Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128

Pierre-Alain Fouque[1], Jérémy Jean[2], and Thomas Peyrin[3]

[1] Université de Rennes 1, France
[2] École Normale Supérieure, France
[3] Nanyang Technological University, Singapore

Abstract. While the symmetric-key cryptography community has now a good experience on how to build a secure and efficient fixed permutation, it remains an open problem how to design a key-schedule for block ciphers, as shown by the numerous candidates broken in the related-key model or in a hash function setting. Provable security against differential and linear cryptanalysis in the related-key scenario is an important step towards a better understanding of its construction.

Using a structural analysis, we show that the full AES-128 cannot be proven secure unless the exact coefficients of the MDS matrix and the S-Box differential properties are taken into account since its structure is vulnerable to a related-key differential attack. We then exhibit a chosen-key distinguisher for AES-128 reduced to 9 rounds, which solves an open problem of the symmetric community. We obtain these results by revisiting algorithmic theory and graph-based ideas to compute all the best differential characteristics in SPN ciphers, with a special focus on AES-like ciphers subject to related-keys. We use a variant of Dijkstra's algorithm to efficiently find the most efficient related-key attacks on SPN ciphers with an algorithm linear in the number of rounds.

Keywords: SPN, Block Cipher, AES, Related-Key, Chosen-Key.

1 Introduction

Block ciphers and hash functions are among the most important primitives in cryptography and while their respective goals are different, they are related in many ways. For example, most compression functions, which can in turn be used to define a hash function, are built upon an internal block cipher thanks to classical constructions [19,28] such as Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO) or Miyaguchi-Preneel (MP). One of the main differences between the two primitives is that in the case of the block cipher, the key input is unknown and uncontrolled by the attacker, whereas for the compression function, the attacker has full control over the key schedule (generally called message expansion in that context). Yet, the so-called *related-key* attack scenario [3, 5] is interesting for both cases. This model allows the attacker to incorporate differences not only in the plaintext or ciphertext input, but also in the key input. While less relevant in practice than the classical single-key model, it is important to analyze block ciphers in the light of related-key attacks since the secret keys are

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 183–203, 2013.

often updated in security protocols or differences can be incorporated using fault attacks. Moreover, related-key attacks are also very important when the block cipher is used as inner primitive of a hash function, and in that setting one can even consider the known-key [22] or chosen-key models [8] where the attacker is given knowledge or complete control of the key and his goal is to exhibit some non-ideal property of the primitive.

Avoiding high-probability related-key differential characteristics is one of the goal of the key schedule, and so far various directions have been investigated to construct this component. Resisting to attacks in this setting has for example been taken into account in the conception of the Whirlpool hash function [1], by using the same AES-like permutation for both the internal permutation and the message expansion part, leading to a strong key schedule in terms of number of Sbox calls, but quite slow as it represents about half of the total amount of computations. As a complete opposite, the designers of the LED block cipher [17] chose to use no key schedule at all, at the expense that an important number of rounds is required. These two functions can both provide provable security with regard to related-key differential attacks, but they also both suffer from efficiency issues. In general (see for example AES or PRESENT [11]), key schedules are built by using an ad-hoc and relatively light function that is quite different from the main permutation, in a hope that this will avoid any correlation between the two components and enforce low-probability related-key differential characteristics. However, because of the heuristic design process and the difficulty of the task, no real security argument is given and this can eventually lead to security issues [7,9]. To help designers and cryptanalysts, many automated differential analyses have already been applied to various primitives [9, 12, 13, 20, 29].

The AES block cipher [14] is currently the most interesting candidate to scrutinize with regard to related-key, chosen-key attacks or when used as a black-box in cryptosystems: during the NIST SHA-3 hash function competition, many candidates [2, 4, 15] reused some components from the AES and related-key attacks on the AES-192 and AES-256 [7, 8] have been discovered. While differential and linear attacks against the AES in the single-key scenario seem to be mastered since the design of the cipher focuses in particular to resist to those class of attacks, provable security against related-key attacks remains more complex to tackle.

Graph Traversal Algorithms. In [24], Matsui proposes an algorithm to find the best differential characteristics for DES. The strategy to find the best one on n rounds first starts by computing the best ones on 1 to $n-1$ rounds. The algorithm works by induction and can be seen as a tree traversal in a depth-first manner, where the tree represents all the possible differential characteristics in the cipher layered by round. The nodes represent the actual differences and the edges the possible transitions between them, and are labeled by their probabilities. One differential characteristic is a path in this tree, and its probability equals the product of all traversed edges. We are looking for the path with the highest probability in this tree. The knowledge of the previous best characteristics, i.e. up to some depth in the tree, allows pruning during the procedure like the A^*

heuristic [18]: the target value being known (the exhaustive search bound), we can reduce the possibilities for each one-round transition. Using such algorithm, the complexity is exponential in the number of nodes in the tree, and therefore in the block-size and the number of rounds, except if the pruning is very efficient.

In modern byte-oriented ciphers, designers ensure there is a fast diffusion and that all actual differential transitions occur with the same probability: all differences become equivalent. Consequently, Matsui's search algorithm becomes less efficient since there is no dominant characteristic. Biryukov and Nikolić propose in [9] to restrict the search to truncated differences to decrease the number of edges in the tree. They also introduce a nice representation of truncated differences to consider the branching (combinatorial explosion of differences) in the key schedule.

In this article, we change the tree representation from the previous works into a graph: the nodes and edges have the same signification as before, but we merge all the nodes representing the same differences into a single one. Matsui's tree encodes all the paths of our graph. This merging allows to view the path search as a Markov process: round i is independent of the paths in rounds 0 to $i-1$. Consequently, the numbers of nodes and edges become linear in the number of rounds. Finding the best differential characteristics is reduced to a shortest path problem: we want all the shortest paths in this graph to get all the differential characteristics with the highest probabilities. We use a variant of Dijkstra algorithm combined with the A^* heuristic to explore a kind of *graph product* in a breadth-first manner. Our algorithm uses a dynamic programming method, which was considered too costly in terms of memory in [9]. This approach solves the problem of finding the best related-key characteristics using graph algorithms in polynomial time in the number of rounds and exponential in the state, whereas the previous best known methods were exponential in both parameters using Matsui's algorithm variants (the search in [9] was made possible thanks to an extreme pruning in the AES tree).

Structural Evaluation. By structural evaluation, we mean the domain of cryptography that analyzes a cryptosystem in terms of generic constructions using black-box elements. We are interested in how the building blocks of the primitives interact together, while "ignoring their semantic definitions as particular functions" as in meet-in-the-middle attacks [10].

In this line of research, a major result is the conception of Rijndael, or how to construct a block cipher provably resistant to differential attacks. Daemen and Rijmen show in [14] a lower bound B_r on the number of active Sboxes for any differential characteristic on r rounds of Rijndael, when no difference is introduced in the key. For an Sbox with maximal differential probability p_{max}, this result allows to upper bound the probability of success of any differential attack on r rounds by $p_{max}^{B_r}$. For k-bit keys block ciphers, the resistance to differential cryptanalysis means $p_{max}^{-B_r} > 2^k$, which gives a criteria on r and p_{max} for the security of the cipher. In [10], Biryukov and Shamir analyze the SASAS construction, alternating five layers of non-linear S and affine A functions. They show that five such rounds are vulnerable to a very efficient structural attack, even though the adversary does not know anything about the inner structure of both S and A. Finally, we mention the work by Miles and Viola in [26], where they prove the security of bounded-input-length pseudo-random functions based on Substitution-Permutation Networks like the AES. In this article, we study the structural analysis of generic Substitution-Permutation Networks in the related-key model. Contrary to the single-key model, it seems impossible to prove anything on the key schedule resistance in the same vein as [14, 26], so we build a tool to study this problem.

