Multiple Limited-Birthday Distinguishers and Applications

Jérémy Jean[1], María Naya-Plasencia[2], and Thomas Peyrin[3](✉)

[1] École Normale Supérieure, Paris, France
Jeremy.Jean@ens.fr
[2] SECRET Project-Team - INRIA Paris-Rocquencourt, Paris, France
[3] Nanyang Technological University, Singapore, Singapore
thomas.peyrin@gmail.com

Abstract. In this article, we propose a new improvement of the rebound techniques, used for cryptanalyzing AES-like permutations during the past years. Our improvement, that allows to reduce the complexity of the attacks, increases the probability of the outbound part by considering a new type of differential paths. Moreover, we propose a new type of distinguisher, the multiple limited-birthday problem, based on the limited-birthday one, but where differences on the input and on the output might have randomized positions. We also discuss the generic complexity for solving this problem and provide a lower bound of it as well as we propose an efficient and generic algorithm for solving it. Our advances lead to improved distinguishing or collision results for many AES-based functions such as AES, ECHO, Grøstl, LED, PHOTON and Whirlpool.

Keywords: AES-like permutation · Distinguishers · Limited-birthday · Rebound attack

1 Introduction

On October the 2nd of 2012, the NIST chose Keccak [4] as the winner of the SHA-3 hash function competition. This competition started on 2008, and received 64 submissions. Amongst them, 56 passed to the first round, 14 to the second and 5 to the final on December 2010. Through all these years, a large amount of cryptanalysis has been published on the different candidates and new techniques have been proposed. One of the new techniques that can be fairly considered as among the most largely applied to the different candidates is the rebound

Jérémy Jean is supported by the French Agence Nationale de la Recherche through the SAPHIR2 project under Contract ANR-08-VERS-014 and by the French *Délégation Générale pour l'Armement* (DGA).

María Naya-Plasencia is partially supported by the French Agence Nationale de la Recherche through the BLOC project under Contract ANR-11-INSE-0011.

Thomas Peyrin is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

T. Lange, K. Lauter, and P. Lisoněk (Eds.): SAC 2013, LNCS 8282, pp. 533–550, 2014.
DOI: 10.1007/978-3-662-43414-7_27,

attack. Presented in [23], at first for analyzing AES-like compression functions, it has found many more applications afterwards.

Rebound attacks is a freedom degrees utilization method, and, as such, it aims at finding solutions for a differential characteristic faster than the probabilistic approach. The characteristic is divided in two parts: a middle one, called inbound, and both remaining sides, called outbound. In the inbound phase, the expensive part of the characteristic, like one fully active AES state around the non-linear transformation, is considered. The rebound technique allows to find many solutions for this part with an average cost of one. These solutions are then exhausted probabilistically forwards and backwards through the outbound part to find one out of them that conforms to the whole characteristic.

Several improvements have appeared through the new analyses, like start-from-the-middle attack [22] or Super-SBoxes [13,19], which allow to control three rounds in the middle, multinbounds [21] which extend the number of rounds analyzed by a better use of the freedom degrees (better ways of merging the inbounds were proposed in [24]), or non-fully-active states [27] that permits to reduce the complexity of the outbound part. In [17], a method for controlling four rounds in the middle with high complexity was proposed, and it allows to reach a total of 9 rounds with regards to distinguishers in the case of a large permutation size.

This class of attacks is interesting mostly for hash functions, because they require the attacker to be able to know and to control the internal state of the primitive, which is not possible if a secret is involved, for example in a block cipher. Yet, another application is the study of block ciphers in the so-called known-key or chosen-key models, where the attacker knows or even has full control of the secret key. These models were recently made popular because many SHA-3 or new hash functions are based on block ciphers or fixed-key permutations, and also one may want to be sure that a cipher has no flaw whatsoever, even in weaker security models.

Various types of attacks are possible for hash functions, such as collision and (second) preimage search, or even distinguishers. Indeed, hash functions being often utilized to mimic the behavior of random oracles [7] in security protocols, e.g. RSA-OAEP [2], it is important to ensure that no special property can be observed that allows an attacker to distinguish the primitive from a random oracle. Distinguishers on hash functions, compression functions or permutations can be very diverse, from classical differential distinguishers (limited-birthday [13] or subspace [20]) to rotational [18] or zero-sum distinguishers [6]. In any case, for the distinguisher to be valid, the cryptanalyst has to compare the cost of finding the specific property for the function analyzed and for an ideal primitive. The bounds compared in this article refer to the computational bounds, and not information-theoretic bounds.

Rebound-like techniques are well adapted for various types of distinguishers and it remains an open problem to know how far (and with what complexity) they can be pushed further to attack AES-like permutations and hash/compression functions. So far, the best results could reach 8 or 9 rounds, depending on the size of the permutation attacked.

Our Contributions. In this paper, we propose a new improvement of the previous rebound techniques, reducing the complexity of known differential distinguishers and by a lower extend, reducing some collision attack complexities. We observed that the gap between the distinguisher complexity and the generic case is often big and some conditions might be relaxed in order to minimize as much as possible the overall complexity. The main idea is to generalize the various rebound techniques and to relax some of the input and output conditions of the differential distinguishers. That is, instead of considering pre-specified active cells in the input and output (generally full columns or diagonals), we consider several possible position combinations of these cells. In some way, this idea is related to the outbound difference randomization that was proposed in [11] for a rebound attack on `Keccak`, a non-`AES`-like function. Yet, in [11], the randomization was not used to reduce the attack complexity, but to provide enough freedom degrees to perform the attack.

As this improvement affects directly the properties of the inputs and outputs, we now have to deal with a new differential property observed and we named this new problem the *multiple limited-birthday problem* (LBP), which is more general than the limited-birthday one. A very important question arising next is: what is the complexity of the best generic algorithm for obtaining such set of inputs/outputs? For previous distinguishers, where the active input and output columns were fixed, the limited-birthday algorithm [13] is yet the best one for solving the problem in the generic case. Now, the multiple limited-birthday is more complex, and in Sect. 3.3 we discuss how to bound the complexity of the best generic distinguisher. Moreover, we also propose an efficient, generic and non-trivial algorithm in order to solve the multiple limited-birthday problem, providing the best known complexity for solving this problem.

