Rotational Cryptanalysis in the Presence of Constants

Tomer Ashur[1] and Yunwen Liu[1,2]*

[1] Dept. Electrical Engineering (ESAT), KU Leuven and iMinds, Leuven, Belgium
[2] College of Science, National University of Defense Technology, Changsha, China
[tomer.ashur,yunwen.liu]@esat.kuleuven.be

Abstract. Rotational cryptanalysis is a statistical method for attacking ARX constructions. It was previously shown that ARX-C, *i.e.*, ARX with the injection of constants can be used to implement any function. In this paper we investigate how rotational cryptanalysis is affected when constants are injected into the state. We introduce the notion of an RX-difference, generalizing the idea of a rotational difference. We show how RX-differences behave around modular addition, and give a formula to calculate their transition probability. We experimentally verify the formula using Speck32/64, and present a 7-round distinguisher based on RX-differences. We then discuss two types of constants: round constants, and constants which are the result of using a fixed key, and provide recommendations to designers for optimal choice of parameters.

Keywords: Rotational cryptanalysis, ARX, RX-difference

1 Introduction

The Addition-Rotation-XOR (ARX) structure is a common design for symmetric-key primitives. The popularity of the structure stems from the fact that by using only three operations, namely addition modulo 2^n, cyclic rotation, and XOR, good confusion and diffusion can be achieved. Examples to the large variety of ARX-based cryptographic primitives include two of the five SHA-3 competition finalists Skein [12] and BLAKE [2], the stream ciphers Salsa20 [5] and ChaCha [4], block ciphers such as TEA [26], XTEA [22] and Speck [3], and the MAC algorithm Chaskey [20].

Like with many other structures, differential cryptanalysis [6] and linear cryptanalysis [19] are the two main approaches for the analysis of ARX-based designs, and a number of heuristic search tools towards finding differential characteristics and linear trails are proposed, for example, [7, 9, 17, 18]. Along with these two general methods, rotational cryptanalysis, a dedicated method for analyzing ARX constructions, drew a lot of attention since its publication. Although the idea of tracking rotational input pairs can be found before being formalized,

* Corresponding author

see for example, [11, 25], rotational cryptanalysis was first proposed as a generic attack technique against ARX-based designs by Khovratovich and Nikolić in 2010 [13] and applied to Threefish, the ARX-based block cipher underlying the hash function Skein. Most notably, the rotational rebound attack [15], which is an extension of rotational cryptanalysis, is by far the best attack against Skein, to date.

In 2015, Khovratovich et al. showed some inaccuracies in the application of the technique [14]. More specifically, they pointed out that the independence assumption in [13] does not hold when an output of a modular addition is directly given as input to another modular addition. They refer to this event as a "chained modular addition", and show that when such a chain exists, the transition probability over both additions is not independent. The latter result does not invalidate the rotational rebound attack since the probability was estimated experimentally. However it does show that further research is needed before rotational cryptanalysis is fully understood.

Similar to the modular chains, another issue that was not rigorously analyzed in [13] and its subsequent works, is the injection of constants. The impact of constants to ARX systems is noticed in, for example, that the designers of the block cipher SEA [25] assert that their construction can resist rotational cryptanalysis due to the nonlinear key schedule and the injection of pseudo-random constants. When an ARX structure includes the injection of constants it is called ARX-C, and it was proven in [13] that this structure is complete, *i.e.*, that any function can be implemented through an ARX-C construction. In most papers on rotational cryptanalysis, heuristic experiments are made to address the influence of constants.

In this paper, we present a novel way to compute the rotational probability of a pair of inputs when constants are injected into the state. We do so by combining the propagation rules of differences through modular addition, with those of rotational cryptanalysis. Informally speaking, when applying rotational cryptanalysis to an ARX primitive with the XOR of constants, it can be regarded as a merge of rotational cryptanalysis and differential cryptanalysis. We verify our results empirically using Speck32/64 and present a 7-round distinguisher based on this technique. As a result, we can propose countermeasures against rotational cryptanalysis, which can serve as guidelines for future ARX designs.

The rest of this paper is organized as follows: in Section 2 we describe our notation, and briefly recall the basis of rotational cryptanalysis. A closed formula for calculating the rotational probability in ARX-C is presented in Section 3. In Section 4 we experimentally verify our results, and discuss possible countermeasures. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Notations

We present our notations in Table 1.

Table 1. The notations used throughout the paper.

$x = (x_{n-1}, \cdots, x_1, x_0)$	An n-bit boolean vector; x_0 is the least significant bit
$x_i \wedge y_i$	The bitwise AND operation between the bit in x_i and the bit in y_i
$x\|\|y$	The concatenation of x and y
$\|x\|$	The hamming weight of the boolean vector x
$SHL(x)$	A non-cyclic left shift of x by one bit
$x\|y$	The vector bitwise OR operation
$x \lll \gamma$	A cyclic left shift of x by γ bits
$\overleftarrow{x}$	$x \lll 1$
$1_{x \preccurlyeq y}$	A characteristic function which evaluates to 1 if and only if $x_i \leq y_i$ for all i
1_i	The ith bit of the n-bit string $0 \ldots 01$, *i.e.*, $1_i = 1$ for $i = 0$ and 0 otherwise
$(I \oplus SHL)(x)$	$x \oplus SHL(x)$
$L(x)^*$	The γ most significant bits of x
$R(x)^*$	The $n - \gamma$ least significant bits of x
$R'(x)^*$	The γ least significant bits of x
$L'(x)^*$	The $n - \gamma$ most significant bits of x

* Note that $x = L(x)||R(x) = L^{'}(x)||R^{'}(x)$.

