A Note on the Empirical Evaluation of Security Margins against Algebraic Attacks (with Application to Low Cost-Ciphers LED and Piccolo) - Full Version -

Vincent Grosso[1⋆], Christina Boura[2,3], Benoît Gérard[1⋆⋆], François-Xavier Standaert[1⋆⋆⋆]

[1] UCL Crypto Group, place du Levant 2, 1348 Louvain-la-Neuve, Belgium.
[2] Equipe SECRET, INRIA Rocquencourt, B.P. 105, F-78153 Le Chesnay Cedex, France.
[3] Gemalto - 6, rue de la Verrerie - 92447 Meudon sur Seine, France.

Abstract. Algebraic attacks are an important class of cryptanalytic techniques. Yet, precisely estimating the security margins that a block cipher may provide against them is generally difficult, as sound theoretical tools are missing for this purpose. Therefore, most recent block cipher proposals combine different heuristic arguments in order to argue about their practical security against such attacks. In this paper, we discuss the relevance and correlation of these arguments, with a practical case-study based on the lightweight ciphers LED and Piccolo.

1 Introduction

The design of modern block ciphers usually comes with arguments of security against several cryptanalysis techniques. These arguments typically include the evaluation of statistical attacks such as linear and differential cryptanalysis [3, 19], and the investigation of structural properties leading to integral or slides attack [5, 18]. A number of well understood heuristic tools are available for this purpose. For example, the wide-trail strategy can be used to design block ciphers that are practically secure against statistical attacks [13]. One interesting feature of these heuristic tools is that they do not only provide security, but also allow the evaluation of (informal) bounds against certain categories of attacks. As a result, they can be used to compare different algorithms.

For other types of attacks though, and in particular for the algebraic cryptanalysis that we consider in this paper, security analyzes are hardly as systematic. This may sound counterintuitive, as algebraic complexity is known to be a central criteria for block cipher security since the seminal work of Shannon [24]. Yet, the discussion of algebraic attacks usually comes late (if at all) in block cipher specifications.

One possible reason of this situation is that the complexity of algebraic attacks is hard to evaluate [8, 9, 12, 20]. It is also not simple to interpret successful attacks against weakened versions of a block cipher. Yet, it remains that instances of (admittedly weak) block ciphers that are actually broken with algebraic attacks exist in the literature (e.g. the Keeloq and MiFare ciphers [10, 11]). Besides, it has also been shown that the algebraic complexity of a block cipher has a significant impact in the context of algebraic side-channel attacks [22, 23]. Hence, in view of the recent design of numerous lightweight block ciphers for constrained applications [15], it becomes increasingly interesting to understand the extent to which these new proposals with simplified structure and low implementation cost remain sufficiently robust against algebraic attacks.

⋆ PhD student supported by the ERC project 280141 - CRASH.
⋆⋆ Postdoctoral researcher supported by Walloon region MIPSs project.
⋆⋆⋆ Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

The security of a block cipher against such cryptanalyses can be argued with different types of metrics. For example, the complexities of the systems of equations representing different block ciphers have been compared in [4]. Another solution is to estimate the number of block cipher rounds needed to reach maximum algebraic degree [6]. More recently, cube testers have been proposed as another systematic alternative to construct algebraic distinguishers [1]. Eventually, the most direct approach is to investigate the complexity of solving a block cipher system of equations, e.g. with a SAT solver or Groebner basis tools [16]. However in this last case, directly solving the system of equations of a cipher should always be impossible (or would be the sign of a very weak design). This raises the question of which reduced versions of a cipher can be used to meaningfully argue about security margins against algebraic cryptanalysis.

In this paper, we consider the lack of comprehensive tools for the evaluation of algebraic attacks and discuss the relevance and limitations of a combined approach. Namely, we mix the estimation of informal criteria such as the system of equations size or the algebraic degree of a block cipher, with heuristic criteria such as the solving time of attacks against different versions of the target cipher. For this purpose, we consider attacks against full ciphers with variable guessing strategies, and attacks against reduced ciphers with fixed guessing strategy. Next, and taking the example of the block ciphers LED and Piccolo [17, 25], we discuss the extent to which these criteria lead to similar intuitions and can be used to estimate security margins against algebraic cryptanalysis. Doing so, we introduce a metric of "Equivalent Encryption Time" (EET) which essentially corresponds to the encryption speed that has to be reached for an exhaustive key search to be more efficient than an algebraic cryptanalysis. Let t be the median time needed for an algebraic attack to succeed, and n be the number of key bits to find, EET is defined as $t/2^n$. One interesting feature of this metric is that it allows comparing the security of different algorithms against algebraic attacks (i.e. their advantage over exhaustive search), independent of their encryption time. We use it to comment on the larger security margins of LED compared to Piccolo, and highlight observations regarding the impact of using a SAT solver in security evaluations.

2 Background

2.1 The LED block cipher

We consider the version of the cipher with 64-bit key that alternates 8 steps with key additions, each step being made of 4 rounds. It is illustrated in Figure 1, where p denotes a plaintext, c a ciphertext, k the cipher key and $\oplus$ the bitwise XOR. Each round is made of a constant addition, the parallel application of the PRESENT S-box and the AES-like operations ShiftRows and MixColumns. The cipher state and the key are represented as (4×4) matrices in $\mathbb{F}_{2^4}$, e.g.

$$s = \begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{pmatrix}.$$

Following, the different round operations are defined as follows.