Finally, we generalize the various rebound-like techniques in Sect. 4 and we apply our findings on various `AES`-like primitives. Due to space constraints, Sect. 5 presents our main results, while the full results are detailed in the extended version of our paper. Our main results dealing with `AES` [9] and `Whirlpool` [1] are summarized and compared to previous works in Table 1. In the full version, we also derive results on `ECHO` [3], `Grøstl` [12], `LED` [15], `PHOTON` [14], that are reported in Appendix A.

2 `AES`-like Permutations

We define an `AES`-like permutation as a permutation that applies N_r rounds of a round function to update an internal state viewed as a square matrix of t rows and t columns, where each of the t^2 cells has a size of c bits. We denote $\mathcal{S}$ the set of all theses states: $|\mathcal{S}| = 2^{ct^2}$. This generic view captures various permutations in cryptographic primitives such as `AES`, `ECHO`, `Grøstl`, `LED`, `PHOTON` and `Whirlpool`.

The round function (Fig. 1) starts by xoring a round-dependent constant to the state in the **AddRoundConstant** operation (AC). Then, it applies a substitution layer **SubBytes** (SB) which relies on a $c \times c$ non-linear bijective S-box S. Finally, the round function performs a linear layer, composed of the

Table 1. Known and improved results for three rebound-based attacks on AES-based primitives.

Target	Subtarget	Rounds	Type	Time	Memory	Ideal	Reference
AES-128	Cipher	8	KK dist.	2^{48}	2^{32}	2^{65}	[13]
		8	KK dist.	2^{44}	2^{32}	2^{61}	Sect. 5.1
		8	CK dist.	2^{24}	2^{16}	2^{65}	[10]
		8	CK dist.	$2^{13.4}$	2^{16}	$2^{31.7}$	Sect. 5.1
AES-128	DM-mode	5	CF collision	2^{56}	2^{32}	2^{65}	[22]
		6	CF collision	2^{32}	2^{16}	2^{65}	Sect. 5.1
Whirlpool	CF	10	dist.	2^{176}	2^{8}	2^{384}	[20]
		10	dist.	$2^{115.7}$	2^{8}	2^{125}	Sect. 5.2
Whirlpool	Permutation	7.5	collision	2^{184}	2^{8}	2^{256}	[20]
		7.5	collision	2^{176}	2^{8}	2^{256}	Sect. 5.2
Whirlpool	Hash func.	5.5	collision	2^{184}	2^{8}	2^{256}	[20]
		5.5	collision	2^{176}	2^{8}	2^{256}	Sect. 5.2

ShiftRows transformation (SR), that moves each cell belonging to the x-th row by x positions to the left in its own row, and the **MixCells** operation (MC), that linearly mixes all the columns of the matrix separately by multiplying each one with a matrix **M** implementing a Maximum Distance Separable (MDS) code, which provides diffusion.

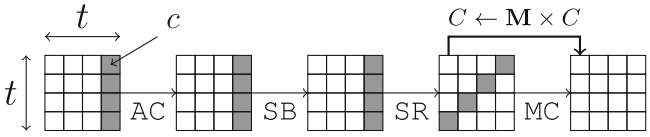


Fig. 1. One round of the AES-like permutation instantiated with $t = 4$.

Note that this description encompasses permutations that really follow the AES design strategy, but very similar designs (for example with a slightly modified **ShiftRows** function or with a **MixCells** layer not implemented with an MDS matrix) are likely to be attacked by our techniques as well. In the case of AES-like block ciphers analyzed in the known/chosen-key model, the subkeys generated by the key schedule are incorporated into the known constant addition layer **AddRoundConstant**.

3 Multiple Limited-Birthday Distinguisher

In this section, we present a new type of distinguisher: the multiple limited-birthday (Sect. 3.3). It is inspired from the limited-birthday one that we recall in Sect. 3.2, where some of the input and output conditions are relaxed. We discuss how to bound the complexity of the best generic algorithm for solving this

problem, as well as we provide an efficient algorithm solving the problem with the best known complexity. Due to the keyless particularity of the primitives, we precise the relevance of distinguishers in that context.

3.1 Structural Distinguishers

We precise here what we consider to be a distinguishing algorithm for a keyless primitives. Let F be a primitive analyzed in the open-key model (either known- or chosen-key). In that context, there is no secret: F could be for instance a hash function or a block cipher where the key is placed into the public domain.

To formalize the problem, we say that the goal of the adversary is to validate a certain property P on the primitive F. For example, if F is a hash function, P could be "find two *different* inputs x, x' such that $F(x) = F(x')$" to capture the collision property. One other example, more related to our approach, would be $P = LPB$, the limited-birthday problem. In that sense, limited-birthday, collision and other similar problems are all particular kinds of distinguishers.

It is easy to see that when no random challenge is input to the adversary (like for collision definition for example) there always exists (at least) one algorithm that outputs a solution to P in constant time and without any query to F. We do not know this algorithm, but its existence can be proven. The main consequence about this argument is the lower bound on the number of queries Q of the distinguishing algorithm. Indeed, because of that algorithm, we have $0 \leq Q$. Therefore, we cannot reach any security notion in that context.

Now, we can circumvent this problem by introducing a challenge C to the problem P, that is, we *force* the distinguishing algorithm to use some value it does not know beforehand. To ease the formal description, one can think of an adversarial model where the memory is restricted to a fixed and constant amount M. That way, we get rid of the trivial (but unknown) algorithms that return a solution to P in constant time, since they do not know the parameter/challenge C. More precisely, if it does return a solution in constant time, then it is a wrong one with overwhelming probability, such that its winning advantage is nearly zero. Consequently, reasonable winning advantages are reached by getting rid of all those trivial algorithms. Then, the lower bound increases and becomes dependent of the size of C.