2.2 Rotational Cryptanalysis

Similar to differential cryptanalysis, rotational cryptanalysis takes advantage of the high probability in the propagation of rotational pairs $(x, x \lll \gamma)$ through the ARX operations. The following proposition provides a general way to compute the propagation of a rotational pair through the modular addition:

Proposition 1 ([10]). *For $x, y \in \mathbb{F}_{2^n}$, and $0 < \gamma < n$,*

$$\Pr[(x \boxplus y) \lll \gamma = (x \lll \gamma) \boxplus (y \lll \gamma)] = (1 + 2^{\gamma - n} + 2^{-\gamma} + 2^{-n})/4.$$

The probability is maximized to $2^{-1.415}$ when n is large and $\gamma = 1$.

Whenever the two inputs to the round modular addition are independent and uniformly distributed, the probabilities of consecutive modular sums can be directly multiplied. However, as was shown in [14], if a modular chain exists, the probability requires an adjustment to the formula, and the resulting probability is in fact smaller. A similar effect was noticed for linear cryptanalysis in [23], and for differential cryptanalysis in [16].

2.3 Description of SPECK

Speck is a family of lightweight block ciphers designed by the NSA in 2013 [3]. A member of the family is denoted by Speck$2n/mn$, where the block size is $2n$ for $n \in \{16, 24, 32, 48, 64\}$, and the key size is mn for $m \in \{2, 3, 4\}$, depending on the desired security.

The round function of Speck receives two words $x^{(i)}$ and $y^{(i)}$, and a round key k_i, all of size n, and outputs two words of size n, $x^{(i+1)}$ and $y^{(i+1)}$, such that

$$(x^{(i+1)}, y^{(i+1)}) = F_{k_i}(x^{(i)}, y^{(i)}) = (f_{k_i}(x^{(i)}, y^{(i)}), f_{k_i}(x^{(i)}, y^{(i)}) \oplus (y^{(i)} \lll \beta)),$$

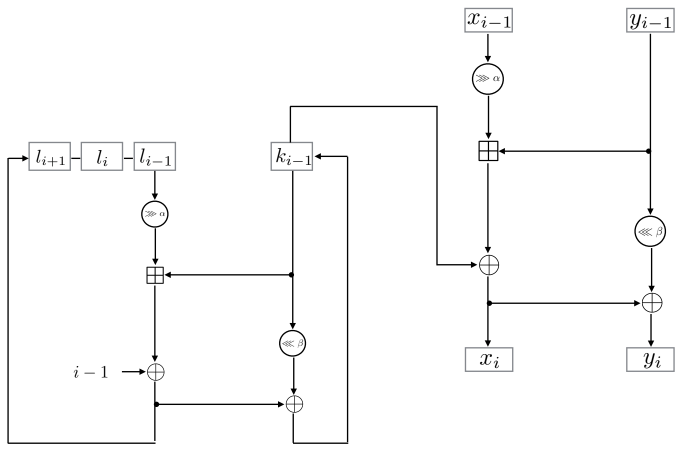

Fig. 1. One round of SPECK

where $f_{k_i}(\cdot,\cdot)$ is

$$f_{k_i}(x^{(i)}, y^{(i)}) = ((x^{(i)} \ggg \alpha) \boxplus y^{(i)}) \oplus k_i.$$

The SPECK key schedule algorithm uses the same round function to generate the round keys. Let $K = (l_{m-2}, ..., l_0, k_0)$ be a master key for SPECK$2n$, where $l_i, k_0 \in \mathbb{F}_{2^n}$. The sequence of round keys k_i is generated as

$$k_{i+1} = f_{ct}(l_i, k_i) \oplus (k_i \lll \beta)$$

for

$$l_{i+m-1} = f_{ct}(l_i, k_i),$$

with $ct = i$ the round number starting from 0.

The rotation offset (α, β) is $(7, 2)$ for SPECK32, and $(8, 3)$ for the larger versions. A single round of SPECK with $m = 4$ is depicted in Figure 1. For more details, we refer the interested reader to the original design [3].

With regards to cryptanalysis, the longest distinguishers for SPECK32/64 are a linear distinguisher for 9 rounds with correlation 2^{-14} due to Yao et al. published in [27], and a 9-round differential distinguisher with probability 2^{-30} due to Biyukov et al. published in [8].

3 Rotational Cryptanalysis in the Presence of Constants

In [13], it was shown that ARX-C, *i.e.*, an ARX construction with constants, is complete. This means that any function can be implemented using the ARX-C operations. In most cases, the constants are injected into the state either through an XOR operation or through modular addition. When the constant c is rotational-invariant, *i.e.*, $c = c \lll \gamma$, for some γ, XORing with c does not change the rotational property of a rotational pair $(x, x \lll \gamma)$. However, whenever c is not rotational-invariant, the properties of the output require further inspection.

In general, when a constant c that is not rotational-invariant is XORed into a rotational pair $(x, x \lll \gamma)$, the output pair $(x \oplus c, (x \lll \gamma) \oplus c)$, no longer form a rotational pair. If this pair is given as an input to the modular addition, the basic formula in Proposition 1 for computing the propagation of the rotational property can no longer be used.