Open-Key Model. The open-key model has been introduced so as to investigate the security of block ciphers in relaxed versions of the standard model. In [22], Knudsen and Rijmen studied what they called the *known-key* model where the key is public and the goal of the adversary consists in distinguishing the cipher from a random permutation. At Crypto 2009, Biryukov et al. introduced in [8] a more relaxed version called *chosen-key* model where the adversary must exhibit a nontrivial property of the cipher when he has the freedom of the key bytes as extra parameters. They show how to find differential q-multicollisions for AES-256 in time $q \cdot 2^{67}$. For an ideal cipher, constructing q-multicollisions would require at least $O\left(q \cdot 2^{\frac{q-1}{q+1}128}\right)$. At Fse 2010, Gilbert and Peyrin introduced in [16] particular properties for the known-key model by using high-probability differential characteristics on 8 rounds of the block cipher AES-128. Given a key k, two known subspaces Δ_{IN} and Δ_{OUT}, they show how to find one pair of inputs (m, m') to the block cipher E_k such that $m \oplus m' \in \Delta_{IN}$ and $E_k(m) \oplus E_k(m') \in \Delta_{OUT}$ more efficiently than a generic attack on a random permutation, based on a restricted variant of the birthday paradox. In this work, given δ, Δ_{IN} and Δ_{OUT}, we are interested in finding a pair of keys (k, k') and a pair of messages (m, m') such that $k \oplus k' = \delta$, $m \oplus m' \in \Delta_{IN}$ and $E_k(m) \oplus E_{k'}(m') \in \Delta_{OUT}$.

Our Contributions. The goal of this article is twofold. First, we describe an efficient and generic tool that computes all the best differential characteristics for general SPN ciphers, in particular for AES-like ciphers, and then we apply it to the structure of the AES-128. While our algorithm also works in the single-key setting and retrieves the tight proven bounds of the AES structure, we focus this article on the related-key model where the classical XOR difference is the relation in the keys. The search complexity for related-key differential is improved from several days of computations in [9] with a depth-first algorithm to a few hours on a single PC using our breadth-first approach. We also show that the theoretical upper bound $2^{-6 \cdot 17} = 2^{-102}$ mentioned in [9] for the best 5-round characteristic cannot be reached, since the truncated characteristic can only be instantiated with a probability at most 2^{-105}.

As an application of our tool, we study AES-128 as a particular SPN cipher. First, we perform a structural analysis where we consider the MDS property of the diffusion layer, but we do not specify its coefficients. The results show that in order to prove the security of 10 rounds of the cipher in the related-key model, one needs to consider more than just its structure: one needs in particular to consider the differential properties of the non-linear Sbox. Secondly, we analyze the structure of AES-128 in the hash function setting, where the key schedule and the message parts can be attacked somewhat independently by the adversary. We also show that this setting cannot be proven secure against differential cryptanalysis unless additionnal information about the instantiation of the SPN cipher are provided (e.g., the Sbox and the linear layer). Finally, we construct a chosen-key distinguisher for 9 rounds of AES-128 that requires 2^{55} simple operations and 2^{32} words of memory storage: this was considered an open problem until now, e.g. [9]. Our distinguisher exhibits a non-random property in the chosen-key setting and such a property for an ideal cipher would be detected only after 2^{68} encryptions queries.

Organization of the Paper. In Section 2, we first introduce definitions regarding the ciphers studied and the types of differences we analyze. We then describe our generic tool and give some improvements in the specific case of AES-like ciphers in Section 3. Finally, we present the results of our structural evaluation in Section 4.1, more specifically on AES-128 in Section 4.2, and we precise the construction of the chosen-key distinguisher for 9 rounds of AES-128 in Section 4.3.

2 Definitions

2.1 SPN and AES-Like Ciphers Description

To keep our reasoning as general as possible, we give in this subsection a generic description of Substitution-Permutation Network (SPN) ciphers, and we identify the subgroup of the AES-like ciphers. We refer to the corresponding specifications for a detailed description of the AES [14,27]. We consider that the block ciphers studied here take as input a plaintext or ciphertext of size n bits, and a key of size k bits. The cipher is composed of R successive applications of a round function, and we denote respectively s_i and k_i the successive internal states of the block cipher and the key schedule after the i-th round. The state s_0 is initialized with the input plaintext and k_0 with the input key. One round i is itself composed of three layers: a key extraction and incorporation layer (AK) where a n-bit round-key rk_{i-1} is extracted from k_{i-1} and xored to s_{i-1}, a block cipher permutation layer $\mathcal{BC}$ that updates the n-bit current state of the block cipher after addition of the subkey, i.e. $s_i = \mathcal{BC}(s_{i-1} \oplus rk_{i-1})$, and a key schedule transformation layer $\mathcal{KS}$ that updates the k-bit current state of the key schedule, i.e. $k_i = \mathcal{KS}(k_{i-1})$. The final ciphertext is then defined as $s_R \oplus rk_R$.

Definition 1. (SPN cipher) *Let a block cipher $\mathcal{E}$ whose internal state is viewed as a t_{BC}-cell vector (where $t_{\mathsf{BC}} = \frac{n}{b}$), each cell representing a b-bit word, and the key schedule as a t_{KS}-cell vector (where $t_{\mathsf{KS}} = \frac{k}{b}$). The block cipher $\mathcal{E}$ is called an SPN cipher when its round function $\mathcal{BC}$ is made of a linear function P and a non-linear permutation S, with $\mathcal{BC} = \mathsf{P} \circ \mathsf{S}$, the latter applying one or distinct b-bit S-Boxes to every cell.*

In the even more particular case of AES-like ciphers, the internal state of $\mathcal{BC}$ can be viewed as a square matrix of b-bit cells with d rows and d columns ($n = d^2 \cdot b$).

A cell of s_i is denoted by $s_i^{x,y}$, where x is its row position and y its column position in the square matrix, starting the counting from 0. Then, the linear layer is itself composed of the $\mathsf{ShiftRows}$ transformation (ShR), that moves each cell by x positions to the left in its own row, and the $\mathsf{MixColumns}$ transformation (MC), that linearly mixes all the columns of the matrix separately. Overall, for AES-like ciphers we have $\mathcal{BC} = \mathsf{P} \circ \mathsf{S} = \mathsf{MC} \circ \mathsf{ShR} \circ \mathsf{S}$ (Figure 1).

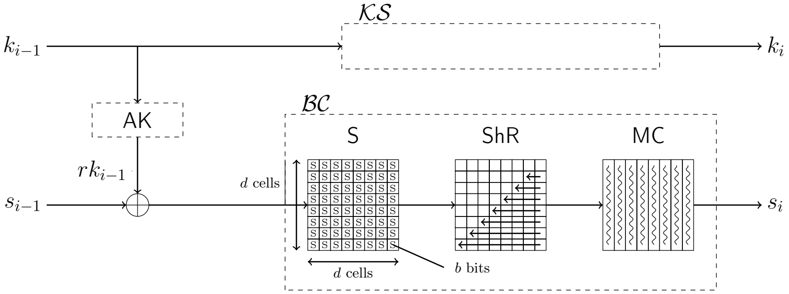


Fig. 1. One round of the generic SPN and AES-like ciphers

2.2 Truncated and Actual Differences

In this article, we are interested in differential attacks [6]. Usually, in this scenario the attacker looks for the bitwise difference between two state values. However, here we also consider truncated differential attacks [21].