As an example, a challenge C could be an particular instantiation of the S-Box used in the primitive F. One could say that C selects a particular primitive F in a space of structurally-equivalent primitives, and asks the adversary to solve P on that particular instance F.

In all the published literature, the distinguishers in the open-key model do not consider any particular challenges, and they also ignore the trivial algorithms. From a structural point of view, there is no problem in doing so since we know that those distinguishers would also work if we were to introduce a challenge. But formally, these are not proper distinguishers because of the constant time algorithms that make the lower bound $0 \leq Q$. In this article, we do not claim to have strong distinguishers in the theoretical sense, but we provide structural

distinguishing algorithms in the same vein as all the previously published results (q-multicollision, k-sum, limited-birthday, etc.).

3.2 Limited-Birthday

In this section, we briefly recall the limited-birthday problem and the best known algorithm for solving it. As described in Sect. 3.1, to obtain a fair comparison of algorithms solving this structural problem, we ignore the trivial algorithms mentioned. That way, we can stick to structural distinguishers and compare their time complexities to measure efficiency.

Following the notations of the previous section, the limited-birthday problem consists in obtaining a pair of inputs (x, x') (each of size n) to a permutation F with a truncated difference $x \oplus x'$ on $\log_2(IN)$ predetermined bits, that generates a pair of outputs with a truncated difference $F(x) \oplus F(x')$ on $\log_2(OUT)$ predetermined bits (therefore IN and OUT represent the set size of the admissible differences on the input and on the output respectively).

The best known cost for obtaining such a pair for an ideal permutation is denoted by $C(IN, OUT)$ and, as described in [13], can be computed the following way:

$$C(IN, OUT) = \max\left\{ \min\left\{ \sqrt{2^n/IN}, \sqrt{2^n/OUT} \right\}, \frac{2^{n+1}}{IN \cdot OUT} \right\}. \quad (1)$$

The main differences with the subspace distinguisher [20] is that in the limited-birthday distinguisher both input and output are constrained (thus limiting the ability of the attacker to perform a birthday strategy), and only a single pair is to be exhibited.

3.3 Multiple Limited-Birthday and Generic Complexity

We now consider the distinguisher represented in Fig. 2, where the conditions regarding previous distinguishers have been relaxed: the number of active diagonals (resp. anti-diagonals) in the input (resp. output) is fixed, but their positions are not. Therefore, we have $\binom{t}{n_B}$ possible different configurations in the input and $\binom{t}{n_F}$ in the output. We state the following problem.

Problem 1 (Multiple limited-birthday). Let $n_F, n_B \in \{1, \ldots, t\}$, F a permutation from the symmetric group $\mathfrak{S}_{\mathcal{S}}$ of all permutations on $\mathcal{S}$, and Δ_{IN} be the set of truncated patterns containing all the $\binom{t}{n_B}$ possible ways to choose n_B active diagonals among the t ones. Let Δ_{OUT} defined similarly with n_F active anti-diagonals. Given F, Δ_{IN} and Δ_{OUT}, the problem asks to find a pair $(m, m') \in \mathcal{S}^2$ of inputs to F such that $m \oplus m' \in \Delta_{IN}$ and $F(m) \oplus F(m') \in \Delta_{OUT}$.

As for the limited-birthday distinguisher, we do not consider this problem in the theoretical sense, as there would be a trivial algorithm solving it (see Sect. 3.1). Therefore, and rather than introducing a challenge that would confuse

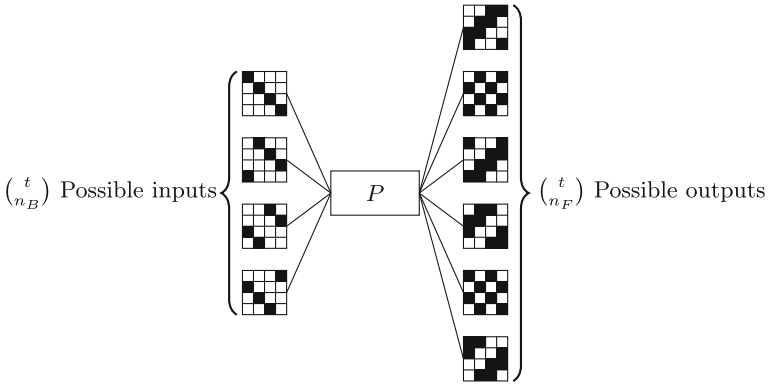


Fig. 2. Possible inputs and outputs of the relaxed generic distinguisher. The blackbox P implements a random permutation uniformly drawn from $\mathfrak{S}_{\mathcal{S}}$. The figure shows the case $t = 4$, $n_B = 1$ and $n_F = 2$.

the description of our algorithm, we are interested in structural distinguishing algorithms, that ignore the constant-time trivial algorithms. Following notations of the previous section, the permutation defined in Problem 1 refer to the general primitive F of Sect. 3.1 and the particular property P the adversary is required to fulfill on P has been detailed in the problem definition.

We conjecture that the best generic algorithm for finding one solution to Problem 1 has a time complexity that is lower bounded by the limited-birthday algorithm when considering $\underline{IN} = \binom{t}{n_B} 2^{t \cdot c \cdot n_B}$ and $\underline{OUT} = \binom{t}{n_F} 2^{t \cdot c \cdot n_F}$. This can be reasonably argued as we can transform the multiple limited-birthday algorithm into a similar (but not equivalent) limited-birthday one, with a size of all the possible truncated input and output differences of $\underline{IN}$ and $\underline{OUT}$ respectively. Solving the similar limited-birthday problem requires a complexity of $C(\underline{IN}, \underline{OUT})$, but solving the original multiple limited-birthday problem would require an equal or higher complexity, as though having the same possible input and output difference sizes, for the same number of inputs (or outputs), the number of valid input pairs that can be built might be lower. This is directly reflected on the complexity of solving the problem, as in the limited-birthday algorithm, it is considered that for 2^n inputs queried, we can build 2^{2n-1} valid input pairs. The optimal algorithm solving Problem 1 would have a time complexity T such that: $C(\underline{IN}, \underline{OUT}) \leq T$.