In the sequel, we define a $((a_1, a_2), \gamma)$-*rotational-XOR-difference* (or in shorthand notation $((a_1, a_2), \gamma)$-RX-difference and RX-difference when $(a_1, a_2), \gamma$ are clear), to be a rotational pair with rotation γ under translations a_1 and a_2, *i.e.*, $(x \oplus a_1, (x \lll \gamma) \oplus a_2)$; we call such a pair an RX-pair. Note that when $a_1 = a_2 = 0$, they simply become a rotational pair. Our goal is to estimate the transition probability with respect to modular addition of two input RX-differences, to an output RX-difference. Without loss of generality, we consider the case where the input rotational pairs are $(x \oplus a_1, y \oplus b_1)$ and $(\overleftarrow{x} \oplus a_2, \overleftarrow{y} \oplus b_2)$, and compute the probability of $\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1)} \oplus \Delta_1 = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2$.

Theorem 1. *Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. Let $a_1, b_1, a_2, b_2, \Delta_1, \Delta_2$ be constants in $\mathbb{F}_{2^n}$. Then,*

$$\begin{aligned}
&\Pr[\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1)} \oplus \Delta_1 = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2] \\
&= 1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \oplus 1 \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-3} \\
&+ 1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-1.415},
\end{aligned} \tag{1}$$

where

$$\delta_1 = R(a_1) \oplus L^{'}(a_2),$$

$$\delta_2 = R(b_1) \oplus L^{'}(b_2),$$

and

$$\delta_3 = R(\Delta_1) \oplus L^{'}(\Delta_2).$$

Note that when all the constants are 0, *i.e.*, $a_1 = a_2 = b_1 = b_2 = \Delta_1 = \Delta_2 = 0$, Theorem 1 predicts $\Pr[\overleftarrow{x \boxplus y} = \overleftarrow{x} \boxplus \overleftarrow{y}]$, which is the normal case for rotational cryptanalysis.

Before moving to prove Theorem 1, we introduce the following two lemmata:

Lemma 1 ([24]). *Let $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{F}_{2^n}$ be constants. Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. The probability of the differential equation*

$$x \boxplus y = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3 \tag{2}$$

is

$$1_{(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \preceq SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}. \tag{3}$$

Proof. The complete proof can be found in [24].

The following example is provided for a better understanding of Lemma 1.

Example 1. Let $n = 8$, $\zeta_1 = \mathtt{E}_{16}, \zeta_2 = \mathtt{9}_{16}$ and $\zeta_3 = \mathtt{F7}_{16}$, we have

$$(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) = \mathtt{10}_{16},$$
$$SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3)) = \mathtt{FE}_{16},$$

and

$$|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))| = |\mathtt{FE}_{16}| = 7.$$

We evaluate the characteristic function $1_{(I\oplus SHL)(\zeta_1\oplus\zeta_2\oplus\zeta_3)\preccurlyeq SHL((\zeta_1\oplus\zeta_3)|(\zeta_2\oplus\zeta_3))}$, and see that it is equal to 1 since no bit in $(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3)$ is larger than the respective bit in $SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))$. The probability is then computed to be $2^{-|SHL((\zeta_1\oplus\zeta_3)|(\zeta_2\oplus\zeta_3))|} = 2^{-7}$.

Lemma 2. *Let $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{F}_{2^n}$ be constants. For independent random variables $x, y \in \mathbb{F}_{2^n}$, the probability of*

$$x \boxplus y \boxplus 1 = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3 \tag{4}$$

is

$$1_{(I\oplus SHL)(\zeta_1\oplus\zeta_2\oplus\zeta_3)\oplus 1\preceq SHL((\zeta_1\oplus\zeta_3)|(\zeta_2\oplus\zeta_3))} \cdot 2^{-|SHL((\zeta_1\oplus\zeta_3)|(\zeta_2\oplus\zeta_3))|}. \tag{5}$$

Proof. Let c be the carry vector of $x \boxplus y$ (*i.e.*, $c = (x \boxplus y) \oplus x \oplus y$), z the output vector of $x \boxplus y$ (*i.e.*, $z = x \boxplus y$), h the carry vector of $z \boxplus 1$ (*i.e.*, $h = (z \boxplus 1) \oplus z \oplus 1$), and $e = c \oplus 1 \oplus h$. Then,

$$x \boxplus y \boxplus 1 = (x \oplus y \oplus c) \boxplus 1 = x \oplus y \oplus c \oplus 1 \oplus h = x \oplus y \oplus e.$$

Denote the i-th bit of the binary expansion of 1 by 1_i. Then the i-th bit of h, h_i can be computed recursively through

$$h_i = \begin{cases} 0 & i = 0, \\ z_{i-1} \wedge h_{i-1} \oplus z_{i-1} \wedge 1_{i-1} \oplus h_{i-1} \wedge 1_{i-1} & i > 0. \end{cases}$$

We get that,

$$h_i = \begin{cases} 0 & i = 0, \\ z_0 \wedge h_0 \oplus z_0 \wedge 1_0 \oplus h_0 \wedge 1_0 = x_0 \oplus y_0 & i = 1, \\ z_{i-1} \wedge h_{i-1} \oplus z_{i-1} \wedge 1_{i-1} \oplus h_{i-1} \wedge 1_{i-1} = (x_{i-1} \oplus y_{i-1} \oplus c_{i-1}) \wedge h_{i-1} & i > 1. \end{cases}$$

