Automatic Search for the Best Trails in ARX: Application to Block Cipher SPECK

Alex Biryukov, Vesselin Velichkov, and Yann Le Corre

Laboratory of Algorithmics, Cryptology and Security (LACS)
University of Luxembourg
{Alex.Biryukov,Vesselin.Velichkov,Yann.LeCorre}@uni.lu

Abstract. We propose the first adaptation of Matsui's algorithm for finding the best differential and linear trails to the class of ARX ciphers. It is based on a branch-and-bound search strategy, does not use any heuristics and returns optimal results. The practical application of the new algorithm is demonstrated on reduced round variants of block ciphers from the SPECK family. More specifically, we report the probabilities of the best differential trails for up to 10, 9, 8, 7, and 7 rounds of SPECK32, SPECK48, SPECK64, SPECK96 and SPECK128 respectively, together with the exact number of differential trails that have the best probability. The new results are used to compute bounds, under the Markov assumption, on the security of SPECK against single-trail differential cryptanalysis. Finally, we propose two new ARX primitives with provable bounds against single-trail differential and linear cryptanalysis – a long standing open problem in the area of ARX design.

Keywords: Symmetric-key, Cryptanalysis, ARX, SPECK

1 Introduction

ARX stands for Addition/Rotation/XOR and denotes a class of cryptographic algorithms based on the simple arithmetic operations: modular addition, bitwise rotation (and bitwise shift) and exclusive-OR. Although the acronym has gained popularity only recently, algorithms using these operations have been designed ever since the 80s.

Some notable historical examples of ARX designs are the block ciphers FEAL (1987), RC5 (1994), and TEA (1994) (with its modified versions XTEA (1997) and XXTEA (1998)). More recent proposals include the stream cipher Salsa20 (2008) and its variant ChaCha (2008); the hash functions BLAKE (2008) (using a modified version of ChaCha) and Skein [12] (2008) (with its underlying block cipher Threefish); the hash function for short messages SipHash (2012) and the block cipher SPECK [2] (2013) (both using a variant of Threefish's MIX operation); the lightweight block cipher LEA (2013) and the MAC algorithm for 32-bit microcontrollers Chaskey (2014) (based on a reduced word-size variant of SipHash's round function).

All mentioned ARX designs are also called *pure*, since they are exclusively composed of the three basic ARX operations. In addition, there is also the subclass of *augmented* ARX designs that consists of a combination of the ARX operations with other bitwise operations such as Boolean operators, Boolean functions, etc. The most eminent representatives of this group are the hash functions from the MD and SHA families.

As evidenced by the long list of proposals, there is a steady interest in the ARX design philosophy. The reason is the simplicity and efficiency in both software and hardware of these designs. In recent years ARX algorithms have become especially attractive in the area of lightweight cryptography for environments with highly constrained resources. According to new results from the Framework for Fair Evaluation of Lightweight Cryptographic Systems (FELICS) [6], presented at the NIST Lightweight Cryptography Workshop 2015 [25], the most efficient lightweight designs have ARX structure.

The ARX class of primitives is often seen as an alternative to the well-established class of S-box based algorithms, among whose most notable representatives are the block cipher AES [8] and the historically significant block cipher DES [24]. While primitives from this class make use of substitution tables (S-boxes) as a source of non-linearity, the only non-linear component in ARX is the modular addition operation. Due to the latter, these primitives are also less vulnerable to cache-timing and side-channel attacks.

While ARX algorithms provide level of security comparable to S-box based ones, they suffer from a major drawback – the methods for their analysis and design are far less rigorous and mature. For S-box based ciphers it is possible to compute provable bounds on the security against the two most powerful cryptanalytic attacks – differential cryptanalysis [3] and linear cryptanalysis [19] (see e.g. [7]). In contrast, the state of the art in the design of ARX can be summarized in the following heuristic common-sense rule: mix the basic arithmetic operations in a *reasonable* way and iterate them over *sufficient* number of rounds. While this strategy seems to be largely successful in practice, it is based more on experience and intuition, rather than on sound scientific arguments.

In this paper we address the mentioned problem by proposing for the first time an algorithm that finds the best differential and linear trails of an ARX cipher for a given number of rounds. It is based on a branch-and-bound search strategy similar to Matsui's search algorithm that was applied to DES [18] and is inspired by the threshold search technique proposed in [5]. While the latter uses heuristics in order to find high-probability trails that are not necessarily optimal, our algorithm does not use any heuristics and finds optimal results.

The trails found with the described method are optimal under the *Markov assumption* [14, Sect. 3, Theorem 2] (see also [8, § 6.2, pp. 84]). The Markov assumption ensures that (a) the analyzed primitive is a Markov cipher in the sense of the definition in [14, Sect. 3] and (b) it can be assumed that its round keys are chosen at random independently (i.e. the Hypothesis of independent round keys [8, § 8.7.2] holds). The Markov assumption allows to treat the rounds of an iterated cipher independently and thus to compute the

differential probability (resp. absolute linear correlation) of an r-round trail as the product of the probabilities (resp. absolute correlations) of its corresponding 1-round trails. **For ciphers that do not satisfy the Markov assumption, fixed keys may exist for which the probability (resp. correlation) of the best differential (resp. linear) trail may significantly deviate from the optimal one as computed with our algorithm.**

As a demonstration of the effectiveness of the technique we apply it to block cipher SPECK and we report for the first time all provably best (under the Markov assumption) differential trails for reduced number of rounds. We also demonstrate that in some cases the threshold search algorithm returns sub-optimal results. These new results are summarized in Table 1.

Table 1. Probabilities of the best (under the Markov assumption) differential trails for SPECK found with Best Search (BS) (Sect. 5) versus best probabilities found with Threshold Search (TS) [4]. The column # lists the number of trails having the best probability. The column R contains the number of rounds.

	Speck32			Speck48			Speck64			Speck96		Speck128	
R	TS	BS	#	TS	BS	#	TS	BS	#	BS	#	BS	#
1	-0	-0	3	-0	-0	3	-0	-0	3	-0	3	-0	3
2	-1	-1	3	-1	-1	3	-1	-1	3	-1	3	-1	3
3	-3	-3	3	-4	-3	2	-3	-3	2	-3	2	-3	2
4	-5	-5	1	-7	-6	2	-7	-6	2	-6	2	-6	2
5	-9	-9	2	-10	-10	4	-13	-10	2	-10	2	-10	2
6	-15	-13	1	-14	-14	2	-21	-15	2	-15	2	-15	2
7	-22	-18	2	-20	-19	2	-27	-21	3	-21	2	-21	≥ 1
8	-26	-24	7	-27	-26	12	-32	-29	≥ 1	< -27		< -26	
9	-30	-30	15	-33	-33	≥ 1	-36	< -31					
10		-34	1	-40	< -34		-40						
11				-47			-44						
12							-47						
13							-52						
14							-60						

As noted, the results shown in Table 1 are to be interpreted under the Markov assumption. In Appendix 8 we show for the first time that SPECK is not, in fact, a Markov cipher. We stress, however, that making the Markov assumption even for non-Markov ciphers is the best that a cryptanalyst can do in order to be able to analyze such constructions. Furthermore, we have experimentally checked that the reported differentials hold for most of the keys and therefore the results shown in Table 4 are meaningful from a practitioner's perspective.

The new technique can also be used to design new ARX primitives with provable security bounds against linear and differential cryptanalysis – a long

standing problem in the area of ARX design. Our main contributions can be summarized as follow:

1. An algorithm for finding the best differential and linear trails in ARX ciphers that satisfy the Markov Assumption.
2. The probabilities of the best differential trails for up to 10, 9, 8, 7, and 7 rounds of Speck32, Speck48, Speck64, Speck96 and Speck128 respectively, together with the exact number of differential trails that have the best probability.
3. A better choice of rotation constants for Speck w.r.t. single-trail differential cryptanalysis.
4. Bounds on the security of Speck, under the Markov assumption, against differential cryptanalysis, based on the reported best trails.
5. Two atomic ARX constructions with provable bounds against single-trail differential and linear cryptanalysis.