We have just provided a lower bound for the complexity of solving Problem 1 in the ideal case, but an efficient generic algorithm was not known. For finding a solution, we could repeat the algorithm for solving the limited-birthday while considering sets of input or output differences that do not overlap, with a complexity of $\min\{C(\overline{IN}, \underline{OUT}), C(\underline{IN}, \overline{OUT})\}$, where $\overline{IN} = 2^{t \cdot c \cdot n_B}$, $\overline{OUT} = 2^{t \cdot c \cdot n_F}$, $\underline{IN} = \binom{t}{n_B} 2^{t \cdot c \cdot n_B}$ and $\underline{OUT} = \binom{t}{n_F} 2^{t \cdot c \cdot n_F}$.

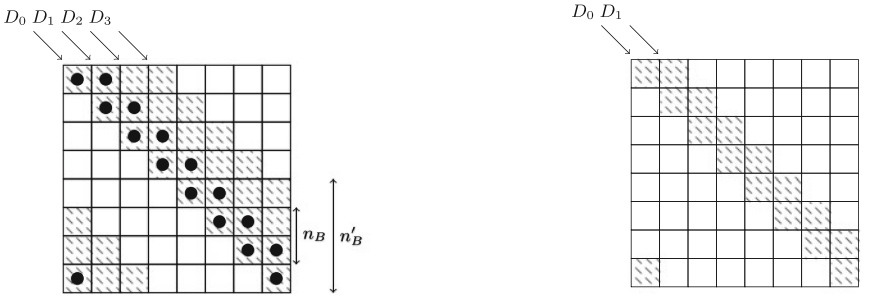
We propose in the sequel a new generic algorithm to solve Problem 1 whose time complexity verifies the claimed bound and improves the complexity of the

algorithm previously sketched. It allows then to find solutions faster than previous algorithms, as detailed in Table 2. Without loss of generality, because the problem is completely symmetrical, we explain the procedure in the forward direction. The same reasoning applies for the backward direction, when changing the roles between input and output of the permutation, and the complexity would then be the lowest one.

From Problem 1, we see that a random pair of inputs have a probability $P_{out} = \binom{t}{n_F} 2^{-t(t-n_F)c}$ to verify the output condition. We therefore need at least P_{out}^{-1} input pairs so that one verifying the input and output conditions can be found. The first goal of the procedure consists in constructing a structure containing enough input pairs.

Structures of Input Data. We want to generate the amount of valid input pairs previously determined, and we want do this while minimizing the numbers of queries performed to the encryption oracle, as the complexity directly depends on them. A natural way to obtain pairs of inputs consists in packing the data into structured sets. These structures contain all 2^{ct} possible values on n'_B different diagonals at the input, and make the data complexity equivalent to $2^{n'_B ct}$ encryptions. If there exists $n'_B \leq n_B$ such that the number N of possible pairs $\binom{2^{n'_B ct}}{2}$ we can construct within the structure verifies $N \geq P_{out}^{-1}$, then Problem 1 can be solved easily by using the birthday algorithm. If this does not hold, we need to consider a structure with $n'_B > n_B$. In this case, we can construct as many as $\binom{n'_B}{n_B} 2^{(n'_B - n_B)tc} \binom{2^{n_B tc}}{2}$ pairs (m, m') of inputs such that $m \oplus m'$ already belongs to Δ_{IN}. We now propose an algorithm that handles this case.

We show how to build a fixed number of pairs with the smallest structure that we could find, and we conjecture that the construction is optimal in the sense this structure is the smallest possible. The structure of input data considers n'_B



(a) Structure. (b) Example of pair.

Fig. 3. Structure of input data: example with $n_B = 2$ and $n'_B = 4$. We construct a pair with n_B active diagonals like (b) from the structure depitected on (a). Hatched cells are active, so that the structure allows to select $\binom{n'_b}{n_B}$ different patterns to form the pairs (one is represented by the bullets •).

diagonals $D_1, \ldots, D_{n'_B}$ assuming all the 2^{ct} possible values, and an extra diagonal D_0 assuming $2^y < 2^{ct}$ values (see Fig. 3). In total, the number of queries equals $2^{y+n'_B tc}$. Within this structure, we can get[1] a number of pairs parameterized by n'_B and y:

$$N_{pairs}(n'_B, y) := \binom{n'_B}{n_B}\binom{2^{n_B ct}}{2} 2^y \, 2^{(n'_B - n_B)tc} + \binom{n'_B}{n_B - 1}\binom{2^{y+(n_B-1)ct}}{2} 2^{(n'_B-(n_B-1))ct}. \quad (2)$$

The first term of the sum considers the pairs generated from n_B diagonals among the $D_1, \ldots, D_{n'_B}$ diagonals, while the second term considers D_0 and $n_B - 1$ of the other diagonals. The problem of finding an algorithm with the smallest time complexity is therefore reduced to finding the smallest n'_B and the associated y so that $N_{pairs}(n'_B, y) = P_{out}^{-1}$. Depending on the considered scenarios, P_{out}^{-1} would have different values, but finding (n'_B, y) such that $N_{pairs}(n'_B, y) = P_{out}^{-1}$ can easily be done by an intelligent search in $\log(t) + \log(ct)$ simple operations by trying different parameters until the ones that generate the wanted amount of pairs P_{out}^{-1} are found.

Generic Algorithm. Once we have found the good parameters n'_B and y, we generate the $2^{y+n'_B ct}$ inputs as previously described, and query their corresponding outputs to the permutation F. We store the input/output pairs in a table ordered by the output values. Assuming they are uniformly distributed, there exists a pair in this table satisfying the input and output properties from Problem 1 with probability close to 1.

