table 3, shows that none of them allows to do the direct sieving with k^1 and k^{11} that was possible in the previous variant. The attack obtained when the master-key difference is chosen at positions 27 and 123 is depicted in Fig. 4. The final data complexity (which is also the time complexity bottleneck) is $2^{7\times 15+1} = 2^{106}$, the time complexity is $2^{7\times a_{2\to 10}+1} + 2^{7\times a_{1\to 11}-18.58} = 2^{106} + 2^{100.4} = 2^{106}$ and the memory complexity is $2^{8\times a_1+1} = 2^9$.

6 Conclusion

In this paper we exhibit related-key attacks on the full-round block cipher PICARO. Our attacks exploit a weakness in the key-schedule as well as the non-bijectivity of the Sbox. We think that a stronger key-schedule with a better diffusion would help thwarting this type of attack.

Despite the fact that our related-key attacks do not represent a threat to the cipher in other more realistic scenarios, the authors aimed at providing related-key resistance, and we believe that such claim should be revised.

A PICARO Sbox

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	08	0c	03	06	06	04	05	06	05	04	0c	0c	04	03	05	03
10	0a	1f	29	3b	4b	55	62	7b	82	95	af	bf	c5	d9	e2	f9
20	01	2d	45	6a	8a	ac	cf	ea	9f	bc	dd	fd	1c	35	5f	75
30	0f	34	61	52	c2	fb	a3	92	13	2b	74	44	db	e1	b3	81
40	0f	44	81	c2	92	db	13	52	b3	fb	34	74	2b	61	a3	e1
50	0e	59	a4	f8	d8	87	7c	28	3c	67	99	c9	e7	b4	4c	14
60	02	63	ca	ad	1d	71	d7	bd	27	41	e3	83	31	5a	f7	9a
70	0f	74	e1	92	52	2b	b3	c2	a3	db	44	34	fb	81	13	61
80	02	83	9a	1d	bd	31	27	ad	f7	71	63	e3	41	ca	d7	5a
90	0e	99	b4	28	f8	67	4c	d8	7c	e7	c9	59	87	14	3c	a4
a0	0a	af	d9	7b	3b	95	e2	4b	62	c5	bf	1f	55	f9	82	29
b0	0a	bf	f9	4b	7b	c5	82	3b	e2	55	1f	af	95	29	62	d9
c0	0e	c9	14	d8	28	e7	3c	f8	4c	87	59	99	67	a4	7c	b4
d0	01	dd	35	ea	6a	bc	5f	8a	cf	1c	fd	2d	ac	75	9f	45
e0	02	e3	5a	bd	ad	41	f7	1d	d7	31	83	63	71	9a	27	ca
f0	01	fd	75	8a	ea	1c	9f	6a	5f	ac	2d	dd	bc	45	cf	35

References

1. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. J. Cryptology **7**(4), 229–246 (1994)
2. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology **4**(1), 3–72 (1991)
3. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
4. Canteaut, A., Chabaud, F.: A New algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. IEEE Trans. Inf. Theory **44**(1), 367–378 (1998)
5. Carlet, C.: Relating Three Nonlinearity Parameters of Vectorial Functions and Building APN Functions from Bent Functions. Des. Codes Crypt. **59**(1–3), 89–109 (2011)
6. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block Ciphers That Are Easier to Mask: How Far Can We Go? In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013)
7. Grosso, V., Leurent, G., Standaert, F.-X., Varici, K.: LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015)
8. Piret, G., Roche, T., Carlet, C.: PICARO – A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 311–328. Springer, Heidelberg (2012)
9. Piret, G., Roche, T., Carlet, C.: PICARO - A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance, extended version, IACR Cryptology ePrint Archive 2012, 358 (2012). http://eprint.iacr.org/2012/358
10. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)