Definition 2. *Let $A = [A^{x,y}]$ and $B = [B^{x,y}]$ two states. We denote their* ***truncated difference*** *by $\Delta = [\Delta^{x,y}]$ with $\Delta^{x,y} = 1$ if and only if $A^{x,y} \neq B^{x,y}$ (active cell), and $\Delta^{x,y} = 0$ otherwise (inactive cell). We denote their* ***actual difference*** *by $\delta = [\delta^{x,y}]$ with $\delta^{x,y} = A^{x,y} \oplus B^{x,y}$.*

First, we need analyze the effect of the cipher transformations on the truncated and actual differences.

The Substitution Layer. One can easily check that the substitution layer S has no effect on the truncated difference of a cell: a cell remains in the same active/inactive situation after application of the transformation. However, S has an effect on the actual difference of every active cells. This effect can be visualized by the differential distribution table (DDT) of the S-Boxes. More precisely, for each possible pair $(\delta_{in}, \delta_{out})$ of actual difference on the input/output of the S-Box S, the table gives the number $\mathrm{DDT}(\delta_{in}, \delta_{out}) = x$ of values X that validate this differential transition, i.e. $S(X) \oplus S(X \oplus \delta_{in}) = \delta_{out}$. Alternatively, $x/2^b$ represents the differential probability of the transition. An important criteria that can be derived from this table is the maximal differential probability p_{max}, which is the highest possible differential probability when $\delta_{in} \neq 0$ and $\delta_{out} \neq 0$.

In order to measure the quality of a truncated differential characteristic, we use the classical counting of the number of active S-Boxes appearing in the characteristic, and we denote it $|\cdot|$.

Definition 3. *Let $v = [\Delta^i]$ be a vector of truncated differences. The* ***weight*** *of v is the number of active differences in v: $\sum_{\Delta^i \neq 0} 1$. We denote it $|v|$ and generalize the notion to any matrix v.*

The Permutation Layer for AES-Like Ciphers. Since the ShR layer only moves the cells around, it only changes the active/inactive cells positions in the internal state, but not their number. The same reasoning applies to the actual differences. The MC transformation being linear, the effect on the values and the actual differences is the same and therefore for each column of the internal state, the output actual differences are simply deduced by the application of the MC linear matrix. Concerning the truncated differences, the effect depends on the branching number B_{MC} of the MC matrix. The branching number is the minimum amount of active cells one can get on both the input and the output of the matrix, excluding the case when there are both null. This measure of the diffusion is crucial for the security of many cryptography primitives and, in general, the MC matrix is Maximum Distance Separable (MDS), that is $B_{MC} = d + 1$ is maximal. A valid truncated differential transition forcing i cells to be inactive on the output happens with probability $2^{-b \cdot i}$.

2.3 Structural Evaluation

Considering only truncated differences enables the cipher designers to get an estimation on the quality of the structure of their primitive in regard to provable security. As an example, the designers of the AES can easily derive the minimal number of active S-Boxes in any number of rounds of the cipher in the single-key model [14], by ignoring the instantiation of the matrix of the diffusion layer. This means that the same reasoning applies for *any* diffusion matrix. In this paper, we are also interested in a generalization of this notion for the related-key model.

More formally, we denote by $E^{S,P}$ a block cipher that uses a substitution layer instantiated by S, and a permutation layer instantiated by P. On the one hand, if E represents the AES family of permutations, we can either plug the AES S-Box S_{AES} ($S \leftarrow S_{\text{AES}}$), or a random bijection S ($S \overset{\$}{\leftarrow} \mathfrak{S}_{2^b}$) selected uniformly at random in the set $\mathfrak{S}_{2^b}$ of the permutations on b bits. In the latter setting, we loose the information $p_{max} = 2^{-(b-2)}$ as we can only get $2^{-(b-1)} \leq p_{max} \leq 1$, but we are interested in how the structure of the AES resists to differential cryptanalysis when we relax the strong information on p_{max}. We perform the same abstraction for the P layer, where the matrix is selected uniformly at random in the set of all the $d \times d$ matrices with coefficients in GF(2^b).

3 Generic Related-Key Differential Characteristic Search Tool for SPN Ciphers

In this section, we explain the inner workings of our generic related-key differential characteristic search tool for SPN ciphers. As a first step, we model the problem by assuming that the cipher round function is a Markov process in regard to the truncated differential characteristic search (Section 3.1). This allows us to reduce the problem to a shortest path search in a special $(r + 1)$-equipartite directed acyclic graph, for which we provide a simple yet powerful algorithm. The precomputation phase of the process is devoted to building the graphs on which we work on (Section 3.2), while the online phase looks for the shortest paths (Section 3.3). We note that we can tweak the Markov assumption to find not only the best truncated differential characteristics, but also the actual difference ones.

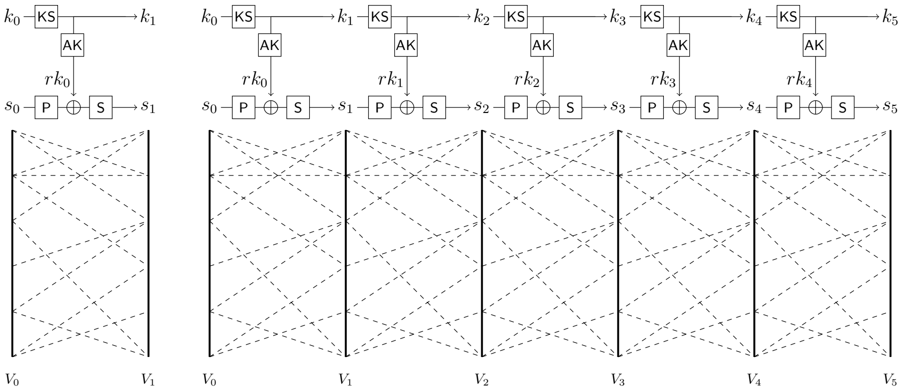
3.1 Differential Characteristic Search as a Graph Modeling of a Markov Process

We describe here an algorithm that generates for any number of rounds *all* the related-key truncated differential characteristics for SPN ciphers with minimal number of active S-Boxes. This analyzes the structure of the cipher in regard to the resistance against related-key attacks. We make a simple assumption: we would like the search to be a Markov process. More precisely, we assume that the possible differential transitions through a round from one truncated state to another one does not depend on previous rounds transitions. If we stick to the real definition of truncated differentials (i.e. without implicit conditions contained), then this assumption is verified for SPN ciphers: knowing the truncated input difference of one round represents all the information needed in order to deduce the possible output ones.

Graph Modeling. In order to find the best r-round related-key truncated differential characteristics, we use a graph modeling of the problem. Let G be the 2-equipartite directed acyclic graph of all the possible one-round transitions. Then all the best r-round related-key truncated differential characteristics correspond to all the shortest paths in the $(r+1)$-equipartite directed acyclic graph G_r built by concatenating r copies of G. Namely, denoting $G = (V_0, V_1 \; ; \; E_{0,1})$ the 2-equipartite graph linking with one cipher round a state in set V_0 to a state in set V_1 using some edge in set $E_{0,1}$, we build the graph G_r representing r rounds of the cipher by $G_r = (V_0, \dots, V_r \; ; \; E)$ such that for all i, the subgraph $(V_i, V_{i+1} \; ; \; E_{i,i+1})$ is equal to G. Note that all edges are oriented from V_i to V_{i+1}, and that $|V_i| = |V_{i+1}|$. The nodes of the graph stand for all the possible pairs $(\Delta_{\mathsf{KS}}, \Delta_{\mathsf{BC}})$ where Δ_{KS} represents the truncated difference on the key schedule state and Δ_{BC} represents the truncated difference on the block cipher state. Since we have $2^{t_{\mathsf{KS}}}$ possible values Δ_{KS} and $2^{t_{\mathsf{BC}}}$ possible values Δ_{BC}, all V_i in the graph are composed of $2^{(k+n)/b}$ nodes. The edges correspond to a possible one-round related-key truncated differential characteristic from the input to the output vertex and in the worst case where all differential transitions are possible, we have $2^{2(k+n)/b}$ edges. A path in G_r is defined as a sequence of $r+1$ nodes, one in each of the V_i.