To find it, we first check for each output if a matching output exists in the list. When this is the case, we next check if the found pair also verifies the input conditions. The time complexity of this algorithms therefore costs about $2^{y+n'_B ct} + 2^{2y+2n'_B tc} P_{out}$ operations. The first term in the sum is the number of outputs in the table: we check for each one of them if a match exists at cost about one. The second term is the number of output matches that we expect to find, for which we also test if the input patterns conform to the wanted ones.

Finally, from the expression of P_{out}, we approximate the time complexity $2^{y+n'_B ct} + 2^{2y+2n'_B tc} P_{out}$ to $2^{y+n'_B ct}$ operations, as the second term is always smaller than the first one. The memory complexity if we store the table would be $2^{y+n'_B ct}$ as well, but we can actually perform this research without memory, as in practice what we are doing is a collision search. In Table 2, we show some examples of different complexities achieved by the bounds proposed and by our algorithm.

[1] When $y = 0$, we compute the number of terms as $N_{pairs}(n'_B, 0) := \binom{n'_B}{n_B}\binom{2^{n_B ct}}{2} 2^{(n'_B - n_B)tc}$.

Table 2. Examples of time complexities for several algorithms solving the multiple limited-birthday problem.

Parameters (t, c, n_B, n_F)	Bound: $C(\underline{IN}, \underline{OUT})$	Our algorithm	$C(\overline{IN}, \underline{OUT})$
(8,8,1,1)	2^{379}	$2^{379.7}$	2^{382}
(8,8,1,2)	$2^{313.2}$	$2^{314.2}$	$2^{316.2}$
(8,8,2,2)	$2^{248.4}$	$2^{250.6}$	$2^{253.2}$
(8,8,1,3)	$2^{248.19}$	$2^{249.65}$	$2^{251.19}$
(4,8,1,1)	2^{61}	$2^{62.6}$	2^{63}
(4,4,1,1)	2^{29}	$2^{30.6}$	2^{31}

4 Truncated Characteristic with Relaxed Conditions

In this section, we present a representative 9-round example of our new distinguisher.

4.1 Relaxed 9-round Distinguisher for AES-like Permutation

We show how to build a 9-round distinguisher when including the idea of relaxing the input and output conditions. In fact, this new improvement allows to reduce the complexity of the distinguisher, as the probability of verifying the outbound is higher. We point out here that we have chosen to provide an example for 9 rounds as it is the distinguisher that reaches the highest number of rounds, solving three fully-active states in the middle. We also recall that for a smaller number of rounds, the only difference with the presented distinguisher is the complexity $C_{inbound}$ for the inbound part, that can be solved using already well-known methods such as rebound attacks, Super-SBoxes or start-from-the-middle, depending on the particular situation that we have. For the sake of simplicity, in the end of this section, we provide the complexity of the distinguisher depending on the inbound complexity $C_{inbound}$.

In the end of the section, we compare our distinguisher with the previously explained best known generic algorithm to find pairs conforming to those cases. We show how the complexities of our distinguisher are still lower than the lowest bound for such a generic case.

Following the notations from [17], we parameterize the truncated differential characteristic by four variables (see Fig. 4) such that trade-offs are possible by finding the right values for each one of them. Namely, we denote c the size of the cells, $t \times t$ the size of the state matrix, n_B the number of active diagonals in the input (alternatively, the number of active cells in the second round), n_F the number of active independent diagonals in the output (alternatively, the number of active cells in the eighth round), m_B the number of active cells in the third round and m_F the number of active cells in the seventh round.

Hence, the sequence of active cells in the truncated differential characteristic becomes:

$$t\,n_B \xrightarrow{R_1} n_B \xrightarrow{R_2} m_B \xrightarrow{R_3} t\,m_B \xrightarrow{R_4} t^2 \xrightarrow{R_5} t\,m_F \xrightarrow{R_6} m_F \xrightarrow{R_7} n_F \xrightarrow{R_8} t\,n_F \xrightarrow{R_9} t^2, \quad (3)$$

with the constraints $n_F + m_F \geq t + 1$ and $n_B + m_B \geq t + 1$ that come from the MDS property, and relaxation conditions on the input and output, meaning that the positions of the n_B input active diagonals, and of the n_F active anti-diagonals generating the output can take any possible configuration, and not a fixed one. This allows to increase the probability of the outbound part and the number of solutions conforming to the characteristic. This is reflected in a reduction of the complexity of the distinguisher. The amount of solutions that we can now generate for the differential path equals to ($\log_2$):

$$\begin{aligned} &\log_2\left(\binom{t}{n_B}\binom{t}{n_F}\right) + ct^2 + ctn_B \\ &\quad - c(t-1)n_B - c(t-m_B) - ct(t-m_F) - c(t-1)m_F - c(t-n_F) \qquad (4) \\ &= c(n_B + n_F + m_B + m_F - 2t) + \log_2\left(\binom{t}{n_B}\binom{t}{n_F}\right). \end{aligned}$$

If follows from the MDS constraints that there are always at least $\binom{t}{n_B}\binom{t}{n_F}2^{2c}$ freedom degrees, independently of t.

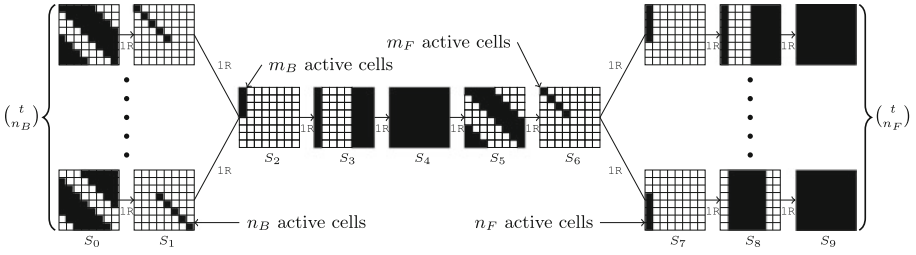


Fig. 4. The 9-round truncated differential characteristic used to distinguish an AES-like permutation from an ideal permutation. The figure shows some particular values: $t = 8$, $n_B = 5$, $m_B = 4$, $m_F = 4$ and $n_F = 5$.

