Cryptanalysis of AES and Camellia with Related S-boxes

Marco Macchetti

Kudelski Group
marco.macchetti@nagra.com

Abstract. Cryptanalysis mainly has public algorithms as target; however cryptanalytic effort has also been directed quite successfully to block ciphers that contain secret components, typically S-boxes. Known approaches can only attack reduced-round variants of the target algorithms, AES being a nice example. In this paper we present a novel cryptanalytic attack that can recover the specification of S-boxes from algorithms that resist to cryptanalysis, under the assumption that the attacker can work on a pair of such block ciphers that instantiate related S-boxes. These S-boxes satisfy the designer's requirements but are weakly diversified; the relationship between these unknown components is used in much the same way as relationship between secret keys is used in related-key attacks. This attack (called related S-box attack) can be used, under certain assumptions, to retrieve the content of the S-boxes in practical time. We apply our attack to two well known ciphers, AES and Camellia; these ciphers use 8-bit S-boxes but are structurally very different, and our attack adapts accordingly. This shows that most probably the same can be applied to other ciphers which can be customized to instantiate unknown 8-bit S-boxes.

1 Introduction

Block cipher design is a well developed research field; the AES contest has without doubt contributed to its growth. Today, not only we have a significant number of good algorithms, we also possess criteria that can be used to design ciphers that are robust against known cryptanalytic techniques, such as linear cryptanalysis [15], differential cryptanalysis [2], algebraic attacks [11].

Those algorithms that are fully public, and withstand all cryptanalytic attacks, are considered to be the best and therefore are used ubiquitously; this is, after all, the main motivation behind the AES contest (and the SHA-3 one). But in some cases there may be a need to keep at least part of an algorithm private. Although this is not commonly seen as good practice, being a contradiction of the famous (although perhaps overestimated) Kerckhoffs principle, it is not rarely done in practice as there may be a good justification.

Considering products such as RFIDs, smart-cards and conditional access tokens, adversaries may be able to compromise the security of part of the system with the ultimate goal of cloning the device (well-funded pirate organizations have the possibility and technical skills to pursue this goal). Basing the

A. Youssef, A. Nitaj, A.E. Hassanien (Eds.): AFRICACRYPT 2013, LNCS 7918, pp. 208–221, 2013.

cryptographic constructions on completely standard algorithms thus gives the adversary an advantage because the cloning procedure is easier; assuming the adversary gains complete control over the block cipher under attack, he can choose a key value and by observing few encryptions he will be able to say which standard algorithm is used. On the other hand, an unknown algorithm forces the attacker to fully reverse-engineer the device, a thing which is definitely more difficult, and costly, than a partially invasive attack. Of course, the algorithms must still be based on solid constructions, well-analyzed and characterized by proofs of security.

In many algorithms, the S-box is a natural candidate for customization, for several reasons. As an example, the Rijndael SPN structure can be easily customized by replacing the standard S-box with a randomly picked 8-bit permutation; the resulting cipher still maintains all the structural properties of AES while it forces the adversary to reverse engineer an implementation to be able to clone the circuit.

Even if the particular S-box used in Rijndael has optimal differential and linear characteristics [12], these parameters can actually be relaxed, since a large margin of security exists with regards to classical attacks in the design of the cipher. For instance, the expected maximum entry in the Differential Distribution Table (DDT) of a random 8-bit S-box is 16 [17], whereas the maximum DDT value of the AES S-box is 4. This means that the probability of differential trails over 4 rounds is increased from 2^{-150} to 2^{-100}, a value that anyway render differential attacks over the full cipher impossible. Regarding algebraic properties, a random 8-bit bijection is likely to show up no monomial characterization, even if the algebraic degree will not be maximal. We also note that the recent biclique attacks which have been shown to work on the full AES [8] and are the most successful attack to date, combine the biclique concept with the use of meet-in-the-middle structures, for which known differentials must be used. These differentials are not known by the attacker if the S-box is unknown.

Thus a randomly-generated S-box (e.g. by means of a true random number generator and application of the Knuth shuffle [14]) is expected to behave well enough. The number of choices is extremely large; taking into account all permutations on 8 bits, we have a customization space of about $\log_2(256!) \approx 1684$ bits. Even giving the adversary the possibility to completely control the key, he cannot recover the content of the secret S-box and use it to clone the device. This is because in the known-key scenario the probability of differential and linear characteristics for the AES is low enough for them to be useless [19].