The paper is organized as follows. We begin in Sect. 2 with a review of previous work on techniques for searching for differential and linear trails in ARX. Sect. 3 provides basic definitions and propositions, necessary to follow the exposition in subsequent sections. A general strategy for searching for the best trails in ARX is described in Sect. 4 and the results from its application to Speck are given in Sect. 5. Two new primitives – MARX and Speckey – with provable bounds against single trail differential and linear cryptanalysis are proposed in Sect. 6 and Section 7 concludes the paper. The notation used in the paper is summarized in Table 2.

Table 2. Notation.

Symbol	Meaning
w	Word size in bits
n	Total number of rounds
r	Iterator over the number of rounds: $1 \leq r \leq n$
N	Cipher block size (in bits)
LSB, MSB	Least Significant Bit, Most Significant Bit
$x[i]$	The i-th bit of w-bit word x: $0 \leq i < w$: $x[0]$ = LSB, $x[w-1]$ = MSB
$x[i:j]$	The sequence of bits $x[i], x[i+1], \ldots, x[j]$ (if $i < j$) or $x[i], x[i-1], \ldots, x[j]$ (if $i > j$)
$\boxplus$	Addition modulo 2^w
α_r, β_r	Input XOR differences (resp. linear masks) to $\boxplus$ at round r
γ_r	Output XOR difference (resp. linear mask) of $\boxplus$ at round r
$(\alpha_r, \beta_r \rightarrow \gamma_r)$	A differential or a linear approximation of $\boxplus$
$\lvert c \rvert$	Absolute value of c

2 Previous Work

Finding high probability (resp. high absolute correlation) trails for ARX has traditionally been a difficult task. The lack of S-boxes in this class of primitives does not allow to efficiently compute the probabilities (resp. correlations) of all possible differential transitions (resp. linear approximations) by the means of the difference distribution table – DDT (resp. linear approximation table – LAT) of the non-linear elements. This makes the construction of trails in ARX a tedious and especially error-prone process as shown in [15]. Furthermore, while most S-box designs are word-based with relatively small word sizes of 4 and 8 bits, all ARX designs are bit-based with typical size of the words 32 and 64 bits. As a consequence it is not possible to apply elegant design strategies such as the wide-trail [7] to design primitives with provable bounds against differential and linear cryptanalysis. Indeed the design of such an ARX construction is still an open problem.

The described difficulties in the analysis and design of ARX have been addressed by several researchers in the past. Depending on the angle from which they approach the problem, their work can broadly be divided into three categories: bottom-up, top-down and approximation-based techniques. We briefly describe these categories below.

Bottom-up Techniques. This category is by far the largest and encompasses methods for the (automatic) construction of differential and linear trails in ARX. Arguably the first such techniques date back to the collisions on the MD and SHA families of hash functions by Wang et al. [34–36]. While these results were reportedly developed by hand, subsequent methods were proposed for the fully automatic construction of differential paths in ARX all of which were applied to *augmented* ARX designs such as SHA1, SHA2, MD4 and MD5. In [16] was proposed a method for the automatic construction of differential trails in *pure* ARX designs and applied to the hash function Skein. While many of the mentioned techniques are general and potentially applicable to any ARX primitive, all of them were applied exclusively to hash functions. To fill the gap, in [5] was proposed the threshold search method for searching for differential trails in ARX ciphers such as TEA, XTEA and Speck. This method was subsequently extended to the case of differentials in [4]. Most recently, in 2015, two new techniques for automatic search for linear trails have been proposed. One has been applied to Speck [37], while the other is dedicated to authenticated encryption schemes [11].

Top-down Techniques. Rather than constructing a trail one round at a time as in the bottom-up approach, top-down techniques consider the cipher as a whole. More precisely, the cipher is represented either as a system of Boolean equations or as a system of mixed-integer inequalities. Each solution to the system corresponds to a valid trail. In the first case, the Boolean equations are transformed into a conjunctive normal form (CNF) formula, whose satisfying assignment/s are found with a SAT solver. In the second case, the problem of searching for

trails is effectively transformed into a mixed-integer linear problem (MILP) that is usually solved by dedicated MILP solvers using linear-programming based branch-and-bound algorithms. The SAT solver approach has been used to find the best differential trails for several rounds of stream cipher Salsa20 and for proving security bounds for the authenticated encryption cipher NORX. As to the MILP-based methods, up to now they have been successful mainly in the analysis of S-box designs [23, 28]. The only applications of MILP to ARX that we are aware of are the results on the augmented ARX cipher Simon [28] and a very recent paper [13] on Speck appearing in this volume of FSE'16.

Approximation-based Techniques. In both top-down and bottom-up approaches, complex techniques for analysis of existing algorithms are developed. In contrast, in what we call approximation-based techniques, the problem is turned around: new primitives are developed so that they are easy to be analyzed *by design*. The main idea is to replace the non-linear component of ARX – the modular addition – by a simpler non-linear approximation that can efficiently and accurately be analyzed with existing methods. A design based on this strategy is the authenticated encryption scheme NORX [1]. In it the addition operation is replaced by the first-order approximation $a \oplus b \oplus (a \wedge b) \ll 1 \approx a \boxplus b$, which effectively limits the carry propagation to a sliding window of 2 bits. The latter significantly facilitates the analysis of the scheme and also makes it hardware efficient.

From the above overview of existing results it is clear that the question of finding optimal trails in pure ARX ciphers has remained largely unexplored so far. The only results in this direction that we are aware of are [21], which applies a SAT solver approach and the MILP-based technique in [13]. While the latter is potentially capable of finding optimal trails, its running time is not well understood. To speed up the search, the authors apply a splicing heuristic and their objective is finding better trails than existing ones rather than finding optimal trails. We address this limitation with the method described in the following sections.

3 Preliminaries

In this section we state basic definitions and propositions, that will be used in later sections. We begin with the definitions of the differential probability xdp^+ and the linear correlation xlc^+.

Definition 1 (xdp^+)**.** *The* `XOR` *differential probability (DP) of addition modulo* 2^w *(*xdp^+*) is the probability with which input* `XOR` *differences* α *and* β *propagate to output* `XOR` *difference* γ *through the modular addition operation. The probability* xdp^+ *is computed over all pairs of* w*-bit inputs* (x, y)*:*

$$\mathrm{xdp}^+(\alpha, \beta \rightarrow \gamma) = 2^{-2w} \cdot \#\{(x, y) : ((x \oplus \alpha) + (y \oplus \beta)) \oplus (x + y) = \gamma\} \ . \quad (1)$$

The linear correlation xlc^+ is defined in a similar way:

Definition 2 (xlc^+). *The* `XOR` *linear correlation (LC) of addition modulo* 2^w *(*xlc^+*) is the correlation of the linear approximation* $(\alpha^T x) \oplus (\beta^T y) = (\gamma^T z)$*, where* $x, y, z : \ x + y = z \mod 2^w$ *are* w*-bit values and* α*,* β *and* γ *are* w*-bit linear masks, all represented as binary vectors of dimension* $w \times 1$*. The operation* $\Gamma^T a$ *denotes the dot product between the transposed vector* Γ *(the mask) and the vector* a*. The correlation* xlc^+ *is computed over all pairs of* w*-bit inputs* (x, y)*:*

$$\mathrm{xlc}^+(\alpha, \beta \to \gamma) = 2^{-2w+1} \cdot \#\{(x, y) : (\alpha^T x) \oplus (\beta^T y) = (\gamma^T z)\} - 1 \ . \quad (2)$$

The absolute value of the linear correlation is denoted by $|\mathrm{xlc}^+|$*.*

The probability xdp^+ has the following property noted in [5, Sect. 2, Proposition 1]:

Proposition 1 (Monotonicity of xdp^+**).** *Let* α*,* β *and* γ *be* w*-bit* `XOR` *differences. Denote with* $\tilde{p}_i$ *(*$w \geq i \geq 1$*) the probability* $\mathrm{xdp}^+(\alpha[i-1:0], \beta[i-1:0] \to \gamma[i-1:0])$ *of the partial differential composed of the* i *LS bits of* α*,* β*,* γ*. Then the probability* xdp^+ *is monotonously decreasing with the word size of the differences in the direction LSB to MSB:*

$$\tilde{p}_1 \geq \tilde{p}_2 \ldots \geq \tilde{p}_{w-1} \geq \tilde{p}_w = \mathrm{xdp}^+(\alpha, \beta \to \gamma) \ . \quad (3)$$

Similar property holds also for $|\mathrm{xlc}^+|$, but in this case the correlation decreases from MSB to LSB of the masks:

Proposition 2 (Monotonicity of xlc^+**).** *Let* α*,* β *and* γ *be* w*-bit linear masks. Denote with* $\tilde{c}_i$ *(*$w-1 \geq i \geq 0$*) the absolute value of the correlation* $\mathrm{xlc}^+(\alpha[w-1:i], \beta[w-1:i] \to \gamma[w-1:i])$ *of the partial linear approximation composed of the* $w - i$ *MS bits of* α*,* β*,* γ*. Then the absolute correlation* $|\mathrm{xlc}^+|$ *is monotonously decreasing with the word size of the masks in the direction MSB to LSB:*

$$\tilde{c}_{w-1} \geq \tilde{c}_{w-2} \ldots \geq \tilde{c}_1 \geq \tilde{c}_0 = |\mathrm{xlc}^+(\alpha, \beta \to \gamma)| \ . \quad (4)$$

The DP and LC of modular addition have been thoroughly studied in the literature and optimal methods for their computation have been proposed by several authors: [17, 33, 22] (for xdp^+) and [32, 26, 20, 27, 10, 33] (for xlc^+). All cited methods are linear in the size of the differences (resp. masks).

In the following sections, for computing xdp^+ we use the method proposed in [17] and for xlc^+ we use the algorithm described in [10].

4 Best Trail Search for ARX

In this section we describe for the first time a Matsui-like algorithm for finding the best differential and linear trails in ARX ciphers for which the Markov assumption holds. Our technique belongs to the class of bottom-up approaches. It is based on Matsui's branch-and-bound algorithm [18], originally designed for the class of S-box ciphers, and is inspired by the threshold search algorithm proposed in [5].

To search for the best trail on n rounds of a cipher, Matsui's algorithm is initialized with the best probabilities $B_1, B_2, \ldots, B_{n-1}$ for the first $n-1$ rounds and an over-estimation $\overline{B}_n \leq B_n$ of the best probability B_n for n rounds (the bound). The search proceeds recursively over the rounds starting from the first ($r = 1$) and gradually builds a trail until the n-th round is reached. At every round $1 \leq r \leq n$ the probability $\prod_{i=1}^{r} p_i$ of the partially constructed trail up to round r is multiplied by the best probability B_{n-r} for the remaining $n-r$ rounds to obtain an estimate for the full trail. If $B_{n-r} \prod_{i=1}^{r} p_i < \overline{B}_n$ (i.e. the estimate is lower than the bound), the algorithm backtracks to the previous round. In this way branches of the recursion tree, that are not prospective, are cut. At the last round the probability of the full trail is compared to the bound and if it is bigger, the bound is set to the new probability: $\overline{B}_n \leftarrow \prod_{i=1}^{n} p_i$. The procedure terminates when the bound $\overline{B}_n$ can not be updated any more. As long as the condition $\overline{B}_n \leq B_n$ is preserved, the returned result is guaranteed to be optimal. The probabilities (resp. correlations) p_i are computed by means of the DDT (resp. LAT) of the cipher's S-box.

In [5] was proposed a variant of Matsui's algorithm applicable to the class of ARX ciphers, called *threshold search*. The main idea is to consider addition modulo 2^w as a large S-box of size $2^{2w} \times 2^w$. Since computation of the full DDT of this S-box is infeasible for typical word sizes of $w \geq 16$ bits, the authors propose to use a DDT with reduced size, called *partial* DDT (or pDDT). The pDDT is composed of (a subset of) all differential transitions that have probability larger than- or equal to a predefined probability threshold. The value of the threshold and the maximum allowed size of the pDDT are chosen heuristically depending on the analyzed primitive. Another proposed heuristic is a limit on the Hamming weight of the differences.

If an input difference with no matching output difference in the pDDT is encountered during the search, a second pDDT is computed on-the-fly. The latter is composed of transitions that (a) have probabilities that are likely to improve the probability of the best trail found so far and (b) are guaranteed to result in input differences to the next round, that have at least one matching output difference in the initial pDDT (as illustrated by the *The Highways and Country Roads Analogy* [5]). Due to the use of the mentioned heuristics, the trails found by the threshold search algorithm are not necessarily optimal.

Inspired by the threshold search approach, we propose a new variant of Matsui's algorithm for the class of ARX. In contrast to [5] our technique does not use any heuristics and finds optimal results. The main new idea is to add a second recursion at bit-level over the bits of the differences (resp. linear masks) in addition to the original recursion over the rounds. This modification preserves the optimality of the search due to the monotonicity properties of modular addition stated as Proposition 1 and Proposition 2 in Sect. 3. These properties allow us, at every round r, to compute the probability of the partially constructed trail at the bit-level using the partially constructed differences (resp. masks) at round r. Unprospective branches of the search tree are thus effectively cut not only at round-level, but also at bit-level.

In more detail, let $\alpha_r[0:i]$, $\beta_r[0:i]$ and $\gamma_r[0:i]$ be resp. input and output differences to the modular addition at round r, that are partially constructed up to bit i (i.e. only the $i+1$ LS bits of the words are assigned). Let $\tilde{p}_r$ be the probability of the corresponding partially constructed differential: $(\alpha_r[0:i], \beta_r[0:i]) \rightarrow \gamma_r[0:i]$. Then at round r and bit i, the algorithm checks whether the following condition holds: $B_{n-r}\tilde{p}_r \prod_{i=1}^{r-1} p_i \geq \overline{B}_n$ i.e. if the product of the probability $\prod_{i=1}^{r-1} p_i$ of the partially constructed trail up to round $r-1$ and the probability $\tilde{p}_r$ of the partially constructed differential up to bit i at round r and the best probability B_{n-r} for the remaining $n-r$ rounds is still at least as good as the bound $\overline{B}_n$. If yes, then the search proceeds recursively to the next bit position $i+1$ or, if $i = w$, to the next round $r+1$. Otherwise, it backtracks to the previous bit or, if $i = 0$, to the previous round.

With the described strategy, we effectively deal with the problem of having to store huge number of possible transitions through the addition operation. Consequently it is not necessary to maintain a (partial) DDT or to use additional heuristics such as probability and Hamming weight thresholds to limit the search and storage space. Moreover, our algorithm is conceptually closer to Matsui's original proposal than the threshold search. In his paper [18], Matsui also describes a second level of recursion over the 8 S-boxes of DES (cf. procedure *Round-2-j* in [18, Sect. 4, pp. 371]). With it the probability of a partial trail is computed up to round $r-1$ and up to S-box i at round r, where $1 \leq i \leq 8$. This S-box level recursion is analogous to the proposed bit-level recursion for modular addition.

In the following sections we use the block cipher Speck to illustrate the application of the new technique in practice.

5 Application to Speck

5.1 Description of Speck

Speck is a family of lightweight block ciphers proposed in [2]. It is composed of the five instances Speck32, Speck48, Speck64, Speck96 and Speck128, corresponding resp. to the block sizes $32, 48, 64, 96$ and 128 bits. Note that the instance SpeckN has $N/2$-bit word size. In the following, with Speck we denote any of the five variants if not otherwise specified.