To find a conforming pair we use the algorithm proposed in [17] for solving the inbound part and finding a solution for the middle rounds. The cost of those uncontrolled rounds is given by:

$$C_{outbound} := \frac{2^{c(t-n_B)}}{\binom{t}{n_B}} \cdot \frac{2^{c(t-n_F)}}{\binom{t}{n_F}} = \frac{2^{c(2t-n_B-n_F)}}{\binom{t}{n_B}\binom{t}{n_F}}, \qquad (5)$$

since we need to pass one $n_B \leftarrow m_B$ transition in the backward direction with $\binom{t}{n_B}$ possibilities and one $m_F \rightarrow n_F$ transition in the forward direction with $\binom{t}{n_F}$ possibilities.

4.2 Comparison with Ideal Case

As we discussed in Sect. 3.3, in the ideal case, the generic complexity T is bounded by $C(\underline{IN}, \underline{OUT}) \leq T \leq \min\left\{C(\underline{IN}, \overline{OUT}), C(\overline{IN}, \underline{OUT})\right\}$, where we have $\underline{IN} = \binom{t}{n_B} 2^{t \cdot c \cdot n_B}$, $\underline{OUT} = \binom{t}{n_F} 2^{t \cdot c \cdot n_F}$, $\overline{IN} = 2^{t \cdot c \cdot n_B}$ and $\overline{OUT} = 2^{t \cdot c \cdot n_F}$.

We proposed the algorithm with the best known complexity for solving the problem in the ideal case in Sect. 3.3, for being sure that our distinguishers have smaller complexity than the best generic algorithm, we compare our complexities with the inferior bound given: $C(\underline{IN}, \underline{OUT})$, so that we are sure that our distinguisher is a valid one. We note that the algorithm we propose gives a distinguisher for 9 rounds of an AES-like permutation as soon as the state verifies $t \geq 8$.

We recall here that the complexity of the distinguishers that we build varies depending on the number of rounds solved in the middle, or the parameters chosen, and we provide some examples of improvements of previous distinguishers and their comparisons with the general bounds and algorithms in the next section.

5 Applications

In this section, we apply our new techniques to improve the best known results on various primitives using AES-like permutations. Due to a lack of space, we do not describe the algorithms in details, and refer to their respective specification documents for a complete description. When we randomize the input/output differences positions, the generic complexities that we compare with are the ones coming from the classical limited-birthday problem $C(\underline{IN}, \underline{OUT})$ (updated with the right amount of differences), since they lower bound the corresponding multiple limited-birthday problem.

5.1 AES

AES-128 [9] is an obvious target for our techniques, and it is composed of 10 rounds and has parameters $t = 4$ and $c = 8$.

Distinguisher. The current best distinguishers (except the biclique technique [5] which allows to do a speed-up search of the key by a factor of 0.27 for the full AES) can reach 8 rounds with 2^{48} computations in the known-key model (see [13]) and with 2^{24} computations in the chosen-key model (see [10]). By relaxing some input/output conditions, we are able to obtain a 8-round distinguisher with 2^{44} computations in the known-key model and with $2^{13.4}$ computations in the chosen-key model.

In the case of the known-key distinguisher, we start with the 8-round differential characteristic depicted in Fig. 5. One can see that it is possible to randomize the position of the unique active byte in both states S_1 and S_6, resulting in 4 possibles positions for both the input and output differences. We reuse the

Super-SBox technique that can find solutions from state S_2 to state S_5 with a single operation on average. Then, one has to pay $2^{24}/4 = 2^{22}$ for both transitions from state S_2 to S_1 backward and from state S_5 to S_6 forward, for a total complexity of 2^{44} computations. In the ideal case, our multiple limited-birthday problem gives us a generic complexity bounded by 2^7.

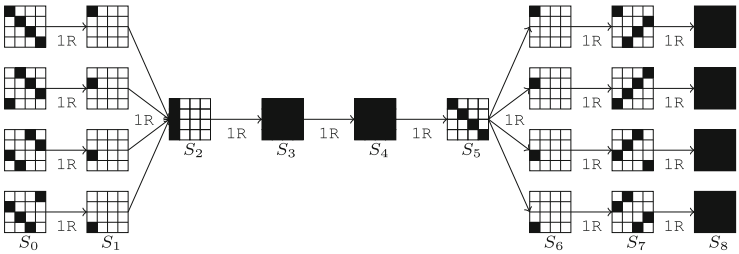


Fig. 5. Differential characteristic for the 8-round known-key distinguisher for `AES-128`

Concerning the chosen-key distinguisher, we start with the 8-round differential characteristic depicted in Fig. 6. Here, we use the technique introduced in [10] that can find solutions from state S_2 to state S_6 with a single operation on average. It is therefore not possible to randomize the position of the unique active byte in state S_6 since it is already specified. However, for the transition from state S_2 to S_1, we let two active bytes to be present in S_2, with random positions (6 possible choices). This happens with a probability $6 \cdot 2^{-16}$ and the total complexity to find a solution for the entire characteristic is $2^{13.4}$ computations. In the ideal case, our multiple limited-birthday problem gives us a generic complexity bounded by $2^{31.7}$.