Even if it is intuitive that some security is added if part of a block cipher specification is kept private, there is little available quantitative analysis of the subject in the literature. We have to say that here we are not focusing on implementation-based attacks, such as fault injection or side channel analysis; it is today known that these techniques can be used to reverse engineer block ciphers, such as in SCARE attacks [10][16][18][13] and in FIRE attacks [3]. The primary goal of this paper is rather to consider the components of a block cipher (such as unknown S-boxes) as another design dimension, and to introduce a new class of quite powerful related-cipher attacks (that we call related-S-box attacks).

Related-key attacks are today accepted as a way to expose weaknesses of a block cipher, and are based on the fact that the adversary may know the relationship between a pair (or a bigger set) of otherwise secret keys [5]. The concept of related ciphers has been analyzed in [22][20], where the existence of ciphers parameterized by variable number of rounds was exploited to determine the value of the key. The relationship between different modes of operation has also be considered in related cipher attacks [21].

We present here a novel type of attack which follows in these footsteps and exploits the existence of a strong relationship between different, but unknown, S-boxes to break the cipher[1]. This is, to our knowledge, the first cryptanalytic attack that can obtain the specification of S-boxes instantiated in block ciphers with the strength of the AES; and under certain assumptions, we do it in practical time.

2 The Related S-box Attack

2.1 Overview and Assumptions

Let us examine the case of AES instantiating an unknown S-box, but with usual key-schedule, round function structure and number of rounds; let us limit our analysis to the 128-bit key variant. Consider the following definition:

Definition 1. *Two S-boxes* $S_1, S_2 : F_{2^m} \rightarrow F_{2^n}$ *are said to be* δ*-related if*

$$S_1(x) = S_2(x) \quad \Leftrightarrow \quad x \notin \Delta$$
$$|\Delta| = \delta \quad , \quad 2 \leq \delta \leq 2^m$$

This definition may seem a bit simplistic, in the sense that it considers the similarity between two S-boxes only in terms of the number of entries on which they agree; this is precisely the characteristic which is used by our attack, and we believe it is the most generic and agnostic notion of similarity. Of course one may think about linearly equivalent S-boxes [4], or even more complex relationships. These cases are also interesting, but the class of attacks that could stem from them is much more limited in the number of rounds that can be attacked[2].

Let us consider two identically structured block ciphers which instance two δ-related S-boxes S_1 and S_2 according to the definition above; we will by analogy call them δ-related block ciphers. Note that in our definition, low values of δ identify pairs of very similar S-boxes, and thus this parameter measures

[1] By *breaking* here we mean that the full specification of the algorithm is retrieved, since the goal of the attacker in our scenario is to clone the device. We assume that all block cipher inputs (including the key) are under the attacker's control.

[2] Our relationship definition has the advantage of capturing the case where physical attacks on memories or logic could result in few entries to be interchanged; more in general, the S-box generation algorithms could also be attacked or poorly implemented and give strongly related S-boxes as result.

the degree of *relationship* between two S-boxes; however, recall that δ is also a measure of the number of entries that *differ* between S_1 and S_2.

The attack starts from the simple but somewhat surprising consideration that these two ciphers behave in the same way in a relatively large amount of cases, depending on the value of δ and on the size and number of S-boxes. In the general case, if we key two block ciphers with the same key k, and we encrypt the same plaintext p, the chance of obtaining the same ciphertext is equal to the probability that no S-box receives as input one of the values in the set Δ (no S-box is Δ-active, in our terminology). If the block cipher contains s S-boxes, this probability is equal to:

$$P(c_1 = c_2 | p_1 = p_2, k_1 = k_2) = \left(\frac{2^m - \delta}{2^m}\right)^s \tag{1}$$

If we look at the case of AES, we have $s = 200$ and $m = 8$; if $\delta = 2$ (the minimum value possible for bijective S-boxes) then the probability becomes:

$$P(c_1 = c_2 | p_1 = p_2, k_1 = k_2) = \left(\frac{256 - 2}{256}\right)^{200} \approx 0.20833 \tag{2}$$

so we expect that in about 1 case out of 5 we observe a collision on the ciphertext values; in this computation we assume that all S-box inputs are uncorrelated and uniformly distributed, which is obviously not true in practice, however this probability is easily confirmed with experiments.