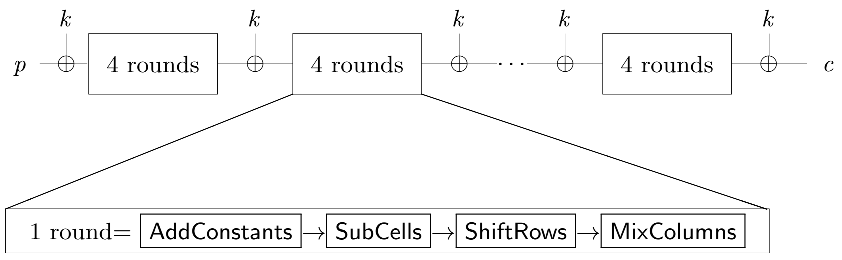

Fig. 1. The LED block cipher.

AddConstants. At each round, the state is XORed with a constant matrix. Constant values are produced by an LFSR with feedback polynomial $X^6 + X^5 + 1$, where the six register values are denoted as $\{rc_i\}_{i=0}^5$, and organized as follows:

$$\begin{pmatrix} 0 \; (0||rc_5||rc_4||rc_3) \; 0 \; 0 \\ 1 \; (0||rc_2||rc_1||rc_0) \; 0 \; 0 \\ 2 \; (0||rc_5||rc_4||rc_3) \; 0 \; 0 \\ 3 \; (0||rc_2||rc_1||rc_0) \; 0 \; 0 \end{pmatrix}.$$

SubCells applies the PRESENT S-box to the 16 cells of the state, defined as follows:

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

ShiftRows rotates the i^{th} row of the state by i positions to the left $(0 \leq i \leq 3)$.

MixColumns finally considers each nibble (4-bit) of the state matrix as an element of $\mathbb{F}_{2^4}$, with $X^4 + X + 1$ as polynomial for the field multiplication. It then multiplies this state with the following constant matrix:

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

2.2 The Piccolo block cipher

We consider the version of the cipher with 80-bit key that is made of 25 rounds. It is illustrated in Figure 2, where p denotes a plaintext, c a ciphertext, wk_i and rk_i the cipher subkeys and $\oplus$ the bitwise XOR. In this picture, F represents a non-linear function made of two SubCell and one Diffusion operations, and RP corresponds to a round permutation made of a wire crossing. More precisely, the 64-bit state of Piccolo is divided in four words of four nibbles. The F function operates on four nibbles and successively applies a layer of S-boxes, a Diffusion operation and the same layer of S-boxes again. The Piccolo S-box is exhaustively defined by the table:

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

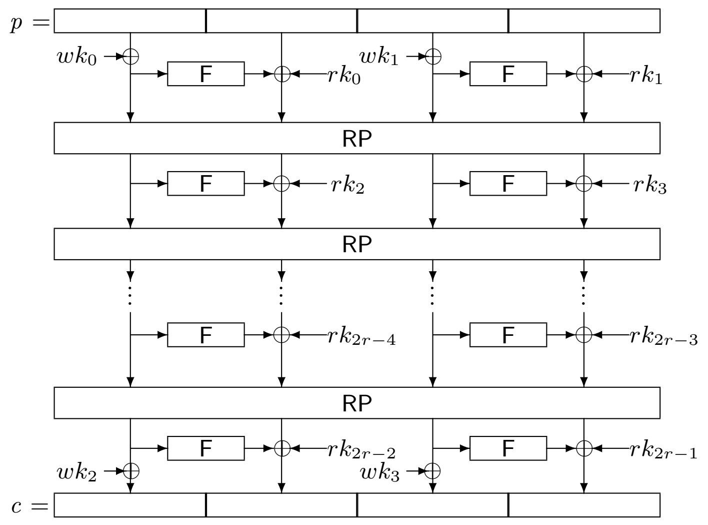

Fig. 2. The Piccolo block cipher.

Then, the Diffusion operation considers the nibbles as elements in $\mathbb{F}_2[X]/(X^4 + X + 1)$ and multiplies the cipher state with the following constant matrix:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}.$$

Besides, the round permutation RP divides the 64-bit state into eight 8-bit data pieces and permutes them as follows:

$$\mathsf{RP} : (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) \mapsto (x_2, x_7, x_4, x_1, x_6, x_3, x_0, x_5).$$

Finally, the key scheduling takes an 80-bit master key k as input and outputs the whitening and the round keys. The first ones are directly defined as:

$$wk_0 \leftarrow k[0 \ldots 7]||k[24 \ldots 31], \qquad wk_1 \leftarrow k[8 \ldots 23],$$
$$wk_2 \leftarrow k[64 \ldots 71]||k[56 \ldots 63], \qquad wk_3 \leftarrow k[48 \ldots 55]||k[72 \ldots 80],$$

where $k[i \ldots j]$ denotes the key bits with indexes between i and j. Next, the round keys are defined in function of the round constants as follows:

$$(rk_{2i}||rk_{2i+1}) \leftarrow (con_{2i}||con_{2i+1}) \oplus \begin{cases} k[32 \ldots 63] & \text{if } i \bmod 5 = 0 \text{ or } 2, \\ k[0 \ldots 31] & \text{if } i \bmod 5 = 1 \text{ or } 4, \\ (k[64 \ldots 79]||k[64 \ldots 79]) & \text{otherwise.} \end{cases}$$

$$(con_{2i}||con_{2i+1}) \leftarrow (c_{i+1}||c_0||c_{i+1}||00||(c_{i+1}||c_0||c_{i+1}) \oplus \{\texttt{0x0F1E2D3C}\},$$

where c_i is the 5-bit representation of the current round index i.