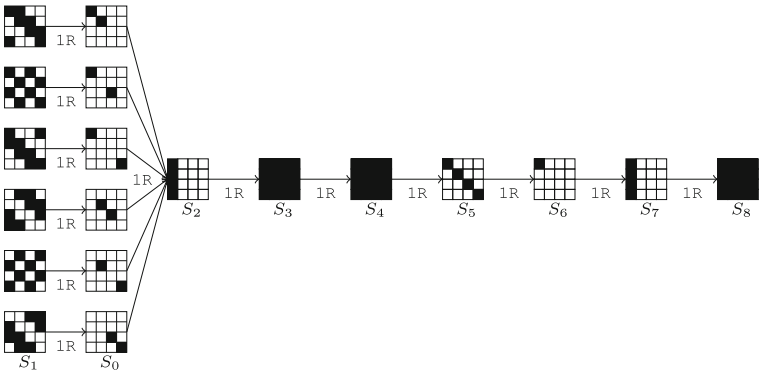


Fig. 6. Differential characteristic for the 8-round chosen-key distinguisher for `AES-128`

Collision. It is also interesting to check what happens if the AES cipher is plugged into a classical Davies-Meyer mode in order to get a compression function. A collision attack for this scenario was proposed in [22] for 6 rounds of AES with 2^{56} computations. By considering the characteristic from state S_1 to state S_7 state in Fig. 5 (the **MixCells** in the last round is omitted for AES, thus S_7 contains only a single active byte), and by using the technique introduced in [10] (only for chosen-key model, but in the Davies-Meyer mode the key input of the cipher is fully controlled by the attacker since it represents the message block input), we can find solutions from state S_2 to state S_6 with a single operation on average. Then, one has to pay a probability 2^{-24} for the differential transition from state S_2 to state S_1 when computing backward. One can not randomize the single active cells positions here because the collision forces us to place them at the very same position. Getting the single input and output active bytes to collide requires 2^8 tries and the total complexity of the 6-round collision search is therefore 2^{32} computations.

5.2 Whirlpool

Whirlpool [1] is a 512-bit hash function whose compression function is built upon a block cipher E in a Miyaguchi-Preneel mode: $h(H, M) = E_H(M) \oplus M \oplus H$. This block cipher E uses two 10-round AES-like permutations with parameters $t = 8$ and $c = 8$, one for the internal state transformation and one for the key schedule. The first permutation is fixed and takes as input the 512-bit incoming chaining variable, while the second permutation takes as input the 512-bit message block, and whose round keys are the successive internal states of the first permutation. The current best distinguishing attack can reach the full 10 rounds of the internal permutation and compression function (with 2^{176} computations), while the best collision attack can reach 5.5 rounds of the hash function and 7.5 rounds of the compression function [20] (with 2^{184} computations). We show how to improve the complexities of all these attacks.

Distinguisher. We reuse the same differential characteristic from [20] for the distinguishing attack on the full 10-round Whirlpool compression function (which contains no difference on the key schedule of E), but we let three more active bytes in both states S_1 and S_8 of the outbound part and this is depicted in Fig. 7. The effect is that the outbound cost of the differential characteristic is reduced to 2^{64} computations: 2^{32} for differential transition from state S_2 to S_1 and 2^{32} from state S_7 to S_8. Moreover, we can leverage the difference position randomization in states S_1 and S_8, which both provide an improvement factor of $\binom{8}{4} = 70$. The inbound part in [20] (from states S_2 to S_7) requires 2^{64} computations to generate a single solution on average, and we obtain a final complexity of $2^{64} \cdot 2^{64} \cdot (70)^{-2} = 2^{115.7}$ Whirlpool evaluations, while the multiple limited-birthday problem has a generic complexity bounded by 2^{125} computations.

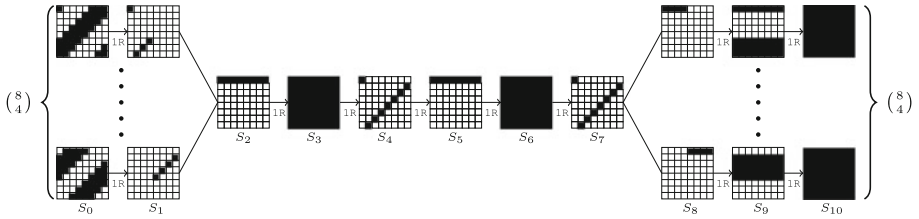


Fig. 7. 10-round truncated differential characteristic for the full `Whirlpool` compression function distinguisher.

Collision. We reuse the same differential characteristic from [20] for the 7.5-round collision attack on the `Whirlpool` compression function (which contains no difference on the key schedule of E), but we let one more active byte in both states S_0 and S_7 of the outbound part (see Fig. 8). From this, we gain an improvement factor of 2^8 in both forward and backward directions of the outbound (from state S_1 to S_0 and from state S_6 to S_7), but we have two byte positions to collide on with the feed-forward instead of one. After incorporating this 2^8 extra cost, we obtain a final improvement factor of 2^8 over the original attack (it is to be noted that this improvement will not work for 7-round reduced `Whirlpool` since the active byte position randomization would not be possible anymore). The very same method applies to the 5.5-round collision attack on the `Whirlpool` hash function.

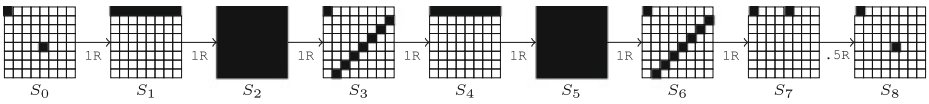


Fig. 8. 7.5-round truncated differential characteristic for the `Whirlpool` compression function collision.

6 Conclusion

In this article, we propose a new type of distinguisher for `AES`-like permutations that we call the *multiple limited-birthday* distinguisher. It generalizes the simple limited-birthday one in the sense that it allows more than just one pattern of fixed difference at both the input and the output of the permutation. We provide an algorithm to efficiently solve the problem for the ideal case, while it remains an open problem to prove its optimality, which can probably be reduced to proving the optimality of the simple limited-birthday algorithm in terms of number of queries. As applications of this work, we show how to improve almost all previously known rebound distinguishers for `AES`-based primitives.

Acknowledgments. We would like to thank Dmitry Khovratovich and the anonymous referees for their valuable comments on our paper.