This fact seems somewhat in contradiction with the belief that a cipher like the AES has good randomization properties and such events should intuitively have a very low probability. If we consider the value of 2^{-64} as threshold for the collision probability, we have that δ can reach the value of 50, i.e. S_1 and S_2 are different for slightly less than one fifth of the values.

By looking at another well known cipher that contains 8-bit S-boxes, Camellia [1], we note that S-box s_1 is directly instantiated in the round function and key-schedule, and also used to derive the other three S-boxes s_2, s_3, s_4 in a way to preserve the value of δ. Since for Camellia we have $s = 192$ and $m = 8$, the collision probability for the different values of δ are even larger than those of AES.

In our attack, we use this collision probability as a tool to obtain the specification of the unknown components, i.e. the complete content of the S-boxes. The attack scenario is the following: we assume that the attacker is able to submit encryption queries to two δ-related block ciphers. We assume that the attacker can re-key the two ciphers as he likes; his goal is to recover the specification of the unknown component (the two δ-related S-boxes S_1, S_2). The attack we present here works on the full AES and Camellia block ciphers and is shown to work in *practical* time for δ up to 16.

Since in the attack we are mainly interested in verifying assumptions on the first round of encryption, we use ciphertext *almost* collisions, i.e. pairs of ciphertexts which differ in 8 or less byte positions. For both AES and Camellia, this means that a difference has most likely been originated within the last round of

encryption, and this is sufficient for our goals. The probability of having such differentials springing from the first rounds should be around 2^{-64}, and therefore if the theoretical probability of collision is significantly greater than this, the approach will work well.

More precisely, by looking at a pair of ciphertexts obtained by encrypting the same plaintext with two δ-related AES ciphers we can make the following list of statements about the most probable explanation of the given observed difference pattern:

1. If the difference between the two ciphertexts is non null on only one byte position, then we most likely have a single Δ-active S-box in the final round of the cipher.
2. If the difference is a full row of the byte matrix, we most likely have a single Δ-active S-box in the key schedule computation of the last subkey (remember that the last round of AES has no MixColumn step).
3. If the difference pattern is (embedded by) a column of the byte matrix, the Δ-active S-box is in the round before the last.
4. If the difference is (embedded by) a double row of the byte matrix, the Δ-active S-box is in the key schedule computation of the second-to-last subkey.

These explanations implicitly consider that the event of having a single Δ-active S-box is much more probable than having two or more of them. Therefore, the probability of observing an almost-collision is equal to the probability of having zero Δ-active S-boxes among the first 160 S-boxes and at most 1 among the remaining 40 S-boxes. This means:

$$P \approx \left(\frac{256-\delta}{256}\right)^{160}\left(\left(\frac{256-\delta}{256}\right)^{40} + 40\left(\frac{256-\delta}{256}\right)^{39}\frac{\delta}{256}\right) \approx 2^{-\delta} \quad \delta \leq 16 \qquad (3)$$

Therefore to estimate attack workload in the rest of the paper we will use this approximated value of the almost-collision probability; the error for $\delta = 8$ is 2.4%.

The attack works in two phases, presented in the two sections below for both AES and Camellia.

2.2 First Phase

The aim of the first phase is to find the complete relationship between S_1 and S_2, i.e. a function $T : F_{2^8} \to F_{2^8}$ for which we have $S_2(T(x)) = S_1(x), \forall x$. Obviously T differs from the identity function in exactly δ values. Note that the knowledge of T says nothing about the values of S_1 and S_2, it is only characterizing their relationship.

AES — In AES the input of the 16 S-boxes of the first round is a XOR between plaintext bytes and key bytes (both controlled by the attacker); thus we can do the following: we initialize 2^m (256) counters, one for each possible S-box input. We then submit a certain number of random (p, k) queries to the two δ-related oracles; if the query results in a collision we increment the counters

corresponding to the 16 inputs of the S-boxes of the first round. The counters corresponding to the values in Δ can never be incremented, except in the case where a difference propagating in the cipher is corrected at a later stage. Since the probability of this event is in general negligible compared to the collision probability, after having observed about 2^{10} collisions, the only counters which are left at 0 give us the S-box inputs in the Δ set. Note that this works even if we do not know in advance the value of δ as we will simply obtain it as $|\Delta|$. Once we know the number and positions of the differences in the two S-boxes S_1 and S_2 we proceed as follows: for all possible pair of values $d_i, d_j \in \Delta$ we generate a set of $2^{5+\delta}$ (p, k) pairs so that all S-boxes in the first round receive random inputs (not belonging to Δ) except one S-box, which will be fed with d_i in the first cipher, and with d_j in the second cipher. If $S_1(d_i) = S_2(d_j)$ we will observe almost collisions for the set of queries, otherwise not. Once all $\delta(\delta - 1)2^{5+\delta}$ queries are made we know T.