2.3 Algebraic attacks

Algebraic attacks have been proposed by Courtois and Pieprzyk in [12] and usually include two distinct steps. In the first place, the adversary describes his target cipher as a system of quadratic, cubic, ... equations. In general, the most challenging part of this step is to describe the non-linear S-boxes. Solutions for this purpose have been described by Biryukov and De Cannière in [4]. Tools to convert sparse systems of low-degree equations into a satisfiability problem have been proposed in [2]. Next, the second step of the attack is to try solving the system. The problem of solving systems of quadratic equations is NP-complete and there exist no efficient way to solve it in general. Yet, different heuristics exist that may be effective against certain instances of problems. Typical examples include the reduction to a Gröbner basis as proposed in [8], or a satisfiability problem as used in [11]. We focus on this second solution in the following sections. Beforehand, it is interesting to recall that the way to turn a cryptosystem into equations is not unique, and the representation of the problem may significantly influence the efficiency of the resulting attack (with both tools).

2.4 Algebraic degree of a Boolean function

From a mathematical point of view, a cipher E is a vectorial Boolean function of dimension n having m variables:

$$\begin{array}{rccl} \mathsf{E}: & (X, K) & \mapsto & Y = \mathsf{E}(X, K), \\ & \mathbb{F}_2^m & \to & \mathbb{F}_2^n. \end{array}$$

That is, each of the n coordinates is a Boolean function with m variables that should not be distinguishable from a randomly generated Boolean function. In general, any non-random behavior of any combination of coordinates can be the sign of a weakness and may be exploited in an attack (e.g. linear cryptanalysis [19], etc.). In the case of algebraic attacks, the degree of its Boolean functions is a good indicator for the strength of a block cipher. It is defined as follows.

Definition 1. *Algebraic degree. Let g be a Boolean function from $\mathbb{F}_2^m$ into $\mathbb{F}_2$. Such a Boolean function can be represented using its algebraic normal form (ANF):*

$$g(x_1, \ldots, x_m) = \sum_{(u_1,\ldots,u_m)\in\mathbb{F}_2^m} a_{(u_1,\ldots,u_m)} \prod_{i=1}^{m} x_i^{u_i}.$$

Such representation is unique and allows a simple definition of the degree of g:

$$\deg(g) \triangleq \max_{(u_1,\ldots,u_m)\in\mathbb{F}_2^m} \left\{ u = \sum_{i=1}^{m} u_i, a_{(u_1,\ldots,u_m)} \neq 0 \right\}.$$

In the case of a vector Boolean function $G = (g_1, \ldots, g_n)$, the degree is defined as the maximum degree of its coordinates:

$$\deg(G) \triangleq \max_{1\leq i\leq n} \deg(g_i).$$

In general, the state size of recent ciphers makes the explicit computation of the ANF for any coordinate of the cipher an intractable problem (a similar comment holds for hash functions). This situation motivated the derivation of bounds to estimate the algebraic degree as part of the security evaluation of a cryptographic primitive.

Taking the example of LED and Piccolo, a natural question is to determine how the degree of these ciphers evolves with the number of round iterations[1]. The most obvious way to do this, is to use what is generally called the *trivial bound*. It consists in bounding the degree of a round permutation of degree d after r iterations by d^r. This trivial bound usually gives good results when the number of rounds is relatively small, but fails as this number increases. For this purpose, a better estimation is needed.

Let us now denote the round functions of LED and Piccolo as $F = L' \circ S \circ L$, with S the non-linear S-box layer and L a linear (over $\mathbb{F}_2^n$) diffusion layer (i.e. two consecutive S-box layers will be separated by the linear function $L \circ L'$). Due to the mixing effect of the linear layer, the following quantity will be of interest for "chaining" rounds.

Definition 2. *Let $G = (g_1, \ldots, g_n)$ be a vector Boolean function of dimension n. Then we denote by $\delta_k(G)$ the maximal degree of the product of at most k coordinates of G:*

$$\delta_k(G) \triangleq \max_{\substack{I \subset \{1,\ldots,n\} \\ \#I \leq k}} \deg\left(\prod_{i \in I} g_i\right).$$

We now present a former result from [7]: it aims to provides a better approximation for the algebraic degree of a block cipher than the trivial bound.

Theorem 1. *If a vector Boolean function S from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ consists in the concatenation of smaller permutations from $\mathbb{F}_2^{n'}$ to $\mathbb{F}_2^{n'}$, then for any vector Boolean function G, the degree of $G \circ S$ is upper-bounded by:*

$$\deg(G \circ S) \leq n - \frac{n - \deg(G)}{\gamma(S)},$$

where $\gamma(S) \triangleq \max_{1 \leq i \leq n'-1} \dfrac{n' - i}{n' - \delta_i(S)}$.