We note that the probability costs are not associated to the edges, but to the output nodes. Indeed, the number of active cells in the output node represents the number of active S-Boxes during this round[1]. We denote C_{BC} (resp. C_{KS}) the total number of active S-Boxes in the internal permutation part of the block

[1] To be able to associate the number of active S-Boxes in the key schedule to the output node as well, we make the weak assumption that one round of the key schedule is composed of an S-Box and a linear layer at most.



(a) Graph G.

(b) Graph G_5.

Fig. 2. Examples of simplified versions of the two graphs G and G_5. Variables s_i and k_i represent the current internal permutation/key state respectively, while rk_i stands for the subkey generated during the round.

cipher (resp. in the key schedule part) in the whole characteristic. Depending on the situation considered, one might want to minimize $C_{\mathsf{BC}} + C_{\mathsf{KS}}$ for classical scenarios, or instead $\max\{C_{\mathsf{BC}}; C_{\mathsf{KS}}\}$ for hash function settings, where the key schedule and the block cipher parts can be attacked sequentially (first the key schedule part, and then the block cipher one).

Theorem 1. (Search algorithm) *Let $\mathcal{E}$ be a SPN cipher on n-bit blocks using a k-bit internal state in the key schedule. Both states are viewed as vectors of b-bit cells. There exists an algorithm $\mathcal{A}$ with a theoretical time complexity of $O(r \cdot 2^{(2n+k)/b})$ that finds all the best characteristics on r rounds of $\mathcal{E}$.*

We emphasize that algorithm $\mathcal{A}$ will find *all* the shortest paths in G_r representing the differential transitions of r rounds of $\mathcal{E}$. Moreover, we note that the time complexity of $\mathcal{A}$ can be greatly reduced with heuristics. We describe in the next two sections our tool that searches for the best r-round related-key truncated differential characteristics.

3.2 Precomputation Phase

The precomputation phase builds the graph G. It can be built and stored efficiently by observing its inner structure: the block cipher internal state output depends only on the block cipher internal state input and the incoming subkey (deduced by the extraction phase from the key schedule internal state input), while the key schedule internal state output depends only on the key schedule internal state input. Therefore, G can actually be described as a special product of two smaller graphs G_{BC} and G_{KS}, such that an edge $(s_i, k_j) \to (s_{i'}, k_{j'})$ exists in G if and only if $k_j \to k_{j'}$ exists in G_{KS} and $(s_i, k_j) \to s_{i'}$ exists in G_{BC}. On the one hand, G_{BC} is a bipartite directed acyclic graph whose input nodes are all the possible block cipher internal state and subkey pairs, and whose output nodes are all the possible block cipher internal state. The edges represent input nodes that can be mapped to output nodes through a valid differential transition. On the other hand, G_{KS} is a 2-equipartite directed acyclic graph, whose input and output nodes are all the possible key schedule internal states. The edges represent input nodes that can be mapped to output nodes through a valid differential.

This observation slightly reduces the amount of computation/memory to build/store G: the number of vertices in G_{BC} is $v_{\mathsf{BC}} = 2^{t_{\mathsf{BC}}+t_{\mathsf{KS}}} + 2^{t_{\mathsf{BC}}}$ and the number of vertices in G_{KS} is $v_{\mathsf{KS}} = 2 \times 2^{t_{\mathsf{KS}}}$. This has to be compared with the $2 \times 2^{t_{\mathsf{BC}}+t_{\mathsf{KS}}}$ nodes in G. For example, in the particular case of the `AES-128`, this trick reduces the number of nodes from 2^{33} in G to $v_{\mathsf{BC}} = 2^{32} + 2^{16}$ in G_{BC} and 2^{16} in G_{KS} and mainly allows to apply an *early-abort* approach to prune edges in G in the online phase. More importantly, the total number of edges shrinks considerably from $e_{\mathsf{BC}} \cdot e_{\mathsf{KS}}$ to $e_G = e_{\mathsf{BC}} + e_{\mathsf{KS}}$, which equals to $2^{33.6} + 2^{22.15}$ in the case of `AES-128`.

The Graph G_{BC}. It can be built by repeating the three following steps for all the $2^{t_{\mathsf{BC}}}$ possible truncated differences Δ_{in} on the input and all the $2^{t_{\mathsf{BC}}}$ possible truncated differences Δ_{out} on the output.

1. Compute all the possible truncated differences Δ_x that can be obtained from Δ_{in} through the P layer (on the backward direction, a truncated difference Δ_{out} stays the same when inverting the S layer).
2. For every Δ_x found, compute all the possible truncated differences Δ_k on the key state that can be obtained from $\mathsf{AK}^{-1}(\Delta_x \oplus \Delta_{out})$.
3. For every Δ_k found, add an edge in G_{BC} from input node (Δ_k, Δ_{in}) to output node Δ_{out} if none exists.

The time complexity to build G_{BC} depends on the average branching B_{P} of the P layer and on the average branching B_{xor} of the subkey XORing layer. It amounts to $2^{2t_{\mathsf{BC}}} \cdot B_{\mathsf{xor}} \cdot B_{\mathsf{P}}$ operations. The memory cost to store G_{BC} corresponds to the number of edges e_{BC} of G_{BC} and is upper bounded by $2^{2t_{\mathsf{BC}}} \cdot B_{\mathsf{xor}} \cdot B_{\mathsf{P}}$ since one operation on step 3 adds at most one edge. We denote $\mathrm{succ}_{\mathsf{BC}}(s, k)$ the set of successors of the state s in the graph G_{BC} using the key k.

The Graph G_{KS}. It is built by simply going through all the $2^{t_{\mathsf{KS}}}$ possible key schedule internal state input truncated differences, checking which output truncated differences can be obtained through the KS layer and adding edges in G_{KS} accordingly[2]. The time and memory complexities depend on the average branching B_{KS} of the KS layer and amounts to $2^{t_{\mathsf{KS}}} \cdot B_{\mathsf{KS}}$ operations. The number of edges e_{KS} of G_{KS} equals $e_{\mathsf{KS}} = 2^{t_{\mathsf{KS}}} \cdot B_{\mathsf{KS}}$. In the sequel, we denote $\mathrm{succ}_{\mathsf{KS}}(k)$ the set of successors of the key k in graph G_{KS}.

3.3 Online Phase

The online phase finds all the shortest paths in G_r with at most $r \cdot (\frac{v_G}{2} \cdot \log(\frac{v_G}{2}) + e_G)$ computations and memory $r \cdot e_G$, thus linear in the number of rounds r. This is

[2] We assume that the key schedule is simple: given a truncated difference on the input, one can find each reachable truncated output difference in constant time. This assumption is weaker than the one from Footnote 1, and verified by most ciphers since a very complex key schedule would make the whole primitive inefficient anyway.