Camellia — For the Camellia cipher, we proceed in an analogous way, but since the subkey used for the first round is obtained with 4 applications of the round function, we cannot use them directly to obtain T. Instead, we will use the S-boxes in the first F function in the key schedule, whose inputs are completely controllable (key bytes are XORed with known constants). The targets are the two s_1 instances in the first F function of the key schedule, and we proceed with the same counter strategy we used for AES; since we have to compensate for the reduced number of S-boxes, we will need about $2^{14+\delta}$ encryptions. In the case of Camellia, the first phase stops here as we cannot use the same technique we used for AES to completely characterize T (this is due to the fact that a XOR difference pattern in a SPN network can be completely eliminated with one Δ-active S-box, while this is not possible in a Feistel structure). However, as we will see below, this has no impact on the attack.

The first phase requires at most 2^{30} encryptions if $\delta \leq 16$ for both algorithms.

2.3 Second Phase

The aim of the second phase is to use the knowledge we obtained on T in order to recover the full specification of the S-boxes S_1, S_2.

AES — The main tool is still the possibility to produce collisions between the two encryption oracles with non-negligible probability, and we use the subkey XORs and the interaction between the key schedule and the round function as a target for our attack. Since it is difficult to impose and verify conditions directly onto the S-box values, we will work on the XOR differences within the S-box entries; that is, we imagine to take an (unknown) entry of the S-box as reference, and we will try to determine the (XOR) difference between this reference value and all other S-box entries.

First, we choose the reference entry b; for simplicity we impose that $b \notin \Delta$. We then generate a set of (p, k) queries of a certain kind; the key value k is the following, where r stands for a random value (i.e. a byte value which changes

for each pair and for each byte position), a and c are fixed byte values inside each set:

$$k = \begin{bmatrix} a \oplus (01) & r & r & b \\ a & r & r & b \\ a & r & r & b \\ a & r & r & b \end{bmatrix}$$

The associated plaintext value in the pair is the following:

$$p = \begin{bmatrix} c \oplus k(0,0) & r & r & r \\ r & c \oplus k(1,1) & r & r \\ r & r & c \oplus k(2,2) & r \\ r & r & r & c \oplus k(3,3) \end{bmatrix}$$

where $k(i,j)$ is the paired key byte at row i and column j; now, for each pair in the set the input of all S-boxes of the second round is:

$$\begin{bmatrix} S(b) \oplus S(c) \oplus a & r & r & r \\ S(b) \oplus S(c) \oplus a & r & r & r \\ S(b) \oplus S(c) \oplus a & r & r & r \\ S(b) \oplus S(c) \oplus a & r & r & r \end{bmatrix}$$

Pairs belonging to a set have fixed value for a and c and random values for bytes marked with r. Each set is made up by $2^{5+\delta}$ pairs. We have a total of 2^{16} sets which account for all possible combinations of values for a and c.

All the pairs of one set are submitted for encryption to the two oracles; if no almost-collision is observed, it means with high probability that the first column of S-boxes in the second round receive an input belonging to the set Δ, i.e.

$$S(b) \oplus S(c) \oplus a \in \Delta \tag{4}$$

For each value of c, this happens for δ values of a that we can denote as a_{δ_i}; let us call the set of these values A. By looking at equation 4 we easily realize that the set Δ and the set A are the same set of values, apart from an additive (XOR) constant, and this constant is precisely one entry of the S-box difference table at index c taking entry at b as base. Thus we can easily reconstruct the full XOR difference table of the S-box S_1; if $c \in \Delta$, we take the additional care of remapping all values of c in the query as submitted to the first oracle (instantiating S_1) with the value $T(c)$ in the query submitted to the second oracle (instantiating S_2). Once we have the complete XOR difference table of S_1, we just have to guess the value of $S_1(b)$ and with a mere 256 trial encryptions we will obtain the complete content of S_1; S_2 is then immediately obtained as we already know the remapping function T.