Corollary 1. *Let F_r be the vector Boolean function corresponding to r iterations of a cipher round-function F as defined earlier. Then, its degree can be upper-bounded according to the degree of $r-1$ iterations of this function and the value of $\gamma(S)$:*

$$\deg(F_r) \leq n - \frac{n - \deg(F_{r-1})}{\gamma(S)}.$$

3 Estimating the algebraic degree of LED and Piccolo

In this section, we estimate the algebraic degree of LED and Piccolo. The goal of this estimation is to investigate possible links between the algebraic degree and the resistance of these block ciphers against the SAT-based algebraic attacks in Section 4.

[1] Since for both ciphers, the key addition corresponds to a bitwise XOR with a constant, the algebraic degree of the cipher can be computed independent of the key scheduling part.

3.1 On the algebraic degree of LED

The non-linear layer of LED consists in the parallel application of the PRESENT S-box. This S-box is a 4-bit permutation with algebraic degree 3. Thus, the degree of one round is also 3. We use Theorem 1 to derive bounds on the degrees for more rounds. Let us recall this bound for this particular instance where $\gamma(S) = 3$:

$$\deg(F_r) \leq 64 - \frac{64 - \deg(F_{r-1})}{3}.$$

It directly gives the results in Table 1. As expected, it is not tight for small degrees.

Table 1. Bounds on the algebraic degree of r-round LED.

Numbers of rounds	Trivial bound	Theorem 1
1	**3**	-
2	**9**	45
3	**27**	47
4	63	**51**
5	63	**59**
6	63	**62**
7	**63**	**63**

3.2 On the algebraic degree of Piccolo

The only source of non-linearity of Piccolo is the 16-bit-to-16-bit function F that is composed of two identical S-box layers separated by a matrix multiplication. As for LED, we start by investigating the algebraic degree for this function and then discuss its extension to the full cipher using Theorem 1.

Degree of F coordinates. Let us denote the input/output bits of the S-box as (x_0, x_1, x_2, x_3) and (y_0, y_1, y_2, y_3), respectively. The ANFs of the S-box coordinates are:

$$\begin{aligned}
y_0 &= x_1 + x_0x_1 + x_2 + x_0x_2 + x_0x_1x_2 + x_3 + x_0x_3 + x_1x_3 + x_1x_2x_3,\\
y_1 &= 1 + x_0 + x_0x_1 + x_1x_2 + x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3,\\
y_2 &= 1 + x_1 + x_2 + x_1x_2 + x_3,\\
y_3 &= 1 + x_0 + x_2 + x_3 + x_2x_3.
\end{aligned}$$

Note that the degree of the S-box is 3 but only the first two coordinates have this maximal degree, the last two only have degree 2. That is, the algebraic degree only captures information about the most complex coordinate of a vector Boolean function. After the parallel application of 4 S-boxes, the resulting 16 output bits $(y_0, \ldots, y_{15})$ are sent to MixColumns. Let us denote by $(z_0, \ldots, z_{15})$ these output bits of MixColumns that can be expressed as functions of y variables as follows.

$$\begin{aligned}
z_0 &= y_3 + y_4 + y_7 + y_8 + y_{12}\\
z_1 &= y_0 + y_3 + y_4 + y_5 + y_7 + y_9 + y_{13}\\
z_2 &= y_1 + y_5 + y_6 + y_{10} + y_{14}\\
z_3 &= y_2 + y_6 + y_7 + y_{11} + y_{15}\\
z_4 &= y_0 + y_7 + y_8 + y_{11} + y_{12}\\
z_5 &= y_1 + y_4 + y_7 + y_8 + y_9 + y_{11} + y_{13}\\
z_6 &= y_2 + y_5 + y_9 + y_{10} + y_{14}\\
z_7 &= y_3 + y_6 + y_{10} + y_{11} + y_{15}\\
z_8 &= y_0 + y_4 + y_{11} + y_{12} + y_{15}\\
z_9 &= y_1 + y_5 + y_8 + y_{11} + y_{12} + y_{13} + y_{15}\\
z_{10} &= y_2 + y_6 + y_9 + y_{13} + y_{14}\\
z_{11} &= y_3 + y_7 + y_{10} + y_{14} + y_{15}\\
z_{12} &= y_0 + y_3 + y_4 + y_8 + y_{15}\\
z_{13} &= y_0 + y_1 + y_3 + y_5 + y_9 + y_{12} + y_{15}\\
z_{14} &= y_1 + y_2 + y_6 + y_{10} + y_{13}\\
z_{15} &= y_2 + y_3 + y_7 + y_{11} + y_{14}
\end{aligned}$$

Combining these equations, we can express the ANFs of the 16 coordinates of F. Due to space limitation we cannot explicitly provide the equations. The degree $d_b(\mathsf{F})$ obtained for the b-th coordinate of F is:

$$d_b(\mathsf{F}) = \begin{cases} 9 & \text{if } b = 0 \mod 4,\\ 8 & \text{if } b = 1 \mod 4,\\ 6 & \text{if } b = 2 \mod 4,\\ 5 & \text{if } b = 3 \mod 4. \end{cases} \tag{1}$$