Algorithm 1 – Search all shortest paths in G_r.

```
function Search($G_r$)
  Copy all nodes of $G_r$ in a new graph $G_r^*$
  for all $v \in V_0,\ c(v) \leftarrow |v|$
  for all $v \in V_1, \dots, V_r,\ c(v) \leftarrow \infty$
  SortList($V_0$)
  for $i = 1 \rightarrow r$ do
    for all $v' \in V_i$, by increasing $c(v')$ do
      for all $v \in \mathrm{succ}(v')$ do
        $\alpha \leftarrow c(v') + |v|$
        if $c(v) = \infty$ then
          $c(v) \leftarrow \alpha$
          Add the edge $v' \rightarrow v$ to $G_r^*$
        else if $c(v) = \alpha$ then
          Add the edge $v' \rightarrow v$ to $G_r^*$
    SortList($V_i$)
  return $G_r^*$
```

possible because G_r is a vertex-weighted directed acyclic graph. Since the edges have a constant weight (the number of active S-Boxes, i.e. the weights, are on the nodes and not the edges), the function we want to minimize for each node $v \in V_i$, $i \in [1, r]$ is: $|v| + \min_{v' \in \mathrm{pred}(v)} (c(v'))$, where $\mathrm{pred}(v) \subseteq V_{i-1}$ is the set of all predecessors of v and $c(v')$ represents the cost of the shortest path to v'. In other words, if we know the shortest path costs to all the $v' \in V_{i-1}$, we find the shortest path to any $v \in V_i$ by choosing the predecessor of v with the minimal cost.

This can easily be done by creating a list containing all the nodes $v' \in V_{i-1}$ sorted increasingly according to the cost of their shortest path $c(v')$. Then, starting from the cheapest v' and ending to the most expensive one, we set the shortest path cost of all the successors v of v' to $|v| + c(v')$ if and only if the cost of v was not set yet (see Algorithm 1). This is an improvement over the simple shortest path computation in a directed acyclic graph using a topological order since we can take advantage of the vertex-weighted property. In practice, we iteratively build a simpler vertex-weighted directed acyclic graph G_r^* from G_r (all the nodes are the same, but with less edges), for which each node $v \in V_i$ has a cost equal to the cost of the shortest path to v in G_r, and an edge leading to $v \in V_i$ represents one of the shortest paths to v (see Figure 3). At this point, in the graph G_r^* the costs assigned to all the nodes v in V_r represent the cost of the shortest path to v in G_r. If v_G represents the number of vertices and e_G the number of edges in the graph G, then the complexity of the shortest path search is about $r \cdot (\frac{v_G}{2} \cdot \log(\frac{v_G}{2}) + e_G)$ operations: the $\frac{v_G}{2} \cdot \log(\frac{v_G}{2})$ term comes from the construction of the sorted list of the nodes at each round, and the e_G term is the number of edges visited during each round as we visit all of them. Note that this is an upper bound on the complexity since we do not need to go through all $\frac{v_G}{2}$ nodes every rounds, but only a subset of them, and we may cut some edges among all the e_G ones. The term $e_G = e_{\mathsf{BC}} + e_{\mathsf{KS}}$ dominates the complexity, and since $e_{\mathsf{BC}} >> e_{\mathsf{KS}}$, it can be approximated by the number $e_{\mathsf{BC}} \leq 2^{(n+k)/b} \times 2^{n/b}$ of edges in G_{BC}. Hence, the total time complexity is $O(r \cdot 2^{(2n+k)/b})$ for r rounds.

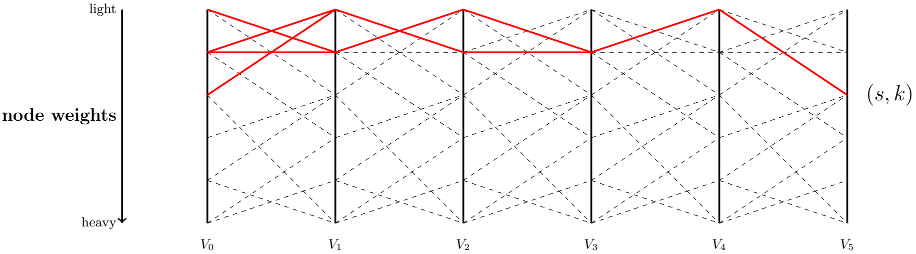


Fig. 3. The dashed edges form an example of a simplified G_5. The thick edges describe paths in the subgraph G_5^* that are shortest paths in G_5 to node (s, k). All the nodes in G_5^* are sorted according to their weight, the top being the cheapest ones.

In order to get *all* the shortest paths in G_r, we need to store at each node $v \in V_i$ not only the first shortest path found to v but all of them (lines 13 and 14 in Algorithm 1). In general, this number is very small and never exceeds the total number of shortest paths anyway. In the worst case where all paths are the shortest, it amounts to the total number of edges $r \cdot e_G$.

As explained previously, in practice we do not use the graph G directly, but two separate graphs G_{BC} and G_{KS}. We can adapt the Algorithm 1 for this setting: in order to build G_r^*, we replace the **for all** loop of line 8 that iterates over all nodes $v' = (s_i, k_i) \in V_i$ by two **for all** loops that describe all $k_{i+1} \in \text{succ}_{\mathsf{KS}}(k_i)$ and all $s_{i+1} \in \text{succ}_{\mathsf{BC}}(s_i, k_{i+1})$.

In [24], Matsui introduces an argument equivalent to the A^* optimization for path-finding or graph traversal algorithms [18] that allows to prune the majority of the edges of G and to avoid the evaluation of many sets of successors. If we know the costs c_k of all k-round characteristics, $1 \leq k \leq n-1$, and we target an n-round characteristic of cost at least c_n, then we can consider only the nodes from V_0 that have a cost at most $c_n - c_{n-1}$, and the ones in V_1 that have a cost at most $c_n - c_{n-2}$. Intuitively, after one round has been passed, we know that we paid *at least* c_1, and since there are $n-1$ remaining rounds to pass, we will need to pay *at least* c_{n-1}. In terms of intervals of costs, for each of the V_i, we only need to consider nodes that have costs in $[c_i, c_n - c_{n-i}]$, $0 \leq i \leq n$ assuming $c_0 = 0$. To take advantage of the A^* heuristic, we sort the sets of successors in both graphs, so that we can perform an extreme pruning of the edges whenever the updated costs exceed the current interval, in an early-abort manner.

Due to space constraints, we cannot detail how to efficiently extend this algorithm to the case of AES-like ciphers, so we continue directly with the consequences of the search for this class of ciphers. However, the full details are given in the extended version of our paper.

4 Applications to SPN and AES-128

4.1 Structural Evaluation of SPN AES-Like Ciphers

We present here the results on the structural evaluation of the AES-like ciphers in regard to the related-key model, which provides an estimation of the security provided by their key schedule. Namely, we ignore the semantic definition of the S-Box and the MDS matrix, and we are only interested in how they can interact in the related-key settings. The results are measured in terms of number of active S-Boxes as in [22], and presented in Table 1. Lines 2 and 3 of the table provide the minimum number of active S-Boxes (line 2) for any number of rounds when implementing an AES-like cipher, and the number of truncated characteristics that reach that bound (line 3). In these two lines, we count the number of active S-Boxes in both the state and the subkeys, whereas in lines 4 and 5 of Table 1, we consider the case of the hash function setting where the block cipher and its key schedule can be attacked somewhat independently.

Table 1. For the AES-128 cipher on r rounds, this table shows: (1) the minimal number $C_{\mathsf{KS}} + C_{\mathsf{BC}}$ of active S-Boxes in **both** the key schedule C_{KS} and in the block cipher C_{BC} achievable in truncated differential characteristics; and (2), the same figures for the minimal number $\max(C_{\mathsf{BC}}, C_{\mathsf{KS}})$ for the hash function setting. Lines 3 and 5 count the number of distinct truncated characteristics that reach that bound.