Hence $e_0 = c_0 \oplus 1 \oplus h_0 = 1$ and

$$\begin{aligned} e_{i+1} &= c_{i+1} \oplus h_{i+1} \\ &= x_i \wedge y_i \oplus x_i \wedge c_i \oplus y_i \wedge c_i \oplus (x_i \oplus y_i \oplus c_i) \wedge h_i \\ &= x_i \wedge y_i \oplus x_i \wedge (c_i \oplus h_i) \oplus y_i \wedge (c_i \oplus h_i) \oplus c_i \wedge h_i \\ &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \oplus (x_{i-1} \wedge y_{i-1} \oplus x_{i-1} \\ &\qquad \wedge c_{i-1} \oplus y_{i-1} \wedge c_{i-1}) \wedge (x_{i-1} \oplus y_{i-1} \oplus c_{i-1}) \wedge h_{i-1} \\ &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \oplus x_{i-1} \wedge y_{i-1} \wedge c_{i-1} \wedge h_{i-1} \\ &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \end{aligned} \tag{6}$$

The last equation holds since $c_0 \wedge h_0 = 0$ and therefore $c_i \wedge h_i = 0$ for all i. In other words, when computing $x \boxplus y \boxplus 1$ we can calculate the carry bit entering position $i+1$ as a function of x_i, y_i, and the previous carry bit, e_i by means of Equation 6. Notice that the recursive formulae for computing $c = (x \boxplus y) \oplus x \oplus y$ and $e = (x \boxplus y \boxplus 1) \oplus x \oplus y$ are similar except that they start with different initial values, we will use S-function to derive the probability of the differential equation which is analogous to the XOR-differential probability of addition [21].

An S-function updates a list of states $S[i], 0 \leq i \leq n-1$ and outputs an n-bit value b with n-bit inputs $a_1, a_2, \ldots, a_k$ by

$$(b_i, S[i+1]) = f((a_1)_i, (a_2)_i, \ldots, (a_k)_i, S[i]), \quad 0 \leq i \leq n-1$$

We define an S-function for the differential equation $x \boxplus y \boxplus 1 = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3$ as follows.

$$\begin{aligned}
(t_1)_i &= x_i \oplus y_i \oplus e_i \\
e_{i+1} &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \\
(t_2)_i &= (x_i \oplus (\zeta_1)_i) \oplus (y_i \oplus (\zeta_2)_i) \oplus g_i \\
g_{i+1} &= (x_i \oplus (\zeta_1)_i) \wedge (y_i \oplus (\zeta_2)_i) \oplus (x_i \oplus (\zeta_1)_i) \wedge g_i \oplus (y_i \oplus (\zeta_2)_i) \wedge g_i
\end{aligned} \tag{7}$$

where $e_0 = 1, g_0 = 0$. Let $S[i] = (e_i, g_i)$, the S-function is defined as

$$((t_1)_i \oplus (t_2)_i, S[i+1]) = f(x_i, y_i, (\zeta_1)_i, (\zeta_2)_i, S[i]), 0 \leq i \leq n-1 \tag{8}$$

where f follows from Equation 7. Define $\omega_i = (\zeta_1)_i || (\zeta_2)_i || (\zeta_3)_i, 0 \leq i \leq n-1$, $L = (1,1), C = (0,1)^T$ and eight 2×2 matrices as

$$A_{000} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_{001} = A_{010} = A_{100} = \frac{1}{2}\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix},$$
$$A_{111} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_{011} = A_{101} = A_{110} = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}.$$

Using the directed acyclic graph method [21], the probability of Equation 4 can be determined by $LA_{\omega_{n-1}} \cdots A_{\omega_1} A_{\omega_0} C$. From [21] we know that the probability of the differential equation $x \boxplus y = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3$ is $LA_{\omega_{n-1}} \cdots A_{\omega_1} A_{\omega_0} C'$, where $C' = (1,0)^T$. The probabilities of Equation 2 and Equation 4 can be fully determined by the same automaton as shown in the Appendix.

We first discuss the condition for the probability to be nonzero, so we omit the value $1/2$ which may occur in the multiplications for now. Since the product of L with any matrices A_j can only be $(1,0), (0,1)$ or $(0,0)$, the probability of Equation 2 (Equation 4 resp.) is nonzero when $LA_{\omega_{n-1}} \cdots A_{\omega_1} A_{\omega_0}$ equals to $(1,0)$ ($(0,1)$ resp.). Therefore the probability of Equation 4 will be nonzero when the first $n-2$ multiplications do not lead to $(0,0)$, meanwhile the $LA_{\omega_{n-1}} \cdots A_{\omega_1} = (0,1)$ and $w_0 = 1,2,4,7$, or $LA_{\omega_{n-1}} \cdots A_{\omega_1} = (1,0)$ and $w_0 = 1,2,4$. Comparing with the probability of Equation 2 in the previous lemma, the probability of Equation 4 is nonzero if the following condition is satisfied

$$(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq SHL((\zeta_1 \oplus \zeta_3) | (\zeta_2 \oplus \zeta_3)).$$

Once the probability is nonzero, it is determined by the number of times that $\omega_i \in \{001, 010, 100, 011, 101, 110\}$, and each time it contributes 1/2 to the overall probability except the first multiplication $LA_{\omega_{n-1}}$, therefore the probability is $2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}$ since it encounters such matrices when $(((\zeta_1)_i \oplus (\zeta_3)_i)|((\zeta_2)_i \oplus (\zeta_3)_i)) = 1$ for certain $i, 0 \leq i \leq n-1$. Therefore the probability of Equation 4 is given by

$$1_{(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}.$$

which leads to the conclusion. □

We can now prove Theorem 1.