Appendix

A Other Results

Table 3. Other improvements for various rebound-based attacks on AES-based primitives. Our results marked as **New** are detailed in the extended version of this article

Target	Subtarget	Rounds	Type	Time	Memory	Ideal	Reference
ECHO	Permutation	7	dist.	2^{118}	2^{38}	2^{1025}	[27]
		7	dist.	2^{102}	2^{38}	2^{256}	**New**
		8	dist.	2^{151}	2^{67}	2^{257}	[24]
		8	dist.	2^{147}	2^{67}	2^{256}	**New**
Grøstl-256	Permutation	8	dist.	2^{16}	2^{8}	2^{33}	[27]
		8	dist.	2^{10}	2^{8}	$2^{31.5}$	**New**
		9	dist.	2^{368}	2^{64}	2^{385}	[17]
		9	dist.	2^{362}	2^{64}	2^{379}	**New**
Grøstl-256	Comp. func.	6	collision	2^{120}	2^{64}	2^{257}	[28]
		6	collision	2^{119}	2^{64}	2^{257}	**New**
Grøstl-256	Hash func.	3	collision	2^{64}	2^{64}	2^{129}	[28]
		3	collision	2^{63}	2^{64}	2^{129}	**New**
LED-64	Cipher	15	CK dist.	2^{16}	2^{16}	2^{33}	[15]
		16	CK dist.	$2^{33.5}$	2^{32}	$2^{41.4}$	[25]
		20	CK dist.	$2^{60.2}$	$2^{61.5}$	$2^{66.1}$	[25]
		19	CK dist.	2^{18}	2^{16}	2^{33}	**New**
PHOTON-80/20/16	Permutation	8	dist.	2^{8}	2^{4}	2^{11}	[14]
		8	dist.	$2^{3.4}$	2^{4}	$2^{9.8}$	**New**
PHOTON-128/16/16	Permutation	8	dist.	2^{8}	2^{4}	2^{13}	[14]
		8	dist.	$2^{2.8}$	2^{4}	$2^{11.7}$	**New**
PHOTON-160/36/36	Permutation	8	dist.	2^{8}	2^{4}	2^{15}	[14]
		8	dist.	$2^{2.4}$	2^{4}	$2^{13.6}$	**New**
PHOTON-224/32/32	Permutation	8	dist.	2^{8}	2^{4}	2^{17}	[14]
		8	dist.	2^{2}	2^{4}	$2^{15.5}$	**New**
		9	dist.	2^{184}	2^{32}	2^{193}	[17]
		9	dist.	2^{178}	2^{32}	2^{187}	**New**
PHOTON-256/32/32	Permutation	8	dist.	2^{16}	2^{8}	2^{25}	[14]
		8	dist.	$2^{10.8}$	2^{8}	$2^{23.7}$	**New**

References

1. Barreto, P.S.L.M., Rijmen, V.: Whirlpool. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security, 2nd edn, pp. 1384–1385. Springer, New York (2011)

2. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
3. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 proposal: ECHO. Submission to NIST (2008)
4. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak reference. Submission to NIST (Round 3) (2011)
5. 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)
6. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of `Keccak` and `Luffa`. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer, Heidelberg (2011)
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology. Revisited J. ACM **51**(4), 557–594 (2004)
8. Canteaut, A. (ed.): FSE 2012. LNCS, vol. 7549. Springer, Heidelberg (2012)
9. Daemen, J., Rijmen, V.: Rijndael for AES. In: `AES` Candidate Conference, pp. 343–348 (2000)
10. Derbez, P., Fouque, P.-A., Jean, J.: Faster chosen-key distinguishers on reduced-round AES. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 225–243. Springer, Heidelberg (2012)
11. Duc, A., Guo, J., Peyrin, T., Wei, L.: Unaligned rebound attack: application to Keccak. In: [8] pp. 402–421
12. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: `Grøstl` - a SHA-3 candidate. Submitted to the competition, NIST (2008)
13. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: improved attacks for `AES`-like permutations. In: [26] pp. 365–383
14. Guo, J., Peyrin, T., Poschmann, A.: The `PHOTON` family of lightweight hash functions. In: [26] pp. 222–239
15. 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)
16. Hong, S., Iwata, T. (eds.): FSE 2010. LNCS, vol. 6147. Springer, Berlin (2010)
17. Jean, J., Naya-Plasencia, M., Peyrin, T.: Improved rebound attack on the finalist `Grøstl`. In: [8] pp. 110–126
18. Khovratovich, D., Nikolic, I.: Rotational cryptanalysis of ARX. In: [16] pp. 333–346
19. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: results on the full `Whirlpool` compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009)
20. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: The rebound attack and subspace distinguishers: application to whirlpool. Cryptology ePrint Archive, Report 2010/198 (2010)
21. Matusiewicz, K., Naya-Plasencia, M., Nikolić, I., Sasaki, Y., Schläffer, M.: Rebound attack on the full Lane compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 106–125. Springer, Heidelberg (2009)
22. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced `Grøstl` compression function, `ECHO` permutation and `AES` block cipher. In: Jacobson Jr, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 16–35. Springer, Heidelberg (2009)

23. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: cryptanalysis of reduced `Whirlpool` and `Grøstl`. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
24. Naya-Plasencia, M.: How to improve rebound attacks. In: [26] pp. 188–205
25. Nikolic, I., Wang, L., Wu, S.: Cryptanalysis of Round-Reduced LED. In: FSE. Lecture Notes in Computer Science (2013) (To appear)
26. Rogaway, P. (ed.): CRYPTO 2011. LNCS, vol. 6841. Springer, Heidelberg (2011)
27. Sasaki, Y., Li, Y., Wang, L., Sakiyama, K., Ohta, K.: Non-full-active Super-Sbox analysis: applications to `ECHO` and `Grøstl`. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 38–55. Springer, Heidelberg (2010)
28. Schläffer, M.: Updated differential analysis of `Grøstl`. `Grøstl` website (2011)