Speck is a pure ARX cipher with a Feistel-like structure in which both branches are modified at every round. Let $X_{r-1,\mathrm{L}}$ and $X_{r-1,\mathrm{R}}$ be respectively the right and left $N/2$-bit input words to the r-th round of SpeckN ($r \geq 1$) and let k_r be the $N/2$-bit round key applied at round r (see Fig. 1 (Left)). Then the output words $X_{r,\mathrm{L}}$, $X_{r,\mathrm{R}}$ from round r (input words to round $r+1$) are computed as follow:

$$X_{r,\mathrm{L}} = ((X_{r-1,\mathrm{L}} \ggg r_1) \boxplus X_{r-1,\mathrm{R}}) \oplus k_r \ , \tag{5}$$

$$X_{r,\mathrm{R}} = (X_{r-1,\mathrm{R}} \lll r_2) \oplus X_{r,\mathrm{L}} \ . \tag{6}$$

The rotation constants r_1, r_2 are specified as: $r_1 = 7, r_2 = 2$ for SPECK32 and $r_1 = 8, r_2 = 3$ for all other versions. The round function of SPECK is depicted in Fig. 1 (Left).

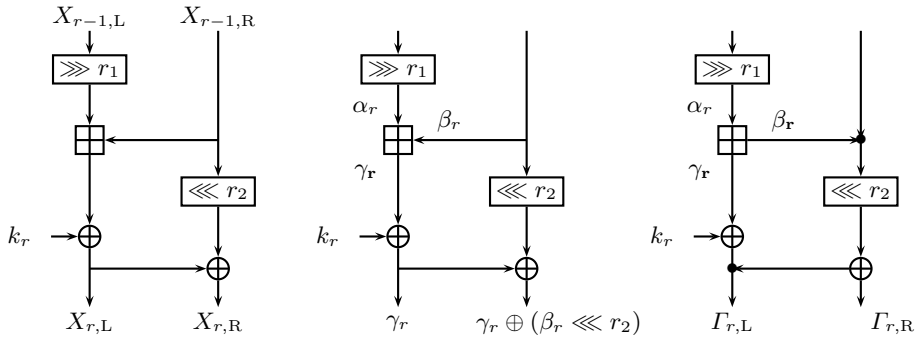

Fig. 1. Left: The round function of SPECK. **Middle:** Propagation of differences: $\alpha_r = \gamma_{r-1} \ggg r_1$, $\beta_r = \gamma_{r-1} \oplus (\beta_{r-1} \lll r_2)$. **Right:** Propagation of linear masks: $\alpha_r = \Gamma_{r-1,\mathrm{L}} \ggg r_1$, $\beta_r = \Gamma_{r-1,\mathrm{R}} \oplus (\Gamma_{i,\mathrm{R}} \ggg r_2)$, $\gamma_r = \Gamma_{i,\mathrm{L}} \oplus \Gamma_{i,\mathrm{R}}$. The $\bullet$ sign denotes a "three-forked branch" and acts as a XOR on the linear masks [18]. Differences $\gamma_{\mathbf{r}}$ (resp. masks $\beta_{\mathbf{r}}$, $\gamma_{\mathbf{r}}$) in bold can be freely chosen.

Every instance of the SPECK family supports several key sizes and the total number of rounds depends on the key size. A summary of the parameters (block size, key size, number of rounds) of all instances of the family is presented in Table 3.

Table 3. SPECK parameters: block size (bits), key size (bits), number of rounds.

Instance SPECKN	Block Size (N)	Word Size ($N/2$)	Key Size	Rounds	Key Size	Rounds	Key Size	Rounds
SPECK32	32	16	64	22				
SPECK48	48	24	72	22	96	23		
SPECK64	64	32	96	26	144	29		
SPECK96	96	48	96	28	144	29		
SPECK128	128	64	128	32	192	33	256	34

The key schedule of SPECK is based on a simple ARX function that is iterated a fixed number of times. We omit its description herein, as it is not relevant to

the presented results. For the detailed description of the SPECK family we refer the reader to the original proposal [2].

5.2 Best Trail Search for SPECK

In this section we apply the technique described in Sect. 4 in order to find the best (under the Markov assumption) linear and differential trails of reduced-round variants of SPECK.

Differential Trail Search. The pseudo-code of the algorithm for the best differential trail search applied to SPECK is shown in Alg. 1. It has three parts: first round (lines (4)-(14)), middle rounds (lines (16)-(25)) and last round (lines (27)-(37)). Every part is composed of two blocks corresponding to the two levels of recursion. In the first round the procedure starts by recursing over the bits of the differences (lines (10)-(14)) beginning with the LSB. When the MSB is reached (line (5)) (i.e. the differences α_r, β_r, γ_r are fully constructed), the procedure switches back to the first block (lines (5)-(8)), where it recurses into the next round (line (8)). The logic for the middle and last rounds is the same with the exception that the bit level recursion is over the bits of the output difference γ_r only (lines (22)-(25) and (34)-(37) resp.) and not over the bits of all differences as in the first round. The reason is that the input differences α_r and β_r to the addition in the middle and last rounds are fixed from the previous round by the following relation: $\alpha_r = \gamma_{r-1} \ggg r_1$, $\beta_r = \gamma_{r-1} \oplus (\beta_{r-1} \lll r_2)$ (see line (7) and Fig. 1 (middle)). In addition, at the last round there is no further round level recursion, but instead the bound $\overline{B}_n$ is updated (line (32)).

We estimate the complexity of the differential search algorithm as follows. Let $m_1 \leq 2^{3w}$ be the number of differences α_1, β_1 and γ_1 in the first round, for which the probability of the differential $(\alpha_1, \beta_1 \to \gamma_1)$ is higher than $\overline{B}_n/B_{n-1}$: $m_1 = \#\{(\alpha_1, \beta_1, \gamma_1) : \mathrm{xdp}^+(\alpha_1, \beta_1 \to \gamma_1) \geq \overline{B}_n/B_{n-1}\}$. Analogously, let $m_r \leq 2^w$ be the number of differences γ_r in any middle or last round $r \geq 2$ for which, for fixed α_r and β_r, the probability of the differential $(\alpha_r, \beta_r \to \gamma_r)$ is higher than $\overline{B}_n/(B_{n-r}\prod_{i=1}^{r} p_i)$: $m_r = \#\{\gamma_r : \mathrm{xdp}^+(\alpha_r, \beta_r \to \gamma_r) \geq \overline{B}_n/(B_{n-r}\prod_{i=1}^{r} p_i)\}$, and let m be the maximum among these values: $m = \max_{n \geq r \geq 2} (m_r)$. Then the complexity of Alg. 1 has the form $\mathcal{O}(\prod_{r=1}^{n} m_r) \leq \mathcal{O}(m_1 m^{r-1})$, which is significantly lower than the complexity of full search $2^{3w}2^{w(r-1)} = 2^{w(r+2)}$ as indicated by our experiments. However, the precise quantification of the values m_r, $r \geq 1$ is difficult, since they change dynamically during the search. The latter is a separate problem in itself, that can be investigated in future research.

Linear Trail Search. The algorithm for linear search for SPECK is analogous to the differential case with one significant difference, arising from the way in which linear masks propagate through the round function (see Fig. 1 (right)). Recall that in the differential search, the differences α_r and β_r in the middle and last rounds are fixed from the previous round. In contrast, in the linear case only the mask α_r is fixed (with the relation $\alpha_r = \gamma_{r-1} \lll r_1$), while β_r depends on the right output masks $\Gamma_{r-1,\mathrm{R}}$ and $\Gamma_{r,\mathrm{R}}$ resp. from the previous and

Algorithm 1 Search for the Best Differential Trails in ARX (Application to SPECK).