Theorem 1. *Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. Let $a_1, b_1, a_2, b_2, \Delta_1, \Delta_2$ be constants in $\mathbb{F}_{2^n}$. Then*

$$\begin{aligned}&\Pr[\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2] = \\ &1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \oplus 1 \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-3} + \\ &1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-1.415}\end{aligned} \tag{9}$$

where $\delta_1, \delta_2,$ and δ_3 are as before.

Proof. Let C^1 be the carry vector of $(x \oplus a_1) \boxplus (y \oplus b_1)$ and let $C^1_{n-\gamma}$ be the carry bit in position $n - \gamma$ (*i.e.*, $C^1_{n-\gamma}$ is the most significant carry produced by $(R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))$). We write $\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1}$ from Equation 9 as the concatenation of its left and right parts.

$$\begin{aligned}&\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} \\ &= \overleftarrow{((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1)||} \\ &\qquad\qquad \overline{((R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))) \oplus R(\Delta_1)} \\ &= ((R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))) \oplus R(\Delta_1)|| \\ &\qquad\qquad ((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1).\end{aligned}$$

Similarly, let C^2 be the carry vector of $(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2)$, and C^2_γ the carry bit in position γ (*i.e.*, C^2_γ is the most significant carry produced by $((L(x) \oplus R^{'}(a_2)) \boxplus (L(y) \oplus R^{'}(b_2))$. we can write $(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2$ from Equation 9 as the concatenation of its left and right parts.

$$\begin{aligned}&(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2 \\ &= ((\overleftarrow{L(x)||R(x)}) \oplus a_2) \boxplus ((\overleftarrow{L(y)||R(y)}) \oplus b_2) \oplus \Delta_2 \\ &= ((R(x)||L(x)) \oplus (L^{'}(a_2)||R^{'}(a_2))) \boxplus ((R(y)||L(y)) \oplus (L^{'}(b_2)||R^{'}(b_2))) \oplus \Delta_2 \\ &= ((R(x) \oplus L^{'}(a_2)) \boxplus (R(y) \oplus L^{'}(b_2)) \boxplus C^2_\gamma) \oplus L^{'}(\Delta_2)|| \\ &\qquad\qquad ((L(x) \oplus R^{'}(a_2)) \boxplus (L(y) \oplus R^{'}(b_2))) \oplus R^{'}(\Delta_2).\end{aligned}$$

We get that

$$\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \mathit{\Delta}_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \mathit{\Delta}_2$$

if and only if

$$\begin{aligned} &((R(x) \oplus L^{'}(a_2)) \boxplus (R(y) \oplus L^{'}(b_2)) \boxplus C_{\gamma}^{2}) \oplus L^{'}(\mathit{\Delta}_2) = \\ &(R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1)) \oplus R(\mathit{\Delta}_1), \end{aligned} \tag{10}$$

and

$$\begin{aligned} &((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C_{n-\gamma}^{1}) \oplus L(\mathit{\Delta}_1) = \\ &((L(x) \oplus R^{'}(a_2)) \boxplus (L(y) \oplus R^{'}(b_2))) \oplus R^{'}(\mathit{\Delta}_2). \end{aligned} \tag{11}$$

Replacing

$$\begin{aligned} R(x^{\dagger}) &= R(x) \oplus L^{'}(a_2) \\ R(y^{\dagger}) &= R(y) \oplus L^{'}(b_2), \end{aligned}$$

we can rewrite Equation 10 as

$$\begin{aligned} &R(x^{\dagger}) \boxplus R(y^{\dagger}) \boxplus C_{\gamma}^{2} = \\ &(R(x^{\dagger}) \oplus R(a_1) \oplus L^{'}(a_2)) \boxplus (R(y^{\dagger}) \oplus R(b_1) \oplus L^{'}(b_2)) \oplus R(\mathit{\Delta}_1) \oplus L^{'}(\mathit{\Delta}_2). \end{aligned} \tag{12}$$

Similarly, by setting

$$\begin{aligned} L(x^{*}) &= L(x) \oplus L(a_1) \\ L(y^{*}) &= L(y) \oplus L(b_1), \end{aligned}$$

Equation 11 reduces to

$$\begin{aligned} &L(x^{*}) \boxplus L(y^{*}) \boxplus C_{n-\gamma}^{1} = \\ &((L(x^{*}) \oplus L(a_1) \oplus R^{'}(a_2)) \boxplus (L(y^{*}) \oplus L(b_1) \oplus R^{'}(b_2))) \oplus R^{'}(\mathit{\Delta}_2) \oplus L(\mathit{\Delta}_1). \end{aligned} \tag{13}$$

We can compute the probability of Equation 12 and Equation 13 by means of Lemma 1 and Lemma 2 based on the values of $C_{n-\gamma}^{1}$ and C_{γ}^{2}.

Case 1: $C_{\gamma}^{2} = 0$, the probability is the difference propagation probability and can be calculated by means of Lemma 1.

