IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2025, No. 1, pp. 542–591. DOI:10.46586/tosc.v2025.i1.542-591

Exact Formula for RX-Differential Probability Through Modular Addition for All Rotations

Alex Biryukov[1,2], Baptiste Lambin[2] and Aleksei Udovenko[2]

[1] DCS, University of Luxembourg, Esch-sur-Alzette, Luxembourg
[2] SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
firstname.lastname@uni.lu

Abstract. This work presents an exact and compact formula for the probability of rotation-xor differentials (RX-differentials) through modular addition, for *arbitrary* rotation amounts, which has been a long-standing open problem. The formula comes with a rigorous proof and is also verified by extensive experiments.
Our formula uncovers error in a recent work from 2022 proposing a formula for rotation amounts bigger than 1. Surprisingly, it also affects correctness of the more studied and used formula for the rotation amount equal to 1 (from TOSC 2016). Specifically, it uncovers rare cases where the assumptions of this formula do not hold. Correct formula for *arbitrary* rotations now opens up a larger search space where one can often find better trails.
For applications, we propose automated mixed integer linear programming (MILP) modeling techniques for searching optimal RX-trails based on our exact formula. They are consequently applied to several ARX designs, including Salsa, Alzette and a small-key variant of Speck, and yield many new RX-differential distinguishers, some of them based on provably optimal trails. In order to showcase the relevance of the RX-differential analysis, we also design Malzette, a 12-round Alzette-based permutation with maliciously chosen constants, which has a practical RX-differential distinguisher, while standard differential/linear security arguments suggest sufficient security.

Keywords: Differential cryptanalysis · Rotational cryptanalysis · RX-differentials

1 Introduction

Rotational cryptanalysis is a cryptanalysis technique introduced in 2010 by Khovratovich and Nikolic [KN10]. It mainly targeted ARX ciphers (based on addition, rotation, and XOR of n-bit words), where one studies how a pair of plaintexts $(x, x \lll k)$ evolves through a block cipher, where $\lll k$ denotes the cyclic left shift (rotation) by k bits. This technique was later generalized by Ashur and Liu in 2016 [AL16] to consider Rotational-XOR differences (RX-differences for short), where now one considers pairs of plaintexts $(x, (x \lll k) \oplus \Delta)$ for some difference Δ. This led to the publication of an attack on a reduced-round version of the Speck block cipher by Liu et al. [LWRA17] as well as a rectangle-like attack on reduced-round Simon by Chen et al. [CZX+23]. In this paper, we focus only on ARX ciphers and do not consider Simon-like ("AndRX") ciphers. Due to the nature of RX-differentials, the resulting distinguishers/attacks are usually in either the weak-key model (i.e. valid only for a subset of keys) or in the related-key model (i.e. where the attacker can also make encryption queries using a rotationally related key).

The work was supported by the Luxembourg National Research Fund's (FNR) and the German Research Foundation's (DFG) joint project APLICA (C19/IS/13641232).

Received: 2024-06-01 Revised: 2024-11-22 Accepted: 2025-01-23 Published: 2025-03-07

Moreover, to the best of our knowledge, all previous RX-differential cryptanalysis on ARX ciphers was done using rotation $k = 1$. The formula given in [AL16] is an approximation of the probability of RX-differentials through the modular addition for the case $k = 1$, the same for the attack on Speck [LWRA17], and finally the automatic tool CASCADA [Ran22] also handles only this case. One exception would be [KNR10] which does use a different value for k, however, it only focuses on pure rotational cryptanalysis and not RX-cryptanalysis (i.e. it considers only differences $\Delta = 0$).

Finally, the authors of [HXW22] claim to provide an exact formula to compute the probability of RX-differentials through the modular addition for $k > 1$. However, the authors themselves reveal that their formula is not exact, as they provide experimental results (see Figure A1 of [HXW22]) that show discrepancies with their theoretical formula (the authors do not comment on this issue). We also performed experiments with their formula and confirmed these discrepancies. In particular, it leads to claims that some RX-differentials through modular addition have a nonzero probability, whereas we show that these are impossible RX-differentials.