Input: |
n : num. rounds; w: word size in bits; r_1, r_2: right and left rot. const.;
r : current round ($n \geq r \geq 1$);
i : current bit position ($w > i \geq 0$);
$B = (B_1, B_2, \ldots, B_{n-1})$: probs. of the best trails for rounds $1, 2, \ldots, (n-1)$;
$\overline{B}_n$: underestimate of the best prob. for n rounds: $\overline{B}_n \leq B_n$;
$T = (T_1, T_2, \ldots, T_{r-1})$: $T_i = (\alpha_i, \beta_i, \gamma_i, p_i) : p_i = \mathrm{xdp}^+(\alpha_i, \beta_i \rightarrow \gamma_i)$, $1 \leq i < r$;
$(\alpha_r, \beta_r, \gamma_r)$: input and output differences to the mod. addition at round r;
$\tilde{p}_r$: probability of the partial differential $(\alpha_r[0:i], \beta_r[0:i] \rightarrow \gamma_r[0:i])$;

Output: |
B_n, T : the best prob. for n rounds and corresponding trail;

1: // Initialization: $r \leftarrow 1$, $i \leftarrow 0$, $\alpha_r \leftarrow \emptyset$, $\beta_r \leftarrow \emptyset$, $\gamma_r \leftarrow \emptyset$
2: **procedure best_diff_search**$(r, i, \alpha_r, \beta_r, \gamma_r)$ **do**
3: // First round
4: **if** $(r = 1) \wedge (r \neq n)$ **then**
5: **if** $i = w$ **then**
6: $p_r \leftarrow \mathrm{xdp}^+(\alpha_r, \beta_r \rightarrow \gamma_r)$; $T_r \leftarrow (\alpha_r, \beta_r, \gamma_r, p_r)$; **add** T_r to T;
7: $i \leftarrow 0$; $\alpha_{r+1} \leftarrow (\gamma_r \ggg r_1)$; $\beta_{r+1} \leftarrow \gamma_r \oplus (\beta_r \lll r_2)$; $\gamma_{r+1} \leftarrow \emptyset$;
8: call **best_diff_search**$(r+1, i, \alpha_{r+1}, \beta_{r+1}, \gamma_{r+1})$
9: **else**
10: **for** $j_\alpha, j_\beta, j_\gamma \in \{0, 1\}$ **do**
11: $\alpha_r[i] \leftarrow j_\alpha$; $\beta_r[i] \leftarrow j_\beta$; $\gamma_r[i] \leftarrow j_\gamma$;
12: $\tilde{p}_r \leftarrow \mathrm{xdp}^+(\alpha_r[0:i], \beta_r[0:i] \rightarrow \gamma_r[0:i])$;
13: **if** $(\tilde{p}_r\ B_{n-1}) \geq \overline{B}_n$ **then**
14: call **best_diff_search**$(r, i+1, \alpha_r, \beta_r, \gamma_r)$
15: // Intermediate rounds
16: **if** $(r > 1) \wedge (r \neq n)$ **then**
17: **if** $i = w$ **then**
18: $p_r \leftarrow \mathrm{xdp}^+(\alpha_r, \beta_r \rightarrow \gamma_r)$; $T_r \leftarrow (\alpha_r, \beta_r, \gamma_r, p_r)$; **add** T_r to T;
19: $i \leftarrow 0$; $\alpha_{r+1} \leftarrow (\gamma_r \ggg r_1)$; $\beta_{r+1} \leftarrow \gamma_r \oplus (\beta_r \lll r_2)$; $\gamma_{r+1} \leftarrow \emptyset$;
20: call **best_diff_search**$(r+1, i, \alpha_{r+1}, \beta_{r+1}, \gamma_{r+1})$
21: **else**
22: **for** $j_\gamma \in \{0, 1\}$ **do**
23: $\gamma_r[i] \leftarrow j_\gamma$; $\tilde{p}_r \leftarrow \mathrm{xdp}^+(\alpha_r[0:i], \beta_r[0:i] \rightarrow \gamma_r[0:i])$
24: **if** $(p_1\ p_2\ \ldots\ p_{r-1}\ \tilde{p}_r\ B_{n-r}) \geq \overline{B}_n$ **then**
25: call **best_diff_search**$(r, i+1, \alpha_r, \beta_r, \gamma_r)$
26: // Last round
27: **if** $(r = n)$ **then**
28: **if** $i = w$ **then**
29: $p_n \leftarrow \mathrm{xdp}^+(\alpha_n, \beta_n \rightarrow \gamma_n)$; $T_n \leftarrow (\alpha_n, \beta_n, \gamma_n, p_n)$; **add** T_n to T;
30: **if** $(p_1\ p_2\ \ldots\ p_{n-1}\ p_n) \geq \overline{B}_n$ **then**
31: // Update bound and return to upper round
32: $\overline{B}_n \leftarrow (p_1\ p_2\ \ldots\ p_{n-1}\ p_n)$
33: **else**
34: **for** $j_\gamma \in \{0, 1\}$ **do**
35: $\gamma_n[i] \leftarrow j_\gamma$; $\tilde{p}_n \leftarrow \mathrm{xdp}^+(\alpha_n[0:i], \beta_n[0:i] \rightarrow \gamma_n[0:i])$
36: **if** $(p_1\ p_2\ \ldots\ p_{n-1}\ \tilde{p}_n) \geq \overline{B}_n$ **then**
37: call **best_diff_search**$(r, i+1, \alpha_r, \beta_r, \gamma_r)$
38: **return**

current round: $\beta_r = \Gamma_{r-1,\mathrm{R}} \oplus (\Gamma_{r,\mathrm{R}} \ggg r_2)$. Due to this fact, in the middle and last rounds the linear search algorithm performs a recursion over the bits of one more variable (β_r) in addition to γ_r. Furthermore, since the mask $\Gamma_{r-1,\mathrm{R}}$ can be freely chosen in the first round, an additional iteration over all such masks is performed. The latter is independent of the bound $\overline{B}_n$ and therefore represents a fixed cost of 2^w additional iterations. All this added complexity makes the linear search algorithm feasible only for the version SPECK32.

Due to the mentioned differences, the complexity of the linear search is significantly higher than the differential search. Let $m_1 \leq 2^{3w}$ be the number of masks α_1, β_1 and γ_1 in the first round, for which the absolute correlation of the linear approximation $(\alpha_1, \beta_1 \rightarrow \gamma_1)$ is higher than $\overline{B}_n/B_{n-1}$: $m_1 = \#\{(\alpha_1, \beta_1, \gamma_1) : |\mathrm{xlc}^+(\alpha_1, \beta_1 \rightarrow \gamma_1)| \geq \overline{B}_n/B_{n-1}\}$. Let $m_r \leq 2^{2w}$ be the number of masks β_r and γ_r in any middle or last round $r \geq 2$ for which, for fixed α_r, the absolute correlation of the linear approximation $(\alpha_r, \beta_r \rightarrow \gamma_r)$ is higher than $\overline{B}_n/(B_{n-r}\prod_{i=1}^{r} c_i)$: $m_r = \#\{(\beta_r, \gamma_r) : |\mathrm{xlc}^+(\alpha_r, \beta_r \rightarrow \gamma_r)| \geq \overline{B}_n/(B_{n-r}\prod_{i=1}^{r} c_i)\}$, and let $m = \max_{n \geq r \geq 2} (m_r)$. Then the complexity of the linear search algorithm has the form: $\mathcal{O}(2^w \prod_{r=1}^{n} m_r) \leq \mathcal{O}(2^w m_1 m^{r-1})$, which is much less than the complexity of full search $2^{4w}2^{2w(r-1)} = 2^{2w(r+1)}$. In the former, notice the factor 2^w due to the additional iteration over all w-bit masks $\Gamma_{r-1,\mathrm{R}}$ in the first round. Again, similarly to the differential case, the precise quantification of the values m_r, $r \geq 1$ in the linear case is difficult.

While the higher complexity of the linear search algorithm makes it infeasible for versions of SPECK other than SPECK32, Alg. 1 is quite practical as shown by the results reported in the following section.

5.3 Results

With Alg. 1 we find the best differential trails for reduced round variants of all versions of SPECK under the Markov assumption. Table 1 compares our results to the ones obtained with the threshold search algorithm with the parameters given in [4, Sect. 6, Table 6]: probability threshold $p_{\mathrm{thres}} = 2^{-5}$, Hamming weight threshold $\mathrm{hw}_{\mathrm{thres}} = 7$ and maximum pDDT size 2^{30}. From the table it can be seen that for certain number of rounds Alg. 1 significantly improves the probabilities found with threshold search.