Case 2: $C_{\gamma}^{2} = 1$, we solve the differential equations using Lemma 2.

Similarly,

Case 3: $C_{n-\gamma}^{1} = 0$, the probability is the difference propagation probability and can be calculated by Lemma 1.

Case 4: $C_{n-\gamma}^{1} = 1$, we solve the differential equations using Lemma 2.

When $\gamma = 1$, $L(\cdot), R^{'}(\cdot)$ represent a single bit, hence,

$$C_{n-\gamma}^{1} = L(a_1) \oplus L(b_1) \oplus L(\mathit{\Delta}_1) \oplus R^{'}(a_2) \oplus R^{'}(b_2) \oplus R^{'}(\mathit{\Delta}_2).$$

In addition, notice that the carry bit of $L(x) \boxplus L(y)$ is independent with that of $R(x) \boxplus R(y)$ when x, y are independent random variables, we have for large n and $\gamma = 1$, $\Pr[C_\gamma^2 = 0] = 3/4$ and $\Pr[C_{n-\gamma}^1 = 0] = 1/2$, since the carry for 1-bit addition is biased, however, for the addition of two random bit strings, the most significant carry bit can be regarded as balanced. Then,

$$\begin{aligned}\Pr[C_\gamma^2 = 0, C_{n-\gamma}^1 = 0] &= 2^{-1.415}\\ \Pr[C_\gamma^2 = 0, C_{n-\gamma}^1 = 1] &= 2^{-1.415}\\ \Pr[C_\gamma^2 = 1, C_{n-\gamma}^1 = 0] &= 2^{-3}\\ \Pr[C_\gamma^2 = 1, C_{n-\gamma}^1 = 1] &= 2^{-3}.\end{aligned}$$

Therefore, the probability is calculated as

$$\begin{gathered}\Pr[C_\gamma^2 = 0, C_{n-\gamma}^1] \cdot \Pr[x \boxplus y = (x \oplus \delta_1) \boxplus (y \oplus \delta_2) \oplus \delta_3] +\\ \Pr[C_\gamma^2 = 1, C_{n-\gamma}^1] \cdot \Pr[x \boxplus y \boxplus 1 = (x \oplus \delta_1) \boxplus (y \oplus \delta_2) \oplus \delta_3],\end{gathered}$$

which concludes the proof. □

Remark 1. The above theorem shows the propagation of RX-differences through a modular addition and how to compute its probability. Given that the inputs are $(x \oplus a_1, y \oplus b_1)$ and $(\overleftarrow{x} \oplus a_2, \overleftarrow{y} \oplus b_2)$ with RX-differences $(((a_1, a_2), 1), ((b_1, b_2), 1))$. Let $z = ((x \oplus a_1) \boxplus (y \oplus b_1))$ and $z' = ((\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2))$, then the formula predicts the probability that z, z' forms a $((0, \overleftarrow{\Delta_1} \oplus \Delta_2), 1)$-RX-difference.

4 Experimental Verification

Note that the theoretical probability formula in Theorem 1 holds for any ARX-C systems, in this section we verify our results empirically using the round function of Speck32/64 as an example. Note that the distinguisher we present in this section covers only 7 rounds of Speck32/64, and is not intended to improve the cryptanalysis of Speck rather than to show that the proposed technique works in practice. We then discuss design recommendations for future ARX designs. All the necessary code for repeating the experiments described in this section can be found at [1].

4.1 Application to SPECK32/64

Using a greedy algorithm, we obtained a 6-round trail with RX-differences for the key-schedule of Speck32/64. In Table 2 we compare the probability predicted by Theorem 1 and the probability obtained by iterating all 2^{32} possible (x, y) with a fixed tuple $(a_1, b_1, \Delta_1, a_2, b_2, \Delta_2)$. As is evident from Table 2, the values match perfectly. In Table 3 we present the empirical probability of the trail over 2^{33} uniformly chosen keys. Interestingly, the probability of the full trail is lower than the one predicted by simply multiplying the round probabilities, suggesting that the left and right inputs to the round function are not independent. Nevertheless,

Table 2. A table comparing the transition probability predicted through Theorem 1 and the empirical probability for uniformly chosen x and y, and a fixed $(a_1, b_1, \Delta_1, a_2, b_2, \Delta_2)$. All RX-differences are in hexadecimal notation.

Round	a_1	b_1	Δ_1	a_2	b_2	Δ_2	Predicted Prob.	Empirical Prob.	Accumulated Prob.
1	0	0	0	0	0	0	$2^{-1.415}$	$2^{-1.415}$	$2^{-1.415}$
2	0	0	0	0	0	0	$2^{-1.415}$	$2^{-1.415}$	$2^{-2.83}$
3	0	1	0	0	1	2	$2^{-2.415}$	$2^{-2.415}$	$2^{-5.245}$
4	0	2	6	0	0	8	$2^{-2.415}$	$2^{-2.415}$	$2^{-7.66}$
5	0	D	C4	0	B	78	$2^{-6.415}$	$2^{-6.415}$	$2^{-14.075}$
6	0	F4	0	1000	50	1088	$2^{-7.415}$	$2^{-7.415}$	$2^{-21.49}$
Total							$2^{-21.49}$		

Table 3. A table describing a 6-round trail, leading to a weak-key class of size 2^{39} in Speck32/64. All RX-differences are in hexadecimal notation.