The computational cost of phase two is roughly equal to $2^{21+\delta}$ queries to the two ciphers. This algorithm has been implemented in C and tested to work; it takes few minutes on a ordinary PC to recover the complete specification of secret 8-related AES block ciphers; the search on 16-related S-boxes is still practical (2^{38} total encryptions). Note that we did not employ parallelization

or dedicated HW for the search, two things which could make the algorithm practical for even bigger values of δ.

This algorithm works on the vast majority of cases, but there are some instances of the problem which are not tractable. The reason is that the set A is not ordered with respect to Δ, in other words we have no means to discriminate between a_{δ_i} and a_{δ_j} as all that we observe from the queries is that no almost collision could be observed, i.e. that we produce some value δ_i on the input of the S-boxes. Thus, when we look for the XOR constant that transforms the set A in Δ we may end up with multiple values. Let's try to define more precisely the problem.

Definition 2. *If we denote as $\sigma_{i,j}$ the XOR difference between δ_i and δ_j, we define the non-ordered set*

$$\Sigma_k = \sigma_{i,j} | i = k \tag{5}$$

Now if $\Sigma_i = \Sigma_j, \forall i, j$ then the algorithm above is guaranteed not to work. Let's see a practical example.

Example 1. Let us take $\delta = 4$, and let's impose that $\sigma_{3,4} = \sigma_{1,2}$. Then we have that:

$$\begin{aligned}
\delta_1 \oplus \delta_2 &= \sigma_{1,2} \\
\delta_1 \oplus \delta_3 &= \sigma_{1,3} \\
\delta_1 \oplus \delta_4 &= \sigma_{1,4}
\end{aligned}$$

$$\begin{aligned}
\delta_2 \oplus \delta_1 &= \sigma_{1,2} \\
\delta_2 \oplus \delta_3 = \sigma_{1,2} \oplus \sigma_{1,3} &= \sigma_{3,4} \oplus \sigma_{1,3} = \sigma_{1,4} \\
\delta_2 \oplus \delta_4 = \sigma_{1,2} \oplus \sigma_{1,4} &= \sigma_{3,4} \oplus \sigma_{1,4} = \sigma_{1,3}
\end{aligned}$$

etc...

so that $\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4$. Thus, when we look for the XOR constant that transforms A in Δ, we will get 4 such values, only one of the four being the true value of $S(b) \oplus S(c)$ for that given value of c. If the set Δ is randomly generated, the chance of falling into this case is the chance that $\delta_3 \oplus \delta_4 = \delta_1 \oplus \delta_2$, i.e. one out of 256. □

Note that the chance of getting such a hard instance is 2^{-32} for $\delta = 8$ and 2^{-64} for $\delta = 16$; thus for interesting cases, we will have only a small chance of not succeeding.

However, if we take $\delta = 2$, our search algorithm will never work; for this case we give here an additional step which can anyway retrieve S_1 and S_2, showing that with little more effort these difficult cases can be overcome. This is particularly interesting because additional properties of the AES algorithm are used and because the case of 2-related S-boxes can perhaps easily be produced on a single device by introducing faults in the S-box computation phase, targeting

for instance a single S-box during the first round of computation. On the other hand, the generalized solution to these hard instances is left as an open problem.

For $\delta = 2$, we get two plausible values for each entry of the S-box XOR difference table; moreover all values come in pairs, so for instance, and depending on the S-box, if

$$S(b) \oplus S(c) \in \{x, y\} \quad , \quad x \oplus y = \delta_1 \oplus \delta_2 \tag{6}$$

then we also have another entry c' for each we also have that

$$S(b) \oplus S(c') \in \{x, y\} \tag{7}$$

and this is because for each real difference value $dS = S(b) \oplus S(c)$ there is always another one equal to $dS \oplus \delta_1 \oplus \delta_2$.

The problem is that we do not know which of the two options is valid for each entry, i.e. in the end if:

$$S(c) = S(b) \oplus x \tag{8}$$

$$S(c') = S(b) \oplus y \tag{9}$$

or vice versa; establishing the real difference table with this information would cost 2^{126} encryption trials, as one can compare the output of each trial with that of the two oracles (in other words we do not need to guess the XOR differences at δ_1 and δ_2).