Then, we can use Theorem 1 to bound the degree of two or more Piccolo rounds. Nevertheless, the irregular distribution of the degrees among the coordinates suggests that such bounds will not be tight (as will be confirmed in Table 2). In the following, we derive bounds for 2 rounds based on the values of $\delta_k(\mathsf{F})$ that we have computed as:

$$\begin{aligned}
\delta_1(\mathsf{F}) &= 9,\\
\delta_2(\mathsf{F}) &= 12,\\
\delta_3(\mathsf{F}) &= 13,\\
\delta_k(\mathsf{F}) &= 14, \quad (k = 4, 5, 6),\\
\delta_k(\mathsf{F}) &= 15, \quad (k = 7, \ldots, 15).
\end{aligned}$$

Bound on the degree of 2 rounds of Piccolo. First let us precise the particular structure of the inputs of the 2-round F functions. The 16-bit state is divided into two bytes, each one being the XOR between a byte of the plaintext and a byte from the output of a previous F function. Given the degrees $d_b(\mathsf{F})$, we can bound the degree of the second-round F function outputs by considering all possibles distributions over the 2 aforementioned bytes. More precisely, let us consider the degree $d_b(F \circ \mathsf{F})$ for a given coordinate b of a second-round F function. Then, the highest degree monomial of its ANF is the product of $d_b(\mathsf{F})$ input bits of the second-round F function considered. These input bits are distributed among the two aforementioned bytes: d_1 belongs to the first and d_2 to the second byte ($d_1 + d_2 = d_b(\mathsf{F})$). Let us recall that the d_1 (resp. d_2) bits correspond to the addition between one output bit of a first-round F function and a plaintext bit. Hence, the degree of the product of those d_1 (resp. d_2) bits can be upper-bounded by $\max_{0 \le i \le d_1} i + \delta_{d_1 - i}(\mathsf{F})$. As a result, we obtain the following bound on the degree of a coordinate of a second-round F function.

$$d_b(F \circ \mathsf{F}) \le \max_{\substack{0 \le i \le d_1 \\ 0 \le j \le d_2 \\ d_1 + d_2 = d_b(\mathsf{F})}} i + j + \delta_{d_1 - i}(\mathsf{F}) + \delta_{d_2 - j}(\mathsf{F}). \quad (2)$$

We now provide the maximum values for the algebraic degree of the different output bits after two rounds of Piccolo, and the corresponding decompositions (d_1, d_2):

b = 0 mod 4 : Maximum 29 obtained for $(d_1, d_2) = (4, 5)$ (or $(3, 6)$ or $(2, 7)$).
b = 1 mod 4 : Maximum 28 obtained for $(d_1, d_2) = (4, 4)$ (or $(3, 5)$ or $(2, 6)$).
b = 2 mod 4 : Maximum 26 obtained for $(d_1, d_2) = (3, 3)$ (or $(2, 4)$).
b = 3 mod 4 : Maximum 25 obtained for $(d_1, d_2) = (2, 3)$.

Bounds on the degree of reduced-round Piccolo. We finally derive bounds on the reduced-round ciphers from the bounds obtained on one and two rounds (using Theorem 1). Table 2 summarizes the results. In the first column, the trivial bound $\deg(F_r) = \deg(F)^r$ is indicated. Then in the second column, the bound obtained using Theorem 1 can be found. In addition, the last column contains a bound on the algebraic degree of reduced-round versions of the cipher obtained using Theorem 1 and the tighter bound on the 2-round version derived in the previous paragraph.

Table 2. Bounds on the algebraic degree of r-round Piccolo.

Numbers of rounds	Trivial bound	Theorem 1	Theorem 1 + (2)
1	**9**	-	-
2	63	47	**29**
3	63	60	**52**
4	63	63	**62**
5	**63**	**63**	**63**

Let us notice that the first non-trivial bound is far from being tight according to the gap we can observe with entries in the last column. This suggests that the algebraic degree of Piccolo may be harder to estimate than the one of LED. It is likely that even the last column does not reflect the actual behavior of this algebraic degree.

Besides, and by contrast with the LED case, these estimations show that the algebraic strength of the Piccolo round-function output bits are not uniform. Hence, the relevance of the algebraic degree of the full function (that is the maximum degree of its coordinates) may be decreased in this case. This fact is naturally amplified by the Feistel structure of Piccolo, as at most half of its output bits might have reached degree 63 after 5 rounds. The other half of the state is composed of bits having at most degree 62 (corresponding to the 4-round outputs). Hence, they should pass through the F function at least once more to expect reaching the maximum degree.

4 Experimenting SAT-based algebraic attacks

Our algebraic cryptanalysis experiments are based on a SAT solver. We used the MiniSat v1.14 SAT solver [14] that is an open-source tool rewarded in different SAT competitions. Exploiting it requires describing the target cipher as a CNF, i.e. a conjunction of disjunction of variables. For this purpose, the straightforward strategy would be to express every ciphertext bit directly in function of the plaintext variables. However, this implies the apparition of numerous high degree monomials that are hardly managed by the solver. To overcome this limitation, the usual approach is to introduce intermediate literals in the cipher description (details are given next).

Since recovering the full cipher keys without additional information than a plaintext/ciphertext pair is (hopefully) difficult, our experiments were performed giving some key bit values as extra information to the solver. Depending on the experiments, this number of bits provided may differ. We will refer as *number of unknown key bits* the remaining number of key bit variables in the system after providing the extra information. Note that we chose to fix the first key bits of the master keys.