Round	a_1	b_1	Δ_1	a_2	b_2	Δ_2	Empirical accumulated Prob.
1	0	0	0	0	0	0	$2^{-1.415}$
2	0	0	0	0	0	0	$2^{-2.873}$
3	0	1	0	0	1	2	$2^{-7.243}$
4	0	2	6	0	0	8	$2^{-9.632}$
5	0	D	C4	0	B	78	$2^{-18.016}$
6	0	F4	0	1000	50	1088	$2^{-25.046}$

this trail suggests that a weak-key class of size $2^{-25} \cdot 2^{64} = 2^{39}$ exists, leading to a 7-round distinguisher for Speck32/64.

We used this trail to construct a 7-round distinguisher for Speck32/64. We started by generating a 64-bit random master-key and checked if it belongs to the weak-key class (*i.e.*, if the resulting subkeys satisfy the trail in Table 3). Once an appropriate key was found, we used it to encrypt 2^{32} chosen plaintexts with a $((0,0),1)$-RX difference. Using Theorem 1, we found a possible trail taking into account the RX-difference propagation through the modular addition, and the RX-difference coming through the key injection.

We repeated this experiment using 2^7 keys. The average number of keys we had to discard before finding a "good" key was $35538653 = 2^{25.1}$, suggesting that the weak-key class is indeed of size 2^{39}. In Table 4 we present the trail, the predicted probability, and the average empirical probability. The average number of input pairs with a ((0,0),1)-difference following the full 7-round trail is 1.33, suggesting a probability of $2^{-31.58}$, whereas this probability should be $(2^{-32})^7 = 2^{-224}$ for a random permutation. Moreover, when taking the differential effect into account (*i.e.*, only checking how many pairs satisfy the required RX-difference in the last round), we see that the average number of such pairs is 3.83, suggesting a probability of $2^{-30.06}$.

Table 4. A table describing the RX-distinguisher for 7-round Speck32/64. All RX-differences are in hexadecimal notation, and γ (*i.e.*, the rotation amount) is 1.

Round	Input diff. (left,right)	Key diff.	Output diff. (left,right)	Predicted accumulated Prob.	Empirical accumulated Prob.
0	0, 0	0	0, 0	$2^{-1.415}$	$2^{-1.415}$
1	0, 0	0	0, 0	$2^{-2.83}$	$2^{-2.85}$
2	0, 0	3	3, 3	$2^{-4.245}$	$2^{-4.27}$
3	3, 3	4	607, 60B	$2^{-8.66}$	$2^{-8.68}$
4	607, 60B	11	40E, 1C22	$2^{-15.075}$	$2^{-15.01}$
5	40E, 1C22	1B8	3992, 491A	$2^{-21.49}$	$2^{-21.44}$
6	3992, 491A	1668	333F, 1756	$2^{-31.905}$	$2^{-31.6}$

4.2 Discussion

When designing a new primitive using the ARX structure, constants can come in two forms: known round constants or unknown constants resulting from the key injection.

When open-key attack models are considered (*i.e.*, related-key, weak-key, and known-key), the resistance against rotational cryptanalysis depends on the quality of the key-schedule and its round constants. For example, Skein's underlying block cipher, Threefish, uses constants of low Hamming weight in its key schedule. Along with the fact that the key is only injected once every four rounds, the RX-differences can propagate with relatively high probability or even be canceled under certain circumstance, leading to attacks against round-reduced Threefish. When taking this approach, designers can use Theorem 1 to find how many rounds are required before the number of "good" keys drops to 0.

When open-key attacks are not allowed and uniformly distributed round keys are XORed into the round function, Theorem 1 can be used to derive average case security bounds, similar to other statistical attacks such as differential cryptanalysis and linear cryptanalysis. In this case, the round constants should be chosen to ensure that RX-differences in consecutive rounds cannot be easily canceled by the key injection.

5 Conclusion

As a recently proposed cryptanalytic technique, the impact of rotational cryptanalysis on ARX constructions is not yet well-understood. In this paper, we generalize the notion of a rotational-pair into a pair with RX-difference, and show a rigorous approach to calculate the probability of rotational cryptanalysis when constants are injected into the round function. We test our results and present a 7-round distinguisher with RX-differences for Speck32/64. As a result, we propose countermeasures against rotational cryptanalysis which would be beneficial for future ARX designs. Future research may extend this analysis to cases where the constants are injected through modular addition instead of XOR, and to cases where the rotation amount differs from 1. We note that we did not try to find the longest RX-difference trail for Speck32/64, and it may be possible that automatic search tools can extend it. Another interesting research direction is to apply our formula to the larger versions of Speck.

Thanks

The authors would like to thank Vincent Rijmen, Marc Stevens, and the anonymous reviewers for their useful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was partially supported by the Research Fund KU Leuven, OT/13/071, by the Flemish Government through FWO Thresholds G0842.13 and by European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 HECTOR and grant agreement No H2020-MSCA-ITN-2014-643161 ECRYPT-NET. Yunwen Liu is partially supported by China Scholarship Council (CSC 201403170380) and National Natural Science Foundation (No. 61672530).