Our Contribution

(Validity criteria and probability formula) As our main contribution, we propose both a simple validity criteria (whether an RX-differential has a zero probability or not) and an exact closed formula for the probability of RX-differentials through the modular addition used in ARX ciphers (addition modulo 2^n for some n), which has been a long standing open problem. The following theorem states our main result:

Theorem (Main)**.** *Let n, k be integers with $1 \leq k < n$ and let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$. Write $\alpha = \alpha_{L'}||\alpha_{R'}, \beta = \beta_{L'}||\beta_{R'}, \Delta = \Delta_{L'}||\Delta_{R'}$, where $\alpha_{L'}, \beta_{L'}, \Delta_{L'} \in \mathbb{F}_2^{n-k}$ and $\alpha_{R'}, \beta_{R'}, \Delta_{R'} \in \mathbb{F}_2^k$. Let us write $\overleftarrow{x}$ for the left rotation of x by k bits. If there exists $x, y \in \mathbb{F}_2^n$ satisfying*

$$[(\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta)] \oplus \overleftarrow{x \boxplus y} = \Delta \tag{1}$$

then, the probability of this equation over all $x, y \in \mathbb{F}_2^n$ can be expressed as

$$T_{n-k}(\alpha_{L'}, \beta_{L'}, \Delta_{L'}, \alpha_0 \oplus \beta_0 \oplus \Delta_0) \times T_k(\alpha_{R'}, \beta_{R'}, \Delta_{R'}, \alpha_k \oplus \beta_k \oplus \Delta_k)$$

$$\text{where } T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, \mathbf{w}) = 2^{-d-1} + \mathbb{1}_{\boldsymbol{\alpha} \oplus \boldsymbol{\beta} \oplus \boldsymbol{\Delta} \in \{0\ldots0,1\ldots1\}} \times \left((-1)^{\mathbf{w}} \times 2^{-m-1}\right)$$

and d is the number of indices i such that $(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\Delta}_i)$ are not all equal (i.e. different from (0,0,0) and (1,1,1)), excluding *the MSB ($i = m - 1$).*

For the special case $k = 1$, our formula uncovers cases where randomness/independence assumptions of [AL16] do not hold. The correction factor ranges from 1/2 to 3/2, per one modular addition transition. Several papers have built on the imprecise results, often claiming optimal trails. Although these cases are rare and might not affect existing trails, optimality claims need to be revisited. We provide a detailed analysis in Section 3.

We also close a related open problem and provide a way to efficiently compute the probability of RX-differentials through *the modular addition with a constant* and the probability of the standard differential through the modular addition with two different constants. The latter can be useful more broadly, for example in differential cryptanalysis of domain separation in symmetric ARX primitives.

(MILP modeling) We show how to model our new formulas with MILP constraints in order to search for high-probability RX-differential trails. We illustrate it on several targets, including Salsa, a small-key variant of Speck, and Alzette [BBdS+20a]. We report many new distinguishers and discuss the results. In particular, we found many cases where the rotation amount $k > 1$ leads to better distinguishers than $k = 1$, contradicting the

previous belief and highlighting the importance of general RX-differential cryptanalysis. However, we emphasize that we do not claim that rotations $k > 1$ are generally preferable over $k = 1$: the extra rotation amounts simply open up a bigger search space which may in principle contain better trails. Our modeling employs several advanced MILP techniques which can be of general interest.

(Malicious permutation Malzette) We also design a proof-of-concept permutation Malzette based on 12-round ARX-box Alzette [BBdS+20a] with maliciously chosen constants, in spirit of previous works about "malicious cryptography" [AAE+14, AY15, Mor15]. This permutation inherits strong bounds against linear/differential cryptanalysis from Alzette, while being susceptible to a practical RX-differential distinguisher. This emphasizes the importance of RX-differential cryptanalysis in the area of symmetric-key designs.

The source code and supporting data of our experiments are publicly available at

https://github.com/cryptolu/RX-Differentials-Probability

Notations

Notations used in the rest of the paper are summarized in Table 1. Modular addition takes precedence over the XOR, i.e., if we write $a \oplus b \boxplus c$, this is equivalent to $a \oplus (b \boxplus c)$. In SHL, the most significant bit is removed.

Table 1: Notations used in this work

Notation	Definition
$x = (x_{n-1}, \ldots, x_0)$	n-bit Boolean vector, x_0 is the least significant bit
$0 \ldots 0, 1 \ldots 1$	the all-zeroes and all-ones bit vectors
LSB	Least significant bit
MSB	Most significant bit
$x \boxplus y$	Addition modulo 2^n
$x \wedge y$	Bitwise AND
$x \vee y$	Bitwise OR
$x \| \| y$	Concatenation of x and y
$\overline{x}$	Bitwise complement of x
$\mathrm{wt}(x)$	Hamming weight of x
$\mathrm{SHL}(x)$	Non-cyclic left shift of x by one bit
$x \lll k$	Cyclic shift to the left of x by k bits
$x \ggg k$	Cyclic shift to the right of x by k bits
$\overleftarrow{x}$	Equivalent to $x \lll k$ where k is implicitly defined
$\overrightarrow{x}$	Equivalent to $x \ggg k$, the inverse of $\overleftarrow{x}$
$x \preccurlyeq y$	$x_i \leq y_i$ for all i
$\mathbb{1}_E$	Evaluates to 1 if the expression E is true, 0 otherwise
I	The identity mapping
$(I \oplus \mathrm{SHL})(x)$	$x \oplus \mathrm{SHL}(x)$
$\Pr_{x,y}[E]$	Probability of E over variables x, y
$\#X$	Cardinality of the set X