4.1 On the size of the CNF representation

In this section we focus on the complexity of the representation of both ciphers. Since the representation may strongly influence the resolution time, we tried to build systems having similar structures. Hence, we used the same construction method for LED and Piccolo. It consists in obtaining a representation by adding intermediate literals before and after each S-box execution and after each bit-wise XOR operation.

More precisely for the 64-bit key full-version of LED, the CNF representation has been obtained by adding intermediate literals before and after both the key additions and the AddConstants operations, and after the SubCells operations. As a result, we obtained a system of approximately 70.000 equations in 12.000 variables with at most 12 literals per clause. For the 80-bit key full-version of Piccolo, the CNF representation has been obtained by adding intermediate literals before and after each S-box in the F-function, and after the bit-wise addition between the state and the round keys (more precisely after the round permutation, but this does not change the representation because the permutation is linear). Concerning the key scheduling, we added literals for all the sub-keys. As a result, we obtained a representation of the Piccolo cryptosystem with 6.000 variables used in 50.000 equations. There are at most 8 variables per clause. For both ciphers, the representation of the S-box has been obtained with the same method, and gives 64 equations of 5 literals each and has 8 literals.

We can see that the LED representation requires more equations than Piccolo (respectively 70.000 and 50.000), while the number of variables in the LED representation is twice the number of variables in Piccolo. Moreover, we notice that the literals are more connected in the LED representation than in the Piccolo representation. This may suggest that LED is more robust than Piccolo against algebraic attacks, at least when deriving representations in such a straightforward fashion.

4.2 Attacking the full-version with variable key-guess sizes

We now consider the evolution of the resolution time of the system as a function of the number of the key bits unknown to the solver. The target ciphers are the full-version of the algorithms proposed at CHES. The representation used are the one described in Section 4.1 and, as mentioned earlier, we have fixed the first key bit values. We aimed at comparing resolution times for a number of unknown key bits ranging from 4 to 18, which translates in providing 46 to 60-bit values (resp. 62 to 76 bits) to the solver for recovering the full key of LED (resp. Piccolo). In the case of LED cipher, we did not manage to obtain results when guessing less than 49 key bits, due to the prohibitive solving time of such experiments. The experimental results obtained are provided in Figure 3, where the curves represent the evolution of the median solving time as a function of the number of key bits unknown to the solver.

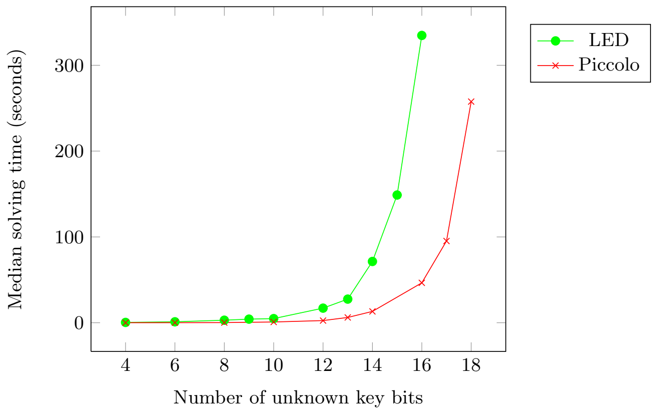

Fig. 3. Solving time vs number of unknown key bits for LED and Piccolo.

As expected, we can observe that the resolution time grows as an exponential function of the number of unknown bits for both ciphers. We also notice that the Piccolo curve is shifted by 2 bits on the x-axis compared to the LED curve. Hence, this metric again suggests that Piccolo could be slightly weaker than LED against algebraic cryptanalysis. Such result naturally has to be considered carefully, since the total number of key bits differs in LED and Piccolo.

Next, we translated these median solving times into EET. Intuitively, the EET represents the encryption speed that should be reached by an implementation of the block ciphers, for the exhaustive search to be more efficient than the SAT solver based cryptanalysis. Results are plotted in Figure 4. We first remark that for a given number of unknown key bits, the EET of different ciphers are proportional to the corresponding median solving times. Hence, comparing LED and Piccolo by using this metric will result in the same conclusions as in Figure 3. Nevertheless, this metric is of interest since it allows the observation of some phenomenons that are not straightforwardly visible when considering the median solving time in Figure 3.

First, and in both plots, two different parts can be observed: the EET initially decreases up to a certain point, where it then becomes stable. While having the same shape, both curves differ in the point where the EET becomes stable. The EET for

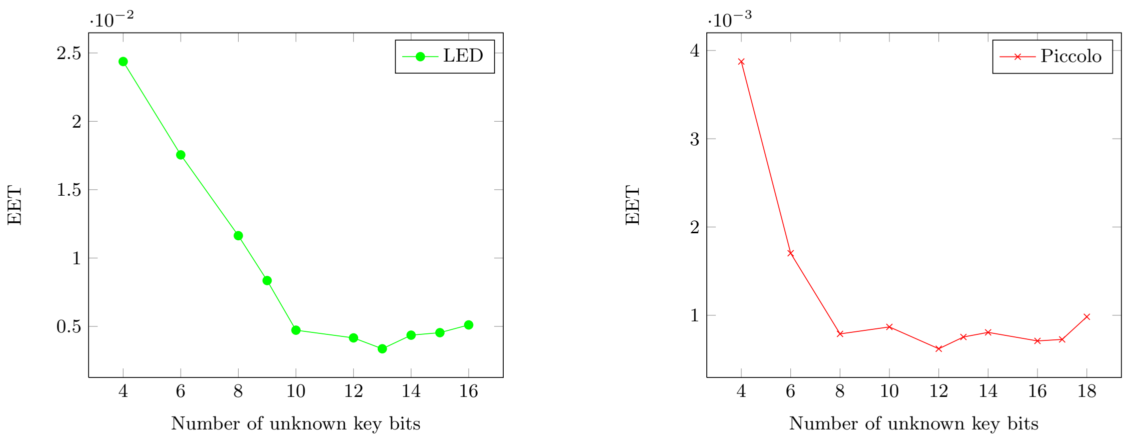

Fig. 4. EET vs number of unknown key bits for LED and Piccolo.