Appendix

The automaton representation of computing the probability of Equation 2 and Equation 4 is shown in Figure 2. It starts at state $(1, 1)$, reads $\omega_i = (\zeta_1)_i||(\zeta_2)_i||(\zeta_3)_i$ from $i = n - 1$ to 0 by regarding ω_i as the binary representation of a decimal number and updates the automaton. The probability of Equation 2 (Equation 4 resp.) is nonzero if the automaton ends at state $(1, 0)$ (state $(0, 1)$ resp.), and every time it passes through a full line, the probability is halved.

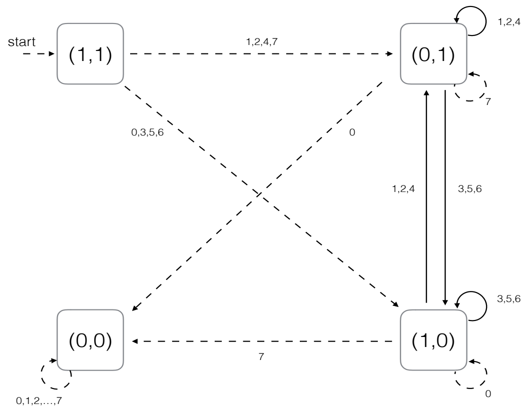

Fig. 2. Automaton representation of the probability of differential equations

References

1. Ashur, T., Liu, Y.: Auxiliary Package for this Paper. `http://homes.esat.kuleuven.be/~tashur/Rotational\_Cryptanalysis\_in\_the\_Presence\_of\_Constants.zip`
2. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE. Submission to NIST (2008)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The Simon and Speck lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference - DAC 2015. pp. 175:1–175:6. ACM (2015)
4. Bernstein, D.J.: ChaCha, a variant of Salsa20. `http://cr.yp.to/chacha.html`
5. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: New stream cipher designs, pp. 84–97. Springer (2008)
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
7. Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Topics in Cryptology - CT-RSA 2014, pp. 227–250. Springer (2014)
8. Biryukov, A., Velichkov, V., Corre, Y.L.: Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 289–310. Springer (2016), `http://dx.doi.org/10.1007/978-3-662-52993-5_15`
9. Biryukov, A., Velichkov, V., Le Corre, Y.: Automatic search for the best trails in ARX: Application to block cipher Speck. Cryptology ePrint Archive, Report 2016/409 (2016), `http://eprint.iacr.org/`
10. Daum, M.: Cryptanalysis of Hash Functions of the MD4-Family. Ph.D. thesis, Ruhr-Universitt Bochum (2005)
11. Dunkelman, O., Indesteege, S., Keller, N.: A differential-linear attack on 12-round Serpent. In: Progress in Cryptology-INDOCRYPT 2008, pp. 308–321. Springer (2008)
12. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family. Submission to NIST (round 3) (2010)
13. Khovratovich, D., Nikolić, I.: Rotational cryptanalysis of ARX. In: Fast Software Encryption. pp. 333–346. Springer (2010)
14. Khovratovich, D., Nikolić, I., Pieprzyk, J., Sokołowski, P., Steinfeld, R.: Rotational cryptanalysis of ARX revisited. In: Fast Software Encryption. pp. 519–536. Springer (2015)
15. Khovratovich, D., Nikolić, I., Rechberger, C.: Rotational rebound attacks on reduced Skein. In: Advances in Cryptology-ASIACRYPT 2010, pp. 1–19. Springer (2010)
16. Knudsen, L.R., Rijmen, V., Rivest, R.L., Robshaw, M.J.: On the design and security of RC2. In: Fast Software Encryption. pp. 206–221. Springer (1998)
17. Leurent, G.: Construction of differential characteristics in ARX designs application to Skein. In: Advances in Cryptology - CRYPTO 2013, pp. 241–258. Springer (2013)
18. Liu, Y., Wang, Q., Rijmen, V.: Automatic search of linear trails in ARX with applications to SPECK and Chaskey. In: International Conference on Applied Cryptography and Network Security. pp. 485–499. Springer (2016)

19. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Advances in Cryptology - EUROCRYPT 1993. pp. 386–397. Springer (1994)
20. Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: an efficient MAC algorithm for 32-bit microcontrollers. In: Selected Areas in Cryptography–SAC 2014, pp. 306–323. Springer (2014)
21. Mouha, N., Velichkov, V., De Canniere, C., Preneel, B.: The differential analysis of S-functions. In: International Workshop on Selected Areas in Cryptography. pp. 36–56. Springer (2010)
22. Needham, R.M., Wheeler, D.J.: TEA extensions. Tech. rep. (1997)
23. Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Fast Software Encryption. pp. 144–162. Springer (2006)
24. Schulte-Geers, E.: On CCZ-equivalence of addition mod 2^n. Designs, Codes and Cryptography 66(1-3), 111–127 (2013)
25. Standaert, F.X., Piret, G., Gershenfeld, N., Quisquater, J.J.: SEA: A scalable encryption algorithm for small embedded applications. In: Smart Card Research and Advanced Applications, pp. 222–236. Springer (2006)
26. Wheeler, D.J., Needham, R.M.: TEA, a tiny encryption algorithm. In: Fast Software Encryption - FSE '95. pp. 363–366. Springer (1995)
27. Yao, Y., Zhang, B., Wu, W.: Automatic Search for Linear Trails of the SPECK Family. In: Lopez, J., Mitchell, C.J. (eds.) Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9290, pp. 158–176. Springer (2015), `http://dx.doi.org/10.1007/978-3-319-23318-5_9`