To solve this problem, we will leverage on the properties of the MixColumn operation which is carried out in the first round. We will use this operation to produce the δ_i values at the input of the second round and to establish relationships between the possible values in the XOR difference table of the secret S-box.

Let us choose one index of the secret S-box which is different from those in the set $\{b, \tilde{b'}, \delta_1, \delta_2\}$; let us call this index p_1, let us denote its two possible difference values determined before as $dS'(p_1)$ and $dS''(p_1)$ and let us call its conjugate index $\tilde{p_1}$ (the index with the same set of plausible difference values). Then, consider the index p_2 (or its conjugate, it does not change anything) such that the following condition is valid:

$$(02) \cdot dS'(p_1) \oplus (03) \cdot dS'(p_2) = \delta_1 \tag{10}$$

where multiplication is carried out in GF(2^8) using the AES polynomial. Index p_2 is unique and well determined (up to its conjugate) as:

$$dS'(p_2) = (03)^{-1} \cdot p_1 \oplus (03)^{-1} \cdot (02) \cdot dS'(p_1) \tag{11}$$

is an affine relationship. Note that if Equation 10 holds, then:

$$\begin{aligned} & (02) \cdot dS'(p_1) \oplus (03) \cdot dS''(p_2) = \\ = & (02) \cdot dS'(p_1) \oplus (03) \cdot dS'(p_2) \oplus (03) \cdot (\delta_1 \oplus \delta_2) = \\ = & \delta_1 \oplus (03) \cdot \delta_1 \oplus (03) \cdot \delta_2 = \\ = & (02) \cdot \delta_1 \oplus (03) \cdot \delta_2 \end{aligned} \tag{12}$$

which is always different from both δ_1 and δ_2 since $\delta_1 \neq \delta_2$.

On the other hand:

$$\begin{aligned}
&(02)\cdot dS''(p_1)\oplus(03)\cdot dS''(p_2)=\\
&=(02)\cdot dS'(p_1)\oplus(02)\cdot(\delta_1\oplus\delta_2)\oplus(03)\cdot dS'(p_2)\oplus(03)\cdot(\delta_1\oplus\delta_2)=\\
&=\delta_1\oplus(02)\cdot\delta_1\oplus(03)\cdot\delta_1\oplus(02)\cdot\delta_2\oplus(03)\cdot\delta_2=\\
&=\delta_2
\end{aligned}\tag{13}$$

and again it is easy to see that:

$$(02)\cdot dS''(p_1)\oplus(03)\cdot dS'(p_2)\neq\{\delta_1,\delta_2\}\tag{14}$$

For the sake of clearness let us define the Boolean function

$$\mu(a,b)=\begin{cases}\text{True} & \text{if}\quad (02)\cdot a\oplus(03)\cdot b\ \in\{\delta_1,\delta_2\}\\ \text{False} & \text{if}\quad (02)\cdot a\oplus(03)\cdot b\ \notin\{\delta_1,\delta_2\}\end{cases}$$

then we can summarize the discussion above by saying that

$$\mu(dS'(p_1),dS'(p_2))\iff\mu(dS''(p_1),dS''(p_2))\tag{15}$$
$$\mu(dS'(p_1),dS''(p_2))\iff\mu(dS''(p_1),dS'(p_2))\tag{16}$$

Now consider the two real values of the difference at indexes p_1 and p_2, we write them as $dS(p_1)$ and $dS(p_2)$. If we could produce the value $\mu(dS(p_1),dS(p_2))$ at the input of one S-box, we would build a set of pairs with this characteristic and if no collision would be observed in the two oracles, then we would know that a δ_i was produced, i.e. that:

$$dS(p_1)=dS'(p_1)\quad\Rightarrow\quad dS(p_2)=dS'(p_2)\tag{17}$$
$$dS(p_1)=dS''(p_1)\quad\Rightarrow\quad dS(p_2)=dS''(p_2)\tag{18}$$

and if collisions *could* be observed, then we would know that:

$$dS(p_1)=dS'(p_1)\quad\Rightarrow\quad dS(p_2)=dS''(p_2)\tag{19}$$
$$dS(p_1)=dS''(p_1)\quad\Rightarrow\quad dS(p_2)=dS'(p_2)\tag{20}$$