LED stops decreasing when more than 10 key bits are unknown (stabilizing around an EET equal to $0.5 \cdot 10^{-2}$), while for Piccolo, it stops decreasing when more than 8 key bits are unknown (stabilizing around an EET equal to $1 \cdot 10^{-3}$). This decreasing behavior is mainly due to the construction phase performed by the SAT solver prior to the resolution: as the number of unknown key bits increases, this construction step becomes negligible compared to the resolution time. Since the LED system is bigger than the one of Piccolo, it is natural that the stabilization happens later for LED.

Second, the values of EET obtained when enough key bits are unknown are significantly larger than the actual encryption time, even on a standard PC. This suggests that both ciphers have satisfying security margins against this type of attack. Nevertheless, the EET is larger for LED than for Piccolo (by an approximate factor 5).

4.3 Attacking reduced-round versions.

As a complement to the previous experiments, we now investigate the security of reduced-round versions of our target ciphers, for different number of unknown key bits. This requires a slight adaptation for the LED cipher. Indeed, in its standard version, LED performs a bitwise XOR with the key every four rounds (which makes it difficult to vary the number of rounds with such a small granularity). Hence, for this experiment, we choose to perform only two key-additions at the beginning and at the end of LED, regardless the number of rounds[2]. In this setting, we performed experiments for a number of unknown key bits ranging from 12 to 16 for LED, and from 16 to 20 for Piccolo. This choice has been made in order to obtain equivalent ranges of resolution times for both reduced-round ciphers. Figure 5 depicts the evolution of the solving time depending on the number of rounds we have fixed.

We observe that for both ciphers, the curves obtained have a very similar shape, namely a highly increasing part first, and a stabilized/slowly increasing second part (let us recall that the y-axis is log-scaled). This means that increasing the number of rounds beyond some limit does not provide significantly more security anymore regarding the

[2] The reason why we modified LED in such a way is detailed in Section 4.4.

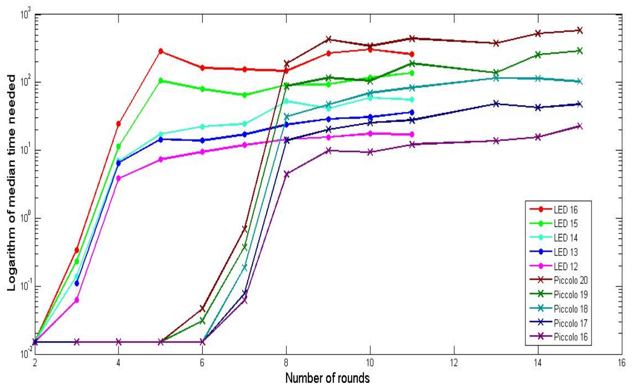

Fig. 5. Solving time for different numbers of unknown key bits and rounds (y-axis scale logarithmic).

solving time of the system. One possible interpretation of this fact is the following. The solving time of an algebraic attack primarily depends on the size of the system and the "complexity" of the equations it aims at solving (partially captured by the algebraic degree). As equations are getting more complex with the number of rounds, they reach their maximum "complexity" (and algebraic degree) at some point. From this point on, only the size of the system goes on increasing, hence explaining a slower increase of the solving time. We now discuss the links between the algebraic degree for the reduced-round versions of the ciphers, and the bends observed in Figure 5.

LED cipher. We observe that the fast increase in solving complexity seems to saturate after approximately 5 rounds in this case. In Section 3, we obtained bounds on the algebraic degree of reduced-round LED that were 62 for the 5-round version and 63 (the maximum degree) for the 6-round version. As a result, the position of the bends the figure seems to be reasonably correlated with the algebraic degree for the LED cipher. We assume that this correlation is enhanced thanks to the regularity of the distribution for the degrees over the output coordinates of the cipher.

Piccolo cipher. In this case, the fast increase in solving complexity extends up to approximately 8 rounds. By contrast, the bound obtained in Section 3 becomes close to the maximal degree after only 4 or 5 rounds. Hence, the link between the algebraic degree and the solving time is less direct for Piccolo. As previously mentioned, this may be due to two reasons. First, the bounds derived are upper bounds on the algebraic degree of the cipher. The significant improvements observed by using the additional information (2) to derive bounds for Piccolo suggest that these bounds are probably not as tight as the ones obtained for LED. Secondly, the degrees for the outputs of the F function show that the algebraic complexity of this cipher depends on the position of the output bit considered. However, the algebraic degree only takes into account the maximal degree (i.e. the strongest bit) and does not consider any other information on the distribution of the degrees (median or minimal degrees for instance). Moreover as we remark in Section 3, half of the bits need one more round to reach this degree. This suggests that the bend may correspond to the point where all output bits reach a high degree, and not to the point where the strongest output bit reaches a high degree.