Rounds	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\mathbf{min}(C_{\mathsf{KS}} + C_{\mathsf{BC}})$	0	1	3	9	11	13	15	**21**	23	**25**
Trunc. Char. ($\log_2$)	–	4.52	6.58	10.46	5.00	13.26	16.17	21.34	14.90	21.38
$\mathbf{min}(\mathbf{max}(C_{\mathsf{BC}}, C_{\mathsf{KS}}))$	0	1	3	6	7	9	11	14	15	**17**
Trunc. Char. ($\log_2$)	–	4.00	10.00	11.73	10.00	18.92	23.64	>30	>30	>30

Theorem 2. *It is impossible to prove the security of the full* AES-128 *against related-key differential attacks without considering both the differential property of the S-Box and the* P *layer when two keys verify a certain relation. It is impossible to prove the security of the full* AES-128 *in the hash function setting without considering both the differential property of the S-Box and the* P *layer.*

Proof. First, in the case where we consider related-key differential attacks where two keys are related if their difference verify a certain relation (line 2), we remark that for 10 rounds there exists a truncated differential characteristic counting only 25 S-Boxes. As we discussed before, this means that a differential analysis would run in p_{max}^{-25} operations. Consequently, the structure of AES-128 on its own is not enough to prove the resistance to related-key attacks for any ciphers in this class, we at least need to add a criteria on the S-Box via p_{max}.

Secondly, with an S-Box on n bits ($n = 8$ in the AES), the minimal theoretical p_{max} that can be obtained is $2^{-(n-1)}$: consequently, the largest number of rounds that our structural analysis could attack for AES-like ciphers is 7 rounds. Indeed, for 7 rounds, the 15 active S-Boxes give a differential analysis requiring $p_{max}^{-15} \geq 2^{105}$ computations, which might be smaller than 2^{128}. We note that we do not know how to construct an almost-perfect permutation on n bits acting as an S-Box with $p_{max} = 2^{-(n-1)}$. The S-Box chosen in the AES implements a composition of an affine transformation on the inverse mapping, and reach $p_{max} = 2^{-(n-2)}$. Hence, the largest number of rounds that our structural analysis could attack is 8 rounds. Indeed, for 8 rounds, the 21 active S-Boxes give a differential analysis requiring $p_{max}^{-21} \geq 2^{126}$ computations, which might be smaller than 2^{128}. However, when we instantiate the P layer by the one of the AES-128, we observe that none of the $2^{16.17}$ characteristics found on 7 rounds by our search algorithm nor the $2^{21.34}$ ones for 8 rounds can be instantiated due to linear constraints coming from the key schedule. This means that proving or disproving the security of the AES-128 in the related-key setting needs to consider both the differential properties of the S-Box and the linear equations of the P layer. From an instantiated P layer, we can write a system $\mathcal{Q}$ of linear equations whose solutions are the values of all the truncated differences of the characteristic. Therefore, choosing P such that $\mathcal{Q}$ can be made inconsistent on a small number of rounds brings more security than a random P. Our tool can be used to write this system of linear equations for any truncated differential characteristic.

Finally, for 10 rounds in the hash function setting, there exists characteristics with only 17 active S-Boxes. For the AES-128, in the best case, the differential probability equals $2^{-6 \cdot 17} = 2^{-102}$. In this setting, the adversary is supposed to have full control over the input of both the key schedule and the block cipher, that is why we considered $\max(C_{\mathsf{BC}}, C_{\mathsf{KS}})$ as an objective function to minimize in our search algorithm of Section 3. As the previous structural results, this also means that one cannot prove the security of the full AES-128 against differential cryptanalysis by only analyzing its structure. ■

4.2 Differential Characteristics Results for AES-128

Theorem 3. *After 6 rounds, there is no related-key differential characteristic for* AES-128 *with a probability higher than* 2^{-128}.

Proof. The related-key differential characteristics presented in the previous section are valid only when one deals with truncated differences, and these characteristics give an indication on the structural security provided by the AES-128 key schedule. However, due to the choice of the P layer in AES-128, it turns out that none of them can be instantiated with actual differences, because of inconsistencies in some linear constraints. To overcome this difficulty, and at the cost of a bigger graph G to handle, we first add some more information in the Markov process both on the representation of the key schedule state and the internal permutation state, and we then filter the best characteristics obtained and hope to find one that can be instantiated with actual differences. Our algorithm performs a search fundamentally different from [9], but it finds again and more efficiently the same results.

By a system resolution, we show that from a truncated differential characteristic, we can decide whether it can be instantiated with actual differences, and even find an associated differential characteristic with the greatest probability. As an example, our tool found again the best 17-S-Box truncated differential characteristic on 5 rounds of AES-128, and also found how to achieve the greatest probability 2^{-105} by instantiating the differences. This has to be compared with the upper bound of $2^{-6 \cdot 17} = 2^{-102}$ given in [9] since in the best case, all the AES active S-Boxes have maximal differential probability 2^{-6}. Trying all the possible differences that instantiate this truncated differential characteristic, we show that we cannot reach that bound, but we can only set 15 out of 17 S-Boxes to the maximal differential probability. The following Table 2 reports the best related-key characteristics found by our tool on AES-128 up to 5 rounds, with their respective highest achievable probabilities. Thus, from 6 rounds, there is no related-key differential characteristic for AES-128 with a probability higher than 2^{-128}. ■

Table 2. After 6 rounds, there is no related-key differential characteristic for AES-128 with a probability higher than 2^{-128}. Our tool retrieved the previous known results but also provides the real differential characteristics with maximum probability.

Rounds	**1**	**2**	**3**	**4**	**5**
$\mathbf{min}(C_{\mathsf{KS}} + C_{\mathsf{BC}})$	0	1	5	13	17
$\mathbf{max}\,\mathbf{log_2}(p)$	0	-6	-31	-81	-105
Appendix reference	–	B	C	D	E

4.3 Distinguishing 9 Rounds of AES-128

As another application of our tool, we describe a 9-round distinguisher for AES-128 in the chosen-key model requiring 2^{55} computations and 2^{32} memory. For an ideal cipher, the same property would be detected after 2^{68} encryption queries. Here, the chosen-key model asks the adversary to find a pair of keys (k, k') satisfying $k \oplus k' = \delta$ with a known difference δ, and a pair of messages conforming to a partially instantiated characteristic in the data part.

We achieve this result by considering the best 5-round related-ley differential characteristic and propagating it backwards to reach 9 rounds. The 5 last rounds hence count 6 active S-Boxes in the key schedule part and 11 in the data part (rounds 5 to 9 in Figure 4 and Table 3). By the backward propagation in the key schedule, we reach a total of 15 active S-Boxes for the key schedule differential characteristic, whose probability equals 2^{-101}. Since we have 2^{128} possible key values, we expect 2^{27} pairs of keys to conform to the differential characteristic in the key schedule. In the block cipher part, we prepend three rounds that we plan to control with an average cost of one computation using the **Super-SBox** technique [14, 16, 23], and one more round at the very beginning that we make as sparse as possible. The entire 9-round differential characteristic is depicted on Figure 4 and in Table 3.

The Distinguishing Algorithm. Once this differential characteristic settled, we find inputs that verify the whole characteristic. We start by finding a pair of keys that conforms to the whole differential characteristic in the key schedule. There are about 2^{27} expected such pairs of keys, and we can generate them at an average cost of one computation by picking random values satisfying all the non-linear transitions and efficiently solve the linear system to retrieve all the subkeys: our implementation of that part confirms about 2^{27} are found. We note that the difference $k \oplus k' = \delta$ is already verified at this stage.