The execution times of Alg. 1 for different number of rounds are shown in Table 4. Most of the measurements were done on a PC with Intel® Core™ E5-2637 CPU 3.50GHz. Exceptions are the results for more than 7 rounds and block sizes larger than 48 bits, which were obtained using a parallel version of Alg. 1 executed on the HPC cluster of the University of Luxembourg [29]. The memory requirements in all cases are negligible.

A final note on the search strategy used for obtaining the times in Table 4: when searching for the best probability for n rounds, we initialize the bound $\overline{B}_n$ to the best probability for $(n-1)$ rounds: $\overline{B}_n \leftarrow B_{n-1}$. If no trail with this probability is found, the bound is decreased by a factor of 2: $\overline{B}_n \leftarrow \overline{B}_n/2$. This process continues until a trail with probability equal to the bound is found. Thus

Table 4. Probabilities and running times for the best (under the Markov assumption) differential trails for SPECK obtained with Algorithm 1 ($\log_2$ scale). Platforms: Intel® Core™ E5-2637 CPU 3.50GHz or HPC cluster for $\geq$ 7 rounds. The column t provides the time needed to find a single best trail in s/m/h/d = seconds/minutes/hours/day, where 1 day = 24 hours. Note: times are rounded up.

# R	SPECK32	t	SPECK48	t	SPECK64	t	SPECK96	t	SPECK128	t
1	0	0s	0	0s	0	0s	0	0s	0	0s
2	−1	0s	−1	0s	−1	0s	−1	0s	−1	0s
3	−3	0s	−3	0s	−3	0s	−3	0s	−3	0s
4	−5	0s	−6	0s	−6	0s	−6	6s	−6	22s
5	−9	0s	−10	1s	−10	1m	−10	5m	−10	26m
6	−13	1s	−14	3s	−15	26m	−15	5h	−15	2d
7	−18	1m	−19	1m	−21	4h	−21	5d	−21	3h
8	−24	34m	−26	9m	−27	22h	< -27	3d	< -26	2d
9	−30	12m	−33	7d	< -31	1d				
10	−34	6m	< -34	3h						

the times shown in Table 4 are measured from the start of the program to the moment when the first trail is found.

5.4 Towards Security Bounds for SPECK

The results from Table 4 can be used to trivially obtain upper bounds (under the Markov assumption) on the security of SPECK against single-trail differential cryptanalysis. For example, given the probability p_r of the best trail on r rounds and the probability p_s of the best trail on s rounds, the product $p_r p_s$ gives an upper bound on the probability of any trail on $r + s$ rounds. The latter is equivalent to the statement that any trail on $r + s$ rounds has probability at least $p_r p_s$ or lower. We use this approach to compute upper bounds (under the Markov assumption) on the probabilities of the best trails on all versions of SPECK. The results are shown in Table 5.

In view of the probabilities of the best found trails on SPECK reported in [4, Sect. 6, Table 6], the bounds in Table 5 are not tight.

5.5 On the Choice of Rotation Constants

We investigated the way in which the choice of the rotation constants r_1 and r_2 (see Fig. 1 (Left)) of SPECK32, SPECK48 and SPECK64 influences the DP of the best trails. For that purpose we assume that the exact values of the constants are not as important as their relative difference. Under this assumption, we fixed r_2 to its original value and varied r_1 over the first 16 possibilities. For each choice, we determined the probability of the best differential trail for 9 rounds

Table 5. Upper bounds on the best (under the Markov assumption) probabilities of differential trails on Speck computed using the best probabilities from Table 4 ($\log_2$ scale).

Instance	Upper Bound	Rounds	Upper Bound	Rounds	Upper Bound	Rounds
Speck32	−69	22				
Speck48	−72	22	−76	23		
Speck64	−91	26	−96	29		
Speck96	−90	28	−94	29		
Speck128	−104	32	−104	33	−105	34

Table 6. Best differential probabilities (DP) for 9 rounds of Speck32, 7 rounds of Speck48 and 6 rounds of Speck64 for 16 choices of the rotation constant r_1 with r_2 fixed to its original value (Fig. 1 (Left)) ($\log_2$ scale).

r_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Speck32	−21	−25	−24	−30	−27	−30	−25	−30	−24	−31	−26	−29	−27	−27	−22	−24
Speck48	−11	−15	−20	−16	−21	−21	−19	−21	−19	−17	−19	−20	−13	−21	−20	−19
Speck64	−8	−13	−15	−13	−15	−15	−14	−15	−15	−15	−15	−15	−15	−13	−13	−15

of Speck32, 7 rounds of Speck48 and 6 rounds of Speck64 using Alg. 1. The results are presented in Table 6.

From Table 6 it can be seen that the original choice of rotation constants: $r_1 = 7$ and $r_2 = 2$ for Speck32 and $r_1 = 8$ and $r_2 = 3$ for Speck48 is not optimal w.r.t. the probability of the best differential trail. In the former case, it results in DP of 2^{-30} over 9 rounds, while the optimal choice: $r_1 = 9$ and $r_2 = 2$ results in probability 2^{-31}. In the latter case, the original rotation constants $(8, 3)$ result in DP of 2^{-19} over 7 rounds, while the choices $(4, 3)$, $(5, 3)$, $(7, 3)$ and $(13, 3)$ result in lower probability 2^{-21}. This may hint that we have found better rotation constants for Speck. To be certain however, similar experiments for the linear case must also be conducted. In addition, the implementation cost of each pair of constants must be taken into account. Therefore the optimal choice of r_1 and r_2 requires further investigation.

6 MARX and Speckey: ARX Primitives with Provable Bounds

A limitation of Algorithm 1 is that its complexity significantly increases with the number of rounds and word sizes as indicated by Table 4. To address this problem, in this section we propose two new primitives – MARX and Speckey

for which it is feasible to compute the probabilities and linear correlations of the best trails for any number of rounds and which satisfy the Markov assumption. Both primitives have 32-bit state and 32-bit round key.

MARX (from MIX + ARX) is based on the round function of Threefish-256 [12] (with its basic component – the MIX operation) with 8-bit words. This round function is wrapped within a key addition on the input and on the output and is iterated over a fixed number of rounds. SPECKEY, as the name suggests, is based on block cipher SPECK. More precisely, it is SPECK32 with modified key addition. The round functions of MARX and SPECKEY are shown on Fig. 2.

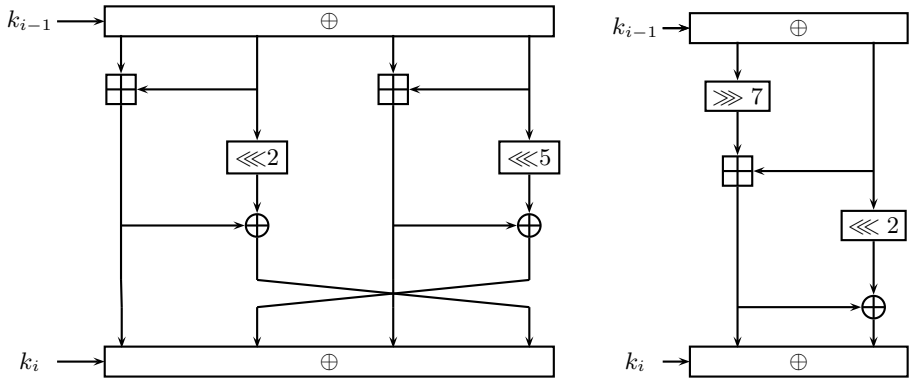

Fig. 2. Left: MARX (from MIX + ARX), based on the round function of Threefish-256 [12] with 32-bit state, 32-bit round key and 8-bit words; **Right:** SPECKEY – a variant of SPECK32 with modified key addition.