4.4 On the heuristic impact of the key addition layers in LED

In this last experiment, we focused on a parasitic effect we observed on LED, that is linked to the heuristics used in the SAT solver. The point to emphasize is that depending on the number of key additions in the LED cipher, the solving time may significantly vary. First observe that the full-round version of LED has 32 rounds (traditionally divided in eight 4-round steps separated by key additions). In order to illustrate the parasitic effect, we used a modified version of LED with 32 rounds, but variable number of key additions. We additionally provided a 48-bit key information to the solver (hence 16 bits remain to be found) and plotted the median solving time as a function of the number of key additions (ranging from 2 to 32) in Figure 6. Interestingly, it clearly exhibits an decrease of the median solving time with the number of key additions.

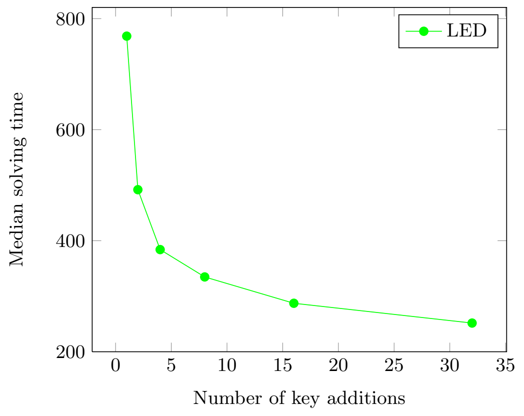

Fig. 6. Effects of the number of LED key additions on the solving time.

While this observation may appear surprising (as the number of key additions has no impact on the algebraic degree), it can be explained by the way the MiniSat solver chooses the order of variables to guess [26]. In summary, MiniSat uses a variant of the VSIDS algorithm in which each literal is being attributed a counter, initialized with the number of occurrences of this literal in the description. Then, literals having the highest counters are assigned first. When a conflict appears, the counters are multiplied by 0.95. Looking at the system of equations of a block cipher, starting with guesses on the internal literals can produce conflicts, as these internal literals are more connected to each others than the key literals. Unfortunately, in our representation of the original LED cipher, these internal literals have greater counters than the key literals. For example, the key variables appear twice in each bitwise XOR between the state and the key, while an internal literal between AddConstants and SubCells might appear up to 66 times. Hence, by repeating key additions, we actually increase the counters corresponding to the key literals and thus suggest to the solver to start guessing a key bit (which may lead to a more efficient solving, as witnessed by the results in Figure 6).

References

1. Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2009.

2. Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson. Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers. Cryptology ePrint Archive, Report 2007/024, 2007. `http://eprint.iacr.org/`.
3. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.
4. Alex Biryukov and Christophe De Cannière. Block Ciphers and Systems of Quadratic Equations. In Thomas Johansson, editor, *FSE*, volume 2887 of *LNCS*, pages 274–289. Springer, 2003.
5. Alex Biryukov and David Wagner. Slide Attacks. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.
6. Christina Boura and Anne Canteaut. On the Algebraic Degree of Iterated Permutations. Finite Fields and Applications - Fq10, Gent, Belgium, 2011.
7. Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 2011.
8. Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. Block Ciphers Sensitive to Gröbner Basis Attacks. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 313–331. Springer, 2006.
9. Carlos Cid and Gaëtan Leurent. An Analysis of the XSL Algorithm. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005.
10. Nicolas Courtois. The Dark Side of Security by Obscurity - and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime. In Eduardo Fernández-Medina, Manu Malek, and Javier Hernando, editors, *SECRYPT*, pages 331–338. INSTICC Press, 2009.
11. Nicolas Courtois, Gregory V. Bard, and David Wagner. Algebraic and Slide Attacks on KeeLoq. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 97–115. Springer, 2008.
12. Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.
13. Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.
14. Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.
15. Thomas Eisenbarth, Zheng Gong, Tim Guneysu, Stefan Heyse, Sebastiaan Indesteege, Stephanie Kerckhof, Francois Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, Francois-Xavier Standaert, and Loic van Oldeneel tot Oldenzeel. Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. ECRYPT Workshop on Lightweight Cryptography, pages 71-86, Louvain-la-Neuve, Belgium, 2011.
16. Jeremy Erickson, Jintai Ding, and Chris Christensen. Algebraic Cryptanalysis of SMS4: Gröbner Basis Attack and SAT Attack Compared. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2009.
17. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Preneel and Takagi [21], pages 326–341.
18. Lars R. Knudsen and David Wagner. Integral Cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.
19. Mitsuru Matsui. Linear Cryptoanalysis Method for DES Cipher. In Tor Helleseth, editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.
20. Sean Murphy and Matthew J. B. Robshaw. Essential Algebraic Structure within the AES. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.
21. Bart Preneel and Tsuyoshi Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
22. Mathieu Renauld and François-Xavier Standaert. Algebraic Side-Channel Attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.
23. Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.
24. Claude E. Shannon. Communication Theory of Secrecy Systems. Bell System Technical Journal, vol. 28(4), page 656-715, 1949.
25. Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In Preneel and Takagi [21], pages 342–357.
26. N. Sörensson and N. Eén. Minisat a sat solver with conflict-clause minimization. In *SAT*, 2004.