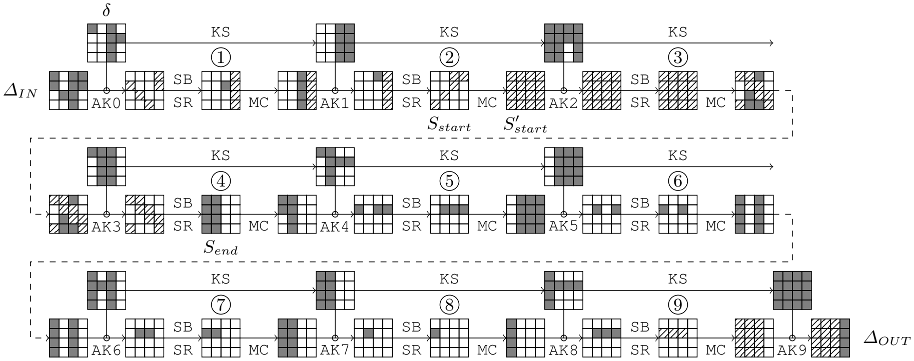


Fig. 4. Differential characteristic of 9-round AES-128 used in the distinguisher. White bytes are inactive, gray bytes have known differences, and hatched bytes are truncated differences.

Table 3. Differential characteristics used in the distinguisher of 9 rounds of AES-128. The known differences are represented by their values from 0x00 to 0xFF, and truncated differences as ??, since their values are unknown, but non-zero. The two lines for state differences are respectively the input difference after key addition and the output difference.

Round	State differences	Key differences
Plaintext	B3??0000 0000??00 28F47A?? ????0000	
1	00??0000 0000??00 000000?? ????0000 00000000 00000000 8EF47A7A ????????	B3000000 00000000 A6F47A7A 008E0000
2	00000000 00000000 28000000 ???????? ???????? ???????? ???????? ????????	00000000 00000000 A6F47A7A A67A7A7A
3	???????? ???????? ???????? ???????? ??0000?? ????7A7A 8E????7A 0000????	8E7A7A7A 8E7A7A7A 288E0000 8EF47A7A
4	??0000?? ????0000 00????00 0000???? 288E0000 8E7A7A7A 00000000 00000000	28000000 A67A7A7A 8EF47A7A 00000000
5	008E0000 00000000 008E0000 008E0000 00000000 8EF47A7A 8EF47A7A 8EF47A7A	28000000 8E7A7A7A 008E0000 008E0000
6	00000000 008E0000 00000000 008E0000 8EF47A7A 00000000 8EF47A7A 00000000	00000000 8E7A7A7A 8EF47A7A 8E7A7A7A
7	00000000 008E0000 008E0000 00000000 8EF47A7A 8EF47A7A 00000000 00000000	8EF47A7A 008E0000 8E7A7A7A 00000000
8	00000000 008E0000 00000000 00000000 8EF47A7A 00000000 00000000 00000000	8EF47A7A 8E7A7A7A 00000000 00000000
9	00000000 008E0000 008E0000 008E0000 ???????? ???????? ???????? 00000000	8EF47A7A 008E0000 008E0000 008E0000
Ciphertext	???????? ???????? ???????? 787A7A7A	78F47A7A 787A7A7A 78F47A7A 787A7A7A

For a pair of keys, we precompute the four arrays T_i containing the paired values of the ith **Super-SBox**: those are four parallel 32-bit non-linear applications obtained by reordering the layers of two rounds of the cipher. To construct the tables T_i, we iterate in parallel over the 2^{32} input values from state S_{end} that corresponds to the ith **Super-SBox** and propagate the values backwards until S'_{start}. We note that the difference in S_{end} is completely determined by our differential characteristic. We store the pair in T_i indexed by its difference[3], so that this precomputation requires 2^{32} simple operations, a memory complexity of 2^{32}, and depends on the selected pair of keys.

We continue by picking random values for the 5-byte difference after the second non-linear layer in S_{start}, which linearly fixes all the differences in S'_{start}. Note that we can repeat this part about $2^{8\cdot 5} = 2^{40}$ times. From the precomputed tables T_i, we find on average one pair of messages that verifies the middle rounds from S_{start} to S_{end}. The remaining of the process is probabilistic: backwards, we expect a fraction of 2^{-7} pairs to pass the unique specified S-Box transition in the second round up to Δ_{IN}. Forwards, we expect a fraction of $2^{-6\times 8} = 2^{-48}$ to verify the 5 last rounds up to Δ_{OUT} (all 8 transitions have been chosen by our tool to be 8 times the same one with maximal probability $p_{max} = 2^{-6}$). Finally, we expect a fraction $2^{-7-48} = 2^{-55}$ of the pairs generated in the middle to propagate correctly forwards and backwards.

By repeating this process for all 2^{40} differences in S_{start} and for 2^{15} distinct pairs of keys, we expect to find a solution for the whole characteristic in $2^{15} \cdot (2^{32} + 2^{40}) \approx 2^{55}$ operations. Note that the freedom degrees left allows to get up to 2^{12} solutions in 2^{67} operations by exhausting the remaining 2^{12} valid pairs of keys.

Ideal Case. For an ideal cipher, the adversary faces a family of random and independent permutations $\{\pi_i, i \in \{0,1\}^{128}\}$. His goal is to find a key k and a pair of messages (m, m') such that: $m \oplus m' \in \Delta_{IN}$ and $\pi_k(m) \oplus \pi_{k\oplus\delta}(m') \in \Delta_{OUT}$, where δ, Δ_{IN} and Δ_{OUT} are specified in Figure 4. Namely, δ is a completely determined 128-bit difference, whereas Δ_{IN} and Δ_{OUT} are two sets of 128-bit differences defined in Figure 4: colored and white bytes are fixed differences, while hatched bytes can take several difference values. On the output, we constrained each of the three independent active bytes after the last non-linear layer of the last round to only 127 reachable difference values (since from a fixed input difference, only 127 output differences can be reached through the AES S-Box), and the MixColumns layer being linear we have $|\Delta_{OUT}| = 127^3 \simeq 2^{21}$. On the input, 4 bytes in Δ_{IN} can take any difference value and 1 byte is constrained to only 127 reachable difference values, thus $|\Delta_{IN}| = 127 \cdot (2^8 - 1)^4 \approx 2^{39}$.

The best known method for the attacker to find (k, m, m') verifying those properties consists in applying the limited birthday algorithm [16]. The additional freedom left in choosing the key bits does not help the attacker to find the actual pair of messages that verifies the required property, since the permutations F_k and $F_{k\oplus\delta}$ have to be chosen beforehand. All in all, the attacker has access to 39 bits of differences at the input and 21 bits in the output, for a pair of permutations on $n = 128$ bits. The limited birthday distinguisher on these permutations finds a solution in time $\max\{\min(2^{IN/2}, 2^{OUT/2}), 2^{IN+OUT-n}\}$, with $IN = n - 39$ and $OUT = n - 21$, which gives a time complexity equivalent to 2^{68} encryption queries.

[3] To simplify, we assume the differences in S'_{start} to be uniformly distributed so that each 32-bit difference appears once. While this simplification is not true in practice, the cost per solution remains one on average, thus it does not change the complexity estimation.

5 Conclusion and Future Works

In this article, we have proposed a simple, efficient and generic algorithm that searches for (truncated) differential characteristics in the single-key, related-key or hash function setting for SPN ciphers. Thanks to this method, we were able to obtain the first non-trivial distinguisher of 9-round AES-128 in the chosen-key model, which has been a long-lasting open problem on this version of AES. We also show that no security proof of AES-128 in the related-key model of the hash function setting can be based only of its structure: one has to take into consideration both the Sbox and the linear layer. We believe this tool will be useful for designers that would like to easily test the security of their own key schedule or message expansion. The research community has still a lot to learn on the security of key schedules and there are many future works possible: extend the 9-round result on AES-128 to the full 10 rounds, find a single-key like security proofs in the related-key model for AES-like ciphers (generic enough to work for any dimension), provide a formal proof of security against differential/linear cryptanalysis for AES in the related-key model, and build simpler, more efficient and more secure key scheduling algorithms.