To choose the rotation constants of MARX, we exhaustively searched all possible pairs of constants and applied Alg. 1 and its linear search version to the resulting variants. Based on the results we selected the constants $r_1 = 2$ and $r_2 = 5$, as they provided differential probability (DP) $\leq 2^{-32}$ and absolute linear correlation (LC) $\leq 2^{-17}$ over a minimal number of rounds, namely 12. As to the word permutation, before settling for the one used in Threefish-256, we also considered a Feistel-like variant in which the words are circularly rotated right by one. However this variant required more rounds to reach full diffusion (best DP 2^{-32} and best absolute LC 2^{-17}) compared to Threefish-256 – on average two more rounds were necessary.

The best DP and LC of MARX and SPECKEY are shown in Table 7.

The main advantage of MARX and SPECKEY over SPECK32 is that due to the full state key addition at the beginning of every round, these two primitives belong to the class of key-alternating ciphers [9, § 5.1, Definition 2], which is a sub-class of Markov ciphers and therefore satisfies the Markov assumption. In addition, due to the 8 bit modular addition, MARX may be a more suitable choice

Table 7. Best differential probabilities (DP) and absolute linear correlations (LC) of MARX and SPECKEY ($\log_2$ scale).

# R	1	2	3	4	5	6	7	8	9	10	11	12
$\mathrm{DP}_{\mathrm{MARX}}$	−0	−0	−1	−2	−5	−9	−14	−20	−25	−29	−32	−34
$\mathrm{LC}_{\mathrm{MARX}}$	−0	−0	−0	−1	−2	−4	−7	−10	−13	−15	−16	−17
$\mathrm{DP}_{\mathrm{Speckey}}$	−0	−1	−3	−5	−9	−13	−18	−24	−30	−34		
$\mathrm{LC}_{\mathrm{Speckey}}$	−0	−0	−1	−3	−5	−7	−9	−12	−14	−17		

for devices with 8-bit microprocessors. A disadvantage is that MARX needs two more rounds to achieve full diffusion compared to SPECK32 (see Table 7) and that both MARX and SPECKEY use more operations per round compared to SPECK32. In Appendix 9 is described a variant of MARX, called MARX2, that achieves full diffusion in the same number of rounds as SPECK32 at the expense of two additional rotation operations.

Finally, we stress that the proposed new primitives are intended to serve mainly as an example of how the best trail search algorithms can be used to design new ARX constructions with provable properties. At present, MARX and SPECKEY have not undergone sufficient analysis against other cryptanalytic techniques for us to have enough confidence in their cryptographic properties.

7 Conclusion

In this paper we proposed for the first time an adaptation of Matsui's algorithm for finding the best differential and linear trails in ARX ciphers. We showed the practical application of the new method on reduced round variants of block ciphers from the SPECK family and we reported the first provably best differential trails on these variants. The new results were used to compute the first bounds (under the Markov assumption) on the security of SPECK against single-trail differential cryptanalysis. In addition, we also reported better choices of the rotation constants for SPECK w.r.t. single-trail differential cryptanalysis. Finally, we proposed two new ARX primitives – MARX and SPECKEY – which satisfy the Markov assumption and have provable bounds against single-trail differential and linear cryptanalysis – a long standing open problem in the area of ARX design. The source code of the tools for best trail search for SPECK, SPECKEY and MARX is publicly available as part of the YAARX Toolkit [30] and a snapshot of the source tree is uploaded on the CryptoLUX website [31].

Acknowledgments

We thank our colleagues from the Laboratory of Algorithmics, Cryptology and Security (LACS) at the University of Luxembourg for the stimulating discus-

sions. Some of the experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [29] – see `http://hpc.uni.lu`.

References

1. J. Aumasson, P. Jovanovic, and S. Neves. NORX: Parallel and Scalable AEAD. In M. Kutylowski and J. Vaidya, editors, *ESORICS 2014*, volume 8713 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2014.
2. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. `http://eprint.iacr.org/`.
3. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.
4. A. Biryukov, A. Roy, and V. Velichkov. Differential Analysis of Block Ciphers SIMON and SPECK. In C. Cid and C. Rechberger, editors, *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 546–570. Springer, 2014.
5. A. Biryukov and V. Velichkov. Automatic Search for Differential Trails in ARX Ciphers. In J. Benaloh, editor, *CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 227–250. Springer, 2014.
6. CryptoLUX. FELICS - Fair Evaluation of Lightweight Cryptographic Systems. `https://www.cryptolux.org/index.php/FELICS`, 2015.
7. J. Daemen and V. Rijmen. AES and the Wide Trail Design Strategy. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 108–109. Springer, 2002.
8. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
9. J. Daemen and V. Rijmen. Probability distributions of Correlation and Differentials in Block Ciphers. *IACR Cryptology ePrint Archive*, 2005:212, 2005.
10. S. M. Dehnavi, A. M. Rishakani, and M. R. M. Shamsabad. A More Explicit Formula for Linear Probabilities of Modular Addition Modulo a Power of Two. Cryptology ePrint Archive, Report 2015/026, 2015. `http://eprint.iacr.org/`.
11. C. E. Dobraunig, M. Eichlseder, and F. Mendel. Heuristic tool for linear cryptanalysis with applications to caesar candidates. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 490 – 509. Springer, 2015.
12. N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker. The Skein Hash Function Family. Submission to the NIST SHA-3 Competition (Round 2), 2009.
13. K. Fu, M. Wang, Y. Guo, S. Sun, and L. Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. Fast Software Encryption, 23rd International Workshop, FSE 2016, Bochum, Germany, March 20–23 (to appear), 2016.
14. X. Lai and J. L. Massey. Markov Ciphers and Differentail Cryptanalysis. In D. W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991.
15. G. Leurent. Analysis of Differential Attacks in ARX Constructions. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2012.

16. G. Leurent. Construction of Differential Characteristics in ARX Designs Application to Skein. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 241–258. Springer, 2013.
17. H. Lipmaa and S. Moriai. Efficient Algorithms for Computing Differential Properties of Addition. In M. Matsui, editor, *FSE*, volume 2355 of *LNCS*, pages 336–350. Springer, 2001.
18. M. Matsui. On Correlation Between the Order of S-boxes and the Strength of DES. In A. D. Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994.
19. M. Matsui and A. Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In *EUROCRYPT*, pages 81–91, 1992.
20. K. A. McKay and P. L. Vora. Analysis of ARX Functions: Pseudo-linear Methods for Approximation, Differentials, and Evaluating Diffusion. Cryptology ePrint Archive, Report 2014/895, 2014. `http://eprint.iacr.org/`.
21. N. Mouha and B. Preneel. Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328, 2013. `http://eprint.iacr.org/`.
22. N. Mouha, V. Velichkov, C. De Cannière, and B. Preneel. The Differential Analysis of S-Functions. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 36–56. Springer, 2010.
23. N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In C. Wu, M. Yung, and D. Lin, editors, *Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.
24. National Institute of Standards, U.S. Department of Commerce. *FIPS 47: Data Encryption Standard*, 1977.
25. NIST. Lightweight Cryptography Workshop 2015. `http://www.nist.gov/itl/csd/ct/lwc_workshop2015.cfm`, July 2015.
26. K. Nyberg and J. Wallén. Improved linear distinguishers for SNOW 2.0. In *Fast Software Encryption*, pages 144–162. Springer, 2006.
27. E. Schulte-Geers. On CCZ-equivalence of Addition mod 2^n. *Des. Codes Cryptography*, 66(1-3):111–127, 2013.
28. S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.
29. S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos. Management of an Academic HPC Cluster: The UL Experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, pages 959–967, Bologna, Italy, July 2014. IEEE.
30. V. Velichkov. YAARX: Yet Another Toolkit for the Analysis of ARX Cryptographic Algorithms. Laboratory of Algorithmics, Cryptology and Security (LACS), University of Luxembourg, 2013–2016. `https://github.com/vesselinux/yaarx` .
31. V. Velichkov and Y. L. Corre. Tool for Searching for Optimal Trails in ARX. Laboratory of Algorithmics, Cryptology and Security (LACS), University of Luxembourg, 2016. `https://www.cryptolux.org` .
32. J. Wallén. Linear Approximations of Addition Modulo 2^n. In T. Johansson, editor, *FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 261–273. Springer, 2003.