In other words, we would establish a link between the real XOR difference value at index p_1 and that at index p_2 and we would decrease by one bit the search space needed to find the real S-box table. The shape of the plaintext and key values in every pair of such set is the following:

$$k=\begin{bmatrix}(01) & r & r & r\\ r & r & r & b\\ r & r & r & r\\ r & r & r & r\end{bmatrix}$$

such that the second subkey is:

$$\begin{bmatrix}S(b) & r & r & r\\ r & r & r & r\\ r & r & r & r\\ r & r & r & r\end{bmatrix}$$

and the associated plaintext value is:

$$p = \begin{bmatrix} p_1 \oplus k(0,0) & r & r & r \\ r & p_2 \oplus k(1,1) & r & r \\ r & r & p_1 \oplus k(2,2) & r \\ r & r & r & p_1 \oplus k(3,3) \end{bmatrix}$$

Therefore at the end of the first round, right after the XOR with the second subkey the leftmost and topmost byte in the state matrix is equal to:

$$\begin{aligned} & S(b) \oplus (02) \cdot S(p_1) \oplus (03) \cdot S(p_2) \oplus S(p_1) \oplus S(p_1) = \\ = \; & S(b) \oplus (02) \cdot dS(p_1) \oplus (02) \cdot S(b) \oplus (03) \cdot dS(p_2) \oplus (03) \cdot S(b) = \\ = \; & (02) \cdot dS(p_1) \oplus (03) \cdot dS(p_2) = \\ = \; & \mu(dS(p_1), dS(p_2)) \end{aligned} \tag{21}$$

while all other bytes are random. This is exactly the value we need to obtain the one bit of information from the set.

Once the link between p_1 and p_2 is established, we can iterate this procedure taking p_2 as starting point and so on; in the end, we will have established links between all XOR differences in the table and the real difference value at index p_1. Now, to obtain the complete S-box we will have just to guess the value at the reference index, $S(b)$, and the XOR difference value at index p_1. With an effort of about 2^{17} encryptions, the search space has thus been reduced to 2^9, which is trivial to brute-force. The procedure has been implemented in C and tested to work.

Camellia — The second phase of attack for the Camellia cipher is rather different from that of AES; our target will not be the XOR difference table of the secret S-box, we will instead retrieve the S-box itself. First, let's take a (p, k) query which leads to an almost-collision; we have already produced a lot of them in the first phase of attack; in the following we will keep the value of the key fixed at k, so that we are sure that no S-box in the key schedule is Δ-active. Then, consider the Camellia F function. First, all input bytes are XORed with subkey bytes (which in our analysis will be considered unknown); then they are passed through an array of S-boxes and then through the mixing layer, known as P function. Let us concentrate our attention on byte 5 of the F output of the first round, which is obtained as the XOR of bytes 1,2,6,7 and 8 of the input (after key addition and S-boxes have been applied). Let us keep the values of the input bytes 1,2,6 and 7 to some values which lead to a ciphertext collision; then, let us prepare $2^{5+\delta}$ queries for each possible combination of values of input bytes 8 and 12; byte 12 is the byte which is XORed with byte 5 of the F output to form an input byte for the second round.

If, for a given value of input byte 8, we find that all values of byte 12 lead to no collision, it means that we are querying the s_1 S-box on byte 8 with a value in the Δ set; this happens for exactly δ values of byte 8 and from those we can easily obtain the value of the subkey byte which XORs with input byte 8.

On the other hand, if for a value of byte 8 we find that exactly δ values of byte 12 lead to no collision, it means we are producing the set Δ on the input of

the S-box in the second round. By comparing the set of these values of byte 12 with Δ, we obtain an entry of S-box s_1 XORed with an unknown but constant byte which is the combination of all unknown constant contributions from other input bytes of round 1 and the subkey byte of round 2. So if we write down all these values, and by making a re-arrangement implied by the subkey byte 8 we have found, we obtain an S-box table which is equal to s_1 apart from an additive constant. So with a mere additional 256 encryptions, we will recover the complete content of s_1, actually the content apart from the indexes in the set Δ. For those, we will have to guess the correct arrangement, i.e. an additional effort of $\delta!$ encryption trials.

The effort of phase 2 is equal to $2^{22+\delta} + \delta!$ encryptions; for $\delta = 16$ the factorial dominates and we have an effort of about 2^{44} encryptions.