Acknowledgements. We would like to thank the Martjin Stam, Christian Rechberger and the anonyous referees for their valuable comments on our paper.

References

1. Barreto, P.S.L.M., Rijmen, V.: The Whirlpool Hashing Function. Submitted to NESSIE (September 2000) (revised May 2003), http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html (June 24, 2009)
2. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA Proposal: ECHO. Submission to NIST (updated) (2009)
3. Biham, E.: New Types of Cryptoanalytic Attacks Using related Keys (Extended Abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994)
4. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Submission to NIST, Round 2 (2009)
5. Biham, E., Dunkelman, O., Keller, N.: A Unified Approach to Related-Key Attacks. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 73–96. Springer, Heidelberg (2008)
6. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 156–171. Springer, Heidelberg (1992)
7. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: [25] 1–18
8. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)

9. Biryukov, A., Nikolić, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010)
10. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. J. Cryptology 23(4), 505–518 (2010)
11. 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)
12. Bouillaguet, C., Derbez, P., Fouque, P.-A.: Automatic Search of Attacks on Round-Reduced AES and Applications. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 169–187. Springer, Heidelberg (2011)
13. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
14. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
15. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – a SHA-3 candidate. Submission to NIST, Round 3 (2011)
16. Gilbert, H., Peyrin, T.: Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010)
17. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
18. Hart, P., Nilsson, N., Raphael, B.: A Formal Basis For The Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems, Science, and Cybernetics SSC-4(2), 100–107 (1968)
19. ISO: ISO/IEC 10118-3:2004: Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions (February 2004)
20. Khovratovich, D., Biryukov, A., Nikolic, I.: Speeding up Collision Search for Byte-Oriented Hash Functions. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 164–181. Springer, Heidelberg (2009)
21. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
22. Knudsen, L.R., Rijmen, V.: Known-Key Distinguishers for Some Block Ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
23. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full whirlpool compression function. In: [25] 126–143
24. Matsui, M.: On Correlation Between the Order of S-boxes and the Strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
25. Matsui, M. (ed.): ASIACRYPT 2009. LNCS, vol. 5912. Springer, Heidelberg (2009)

26. Miles, E., Viola, E.: Substitution-permutation networks, pseudorandom functions, and natural proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 68–85. Springer, Heidelberg (2012)
27. National Institute for Science, Technology (NIST): Advanced Encryption Standard (FIPS PUB 197) (November 2001)
28. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
29. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Best Truncated Differential Characteristics for AES-128

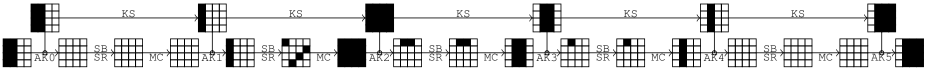


Fig. 5. Best truncated differential characteristics for AES-128 when $r = 5$ rounds with 11 active Sboxes

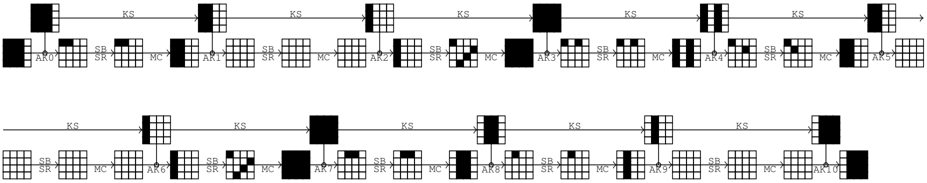


Fig. 6. Best truncated differential characteristics for AES-128 when $r = 10$ rounds with 25 active Sboxes

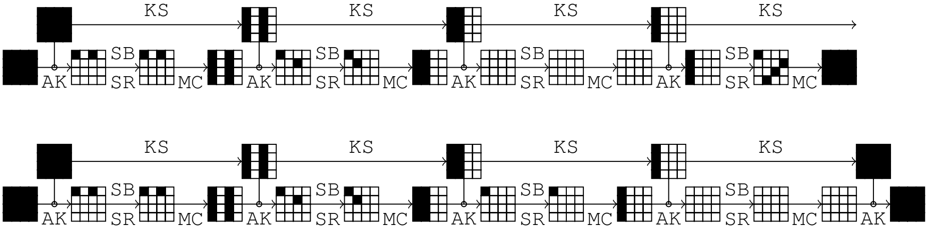


Fig. 7. Best truncated differential characteristics for AES-128 when $r = 8$ rounds with 21 active Sboxes

B Differential Characteristic for 2-Round AES-128

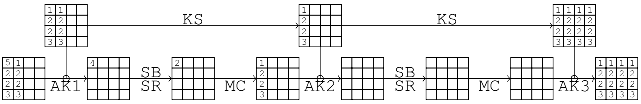


Fig. 8. The best differential characteristic on two rounds of AES-128, which has a probability $p = 2^{-6}$. The vector of differences can take as many as 2^8 values, and for instance: (1,...,5)=(0x1C,0x0E,0x12,0x01,0x1D).

C Differential Characteristic for 3-Round AES-128

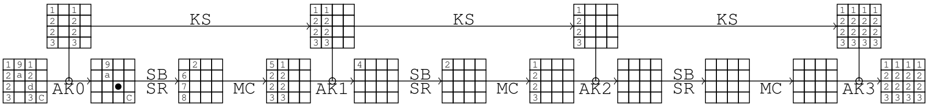


Fig. 9. The best differential characteristic on three rounds of AES-128, which has a probability $p = 2^{-31}$. The vector of differences is (1,...,d)=(0x1C,0x0E,0x12,0x01,0x1D,0x90,0x0D,0x0B,0xB3,0x58, 0x45,0xF7,0x4B).

D Differential Characteristic for 4-Round AES-128

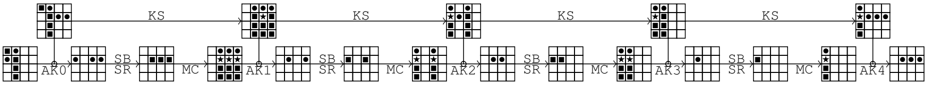


Fig. 10. The first best differential characteristic on four rounds of AES-128, which has a probability $p = 2^{-81}$. Differences are: ■ = 0x7A, ● = 0x8E and ★ = ■ ⊕ ● = 0xF4.

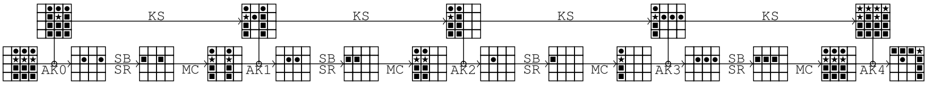


Fig. 11. The second best differential characteristic on four rounds of AES-128, which has a probability $p = 2^{-81}$. Differences are: ■ = 0x7A, ● = 0x8E and ★ = ■ ⊕ ● = 0xF4.

E Differential Characteristic for 5-Round AES-128

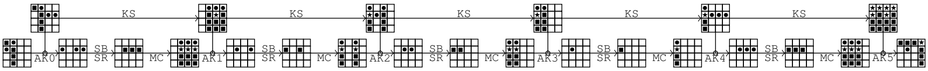


Fig. 12. The best differential characteristic on five rounds of AES-128, which has a probability $p = 2^{-105}$. Differences are: ■ = 0x7A, ● = 0x8E and ★ = ■ ⊕ ● = 0xF4.