33. J. Wallén. On the Differential and Linear Properties of Addition. Master's thesis, Helsinki University of Technology, 2003.
34. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In *Advances in Cryptology–EUROCRYPT 2005*, pages 1–18. Springer, 2005.
35. X. Wang, Y. L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.
36. X. Wang and H. Yu. How to break MD5 and other hash functions. In *Advances in Cryptology–EUROCRYPT 2005*, pages 19–35. Springer, 2005.
37. Y. Yao, B. Zhang, and W. Wu. Automatic search for linear trails of the SPECK family. In *Information Security*, pages 158–176. Springer, 2015.

Appendix

8 Showing that Speck is not a Markov Cipher

In this section we show, by the means of a counter example, that Speck is *not* a Markov cipher. For the purpose, we use an equivalent representation of the round function of Speck (Fig. 1 (left)), shown on Fig. 3 (left).

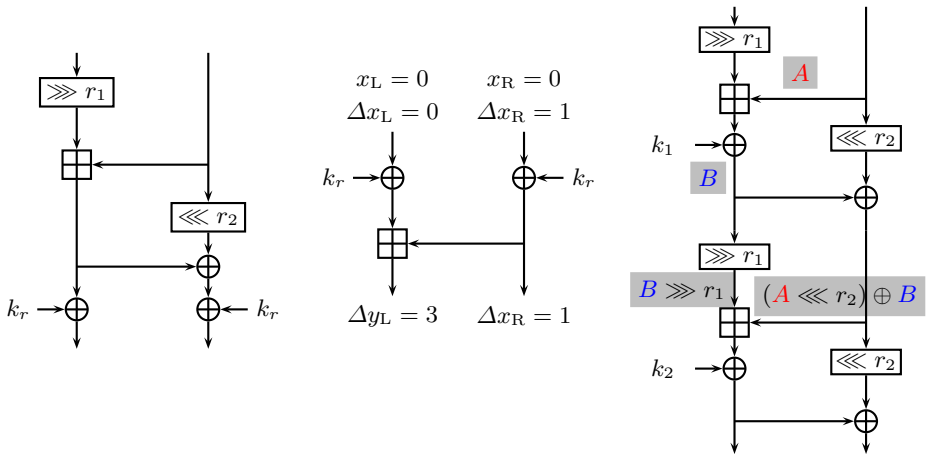

Fig. 3. Left: Equivalent representation of one round of Speck. **Middle:** Main non-linear component of Speck, illustrating two differentials that differ in probability and hence contradict the Markov assumption: $P((3,1)|(0,1),(0,0)) = 2^{-1}$ and $P((3,1)|(0,1)) = 2^{-2}$. **Right:** Dependency between the inputs to the addition in one round (B) and between the inputs to consecutive rounds (A).

According to the formal definition [14, Sect. 3], a Markov cipher is an iterative cipher, whose round function is such that its differential probability is

independent of the input values, under an appropriate definition of a difference. More formally, let f be the round function of an iterative cipher and let x and y be resp. an input and output state and k be the round key: $y = f(x, k)$. Let Δx denote an XOR difference between two input values x and x^*: $\Delta x = x \oplus x^*$. Finally, let $\underset{k}{P}$ and $\underset{x,k}{P}$ denote the differential probability of f resp. over all round keys k and over all input values and round keys (x, k). Then, if a cipher is Markov, the two probabilities should be equal:

$$\underset{k}{P}(\Delta y|\Delta x, x) = \underset{x,k}{P}(\Delta y|\Delta x) \ . \tag{7}$$

In other words, for a Markov cipher the differential probability of f is independent of the input values x for all x (when the subkey is uniformly random).

To show that SPECK is not a Markov cipher, it is enough to provide values for x, Δx and Δy for which equation (7) does not hold. To do this, we use the main non-linear component of SPECK shown on Fig. 3 (middle). For this component, for the following values $x = (x_{\mathrm{L}}, x_{\mathrm{R}}) = (0, 0)$, $\Delta x = (\Delta x_{\mathrm{L}}, \Delta x_{\mathrm{R}}) = (0, 1)$ and $\Delta y = (\Delta y_{\mathrm{L}}, \Delta y_{\mathrm{R}}) = (3, 1)$ the two probabilities in (7) are not equal:

$$\begin{aligned} &\underset{k}{P}(\Delta y = (3,1)|\Delta x = (0,1), x = (0,0)) = 2^{-1} \\ \neq &\underset{x,k}{P}(\Delta y = (3,1)|\Delta x = (0,1)) = 2^{-2} \ . \end{aligned} \tag{8}$$

An illustrative example of the dependency between the inputs to the addition operation in one round and between the inputs to consecutive rounds is shown on Fig. 3 (right).

9 MARX2: A Variant of MARX with Improved Diffusion

In this section we describe MARX2 – a variant of MARX with improved diffusion. MARX2 has two additional rotation operations and its round function is depicted in Fig. 4.

As can be seen from Fig. 4, MARX2 is composed of two parallel applications of the round function of SPECK with 8-bit words. Due to the additional rotation operations it achieves full diffusion in the same number of rounds as SPECK32, namely 10. The best DP and LC for the recommended rotation amounts $(r_1, r_2, r_3, r_4) = (2, 3, 1, 7)$ are shown in Table 8.

The rotation constants of MARX2 have been chosen by exhaustively searching over all four rotation values (4095 values in total, excluding the all-zero choice). For each set of amounts we applied Alg. 1 and its linear search variant and recorded the number of rounds necessary to reach full diffusion. The results show that no set of rotation constants exists for which full diffusion can be reached in less than 10 rounds. From the constants that ensure diffusion in 10 rounds we have selected $(r_1, r_2, r_3, r_4) = (2, 3, 1, 7)$ as the recommended choice since for this set we get slightly better DP than SPECK32 (2^{-35} vs. 2^{-34}). In addition, all constants from the set are different and are not multiples of each

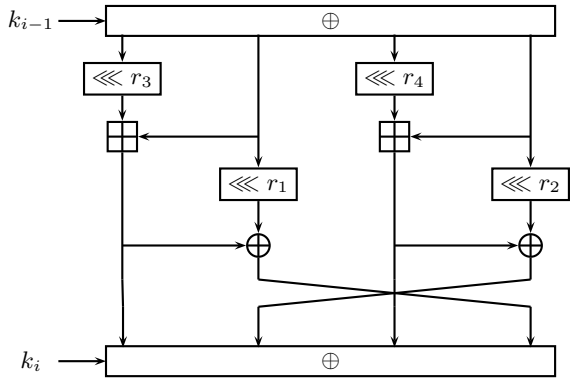

Fig. 4. MARX2: a variant of MARX with two additional rotations for improved diffusion. The recommended rotation amounts are $(r_1, r_2, r_3, r_4) = (2, 3, 1, 7)$

Table 8. Best differential probabilities (DP) and absolute linear correlations (LC) of MARX2 with rotation constants $(r_1, r_2, r_3, r_4) = (2, 3, 1, 7)$ – see Fig. 4 ($\log_2$ scale).

# R	1	2	3	4	5	6	7	8	9	10
$\mathrm{DP}_{\mathrm{MARX2}}$	-0	-1	-3	-5	-11	-16	-22	-25	-29	-35
$\mathrm{LC}_{\mathrm{MARX2}}$	-0	-0	-1	-3	-5	-8	-10	-13	-15	-17

other, which may also be considered desirable properties. Other choices that also result in full diffusion for 10 rounds are: $(2, 3, 7, 2)$, $(2, 3, 1, 2)$ and $(5, 5, 2, 7)$. Note that the constants (r_1, r_2, r_3, r_4) and (r_2, r_1, r_4, r_3) are equivalent.

Finally, we note that MARX2 is an illustration of another strategy for increasing the block size of an ARX cipher. Rather than increasing the word size from N to $2N$, as is done in Speck, in order to increase the block size the designer may alternatively double the number of N-bit block components as in MARX2. This approach may result in improved efficiency on some platforms, such as e.g. 32-bit ARM, where the cost of a bit rotation is constant.