3 Discussion and Conclusions

Our attack is easily applicable only if the size of S-boxes is such that the collision probability is high enough to practically employed; 8-bit S-boxes are good candidate. Apart from this, we have seen successful reverse engineering of two quite different ciphers (AES and Camellia); we expect that the attack can be applied also to other ciphers based on large S-boxes (Clefia, Twofish and Kasumi among the others). However, if we try to apply our attack to ciphers which instance 4-bit S-boxes, we see that the collision probability is too small to be used, even for the smallest values of δ. For example, 2-related instances of PRESENT [7] would show a collision probability of only 2^{-100}. This is a point in favor of such ciphers, which are in general more compact for hardware implementations and seem to be more flexible under this aspect.

Previous work exist on the utilization of cryptanalysis to retrieve the content of unknown S-boxes, see for instance [9][6]. These papers present techniques which can obtain the S-boxes of reduced-round variants of SPN ciphers. In this paper, we take a different approach and we show that even full ciphers which are designed to be hermetic and resistant to related cipher attacks, can be attacked, provided that the adversary has access to at least two δ-related instances.

If instances of S-boxes are randomly chosen, the probability of success of the presented attack is negligible. In general, we can conclude that the probability of collision between different block cipher instances should be verified to be sufficiently low during the design phase, because it is a tool that can be used by attackers whose goal is to obtain the complete specification of the algorithm.

Also, care should be taken w.r.t to physical attacks, such as fault injection, because it is imaginable that this type of attack could make a single faulty circuit behave like a pair of δ-related ciphers. In this case an attacker may be able to attack a single instance of unknown AES-like cipher using the techniques presented in this paper. We think that this could be an interesting direction for future research, especially considering the fact that FIRE attacks on AES ciphers have not yet been developed.

References

[1] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camellia*: A 128-bit block cipher suitable for multiple platforms - design and analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001)
[2] Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of Cryptology 4, 3–72 (1991)
[3] Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
[4] Biryukov, A., De Cannire, C., Braeken, A., Preneel, B.: A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 648–648. Springer, Heidelberg (2003)
[5] Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
[6] Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 395–405. Springer, Heidelberg (2001)
[7] Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
[8] Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
[9] Borghoff, J., Knudsen, L.R., Leander, G., Thomsen, S.S.: Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 270–289. Springer, Heidelberg (2011)
[10] Clavier, C.: An improved SCARE cryptanalysis against a secret A3/A8 GSM algorithm. In: McDaniel, P., Gupta, S.K. (eds.) ICISS 2007. LNCS, vol. 4812, pp. 143–155. Springer, Heidelberg (2007)
[11] Courtois, N.T., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
[12] Daemen, J., Rijmen, V.: The Design of Rijndael - AES - The Advanced Encryption Standard. Springer (2002)
[13] Guilley, S., Sauvage, L., Micolod, J., Réal, D., Valette, F.: Defeating Any Secret Cryptography with SCARE Attacks. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 273–293. Springer, Heidelberg (2010)
[14] Knuth, D.E.: The Art of Computer Programming, 3rd edn. Addison-Wesley (1997)
[15] Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
[16] Novak, R.: Side-Channel Attack on Substitution Blocks. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 307–318. Springer, Heidelberg (2003)
[17] O'Connor, L.: On the Distribution of Characteristics in Bijective Mappings. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 360–370. Springer, Heidelberg (1994)

[18] Réal, D., Dubois, V., Guilloux, A.-M., Valette, F., Drissi, M.: SCARE of an Unknown Hardware Feistel Implementation. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 218–227. Springer, Heidelberg (2008)
[19] Sasaki, Y.: Known-Key Attacks on Rijndael with Large Blocks and Strengthening *ShiftRow* Parameter. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. LNCS, vol. 6434, pp. 301–315. Springer, Heidelberg (2010)
[20] Sung, J., Kim, J., Lee, C., Hong, S.: Related-Cipher Attacks on Block Ciphers with Flexible Number of Rounds. In: Research in Cryptology - 1st Western European Workshop, WEWoRC 2005, Leuven-Heverlee,be. LNCS, p. 10. Springer (2005)
[21] Wang, D., Lin, D., Wu, W.: Related-mode attacks on ctr encryption mode. I. J. Network Security 4(3), 282–287 (2007)
[22] Wu, H.: Related-Cipher Attacks. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 447–455. Springer, Heidelberg (2002)