2 Exact Probability of RX-Differentials

The goal of this section is to give a closed formula to evaluate the probability of the equation

$$(\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta) \;\oplus\; \overleftarrow{x \boxplus y} = \Delta \tag{2}$$

where $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ are fixed differences, and $x, y \in \mathbb{F}_2^n$ are independent uniform random variables. We will also use a shorthand

$$\chi = \alpha \oplus \beta \oplus \Delta$$

2.1 Lemmas from previous results

We start with two previous results about standard differentials over $\boxplus$.

Lemma 1 ([LM01, Sch13]). *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be constants and $x, y \in \mathbb{F}_2^n$ independent uniformly random variables. Then,*

$$\Pr_{x,y}\left[\ (x \oplus \alpha) \boxplus (y \oplus \beta) \ \oplus\ x \boxplus y = \Delta\ \right] = \mathbb{1}_{(I \oplus \mathrm{SHL})(\chi) \preccurlyeq \mathrm{SHL}((\alpha \oplus \Delta) \vee (\beta \oplus \Delta))} \cdot 2^{-\operatorname{wt}(\mathrm{SHL}((\alpha \oplus \Delta) \vee (\beta \oplus \Delta)))}$$

Equivalently, for a valid difference tuple (α, β, Δ), the probability is given by 2^{-d} where d is the amount of indices i such that $(\alpha_i, \beta_i, \Delta_i)$ are not all equal (i.e. different from 000 and 111) *excluding* the MSB (i.e. $i < n-1$). Furthermore, for (α, β, Δ) to be valid, we need at least to have $\chi_0 = \alpha_0 \oplus \beta_0 \oplus \Delta_0 = 0$.

Lemma 2 ([AL16]). *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be constants and $x, y \in \mathbb{F}_2^n$ independent uniformly random variables. Then,*

$$\begin{aligned}&\Pr_{x,y}\left[\ (x \oplus \alpha) \boxplus (y \oplus \beta) \ \oplus\ x \boxplus y \boxplus 1 = \Delta\ \right]\\ =&\Pr_{x,y}\left[\ (x \oplus \alpha) \boxplus (y \oplus \beta) \boxplus 1 \ \oplus\ x \boxplus y = \Delta\ \right]\\ =&\mathbb{1}_{(I \oplus \mathrm{SHL})(\chi) \oplus 1 \preccurlyeq \mathrm{SHL}((\alpha \oplus \Delta) \vee (\beta \oplus \Delta))} \cdot 2^{-\operatorname{wt}(\mathrm{SHL}((\alpha \oplus \Delta) \vee (\beta \oplus \Delta)))}\end{aligned}$$

The difference from the previous lemma is the validity criteria, which in particular requires $\chi_0 = \alpha_0 \oplus \beta_0 \oplus \Delta_0 = 1$ instead, and the conditions for the other positions are the same.

2.2 Decomposition of the problem

For words $x, y \in \mathbb{F}_2^n$, we denote $x = x_L || x_R$ and $y = y_L || y_R$ with $x_L, y_L \in \mathbb{F}_2^k$ and $x_R, y_R \in \mathbb{F}_2^{n-k}$. Similarly, for differences $\alpha, \beta, \Delta \in \mathbb{F}_2^n$, we write $\alpha = \alpha_{L'} || \alpha_{R'}$ (and respectively for β and Δ) where $\alpha_{L'} \in \mathbb{F}_2^{n-k}$ and $\alpha_{R'} \in \mathbb{F}_2^k$.

Moreover, we denote by $c_R \in \mathbb{F}_2$ the outgoing carry of $x_R \boxplus y_R$, i.e. $c_R = 0$ if $x_R + y_R < 2^{n-k}$ and $c_R = 1$ otherwise. Similarly, we denote by $c_L \in \mathbb{F}_2$ the outgoing carry of $(x_L \oplus \alpha_{R'}) \boxplus (y_L \oplus \beta_{R'})$. Using these notations, we can write

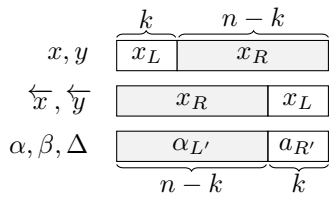

Figure 1: Splitting of vectors. The most significant bits are on the left, the least significant bits are on the right.

$$\begin{aligned}x \boxplus y &= x_L \boxplus y_L \boxplus c_R \ ||\ x_R \boxplus y_R\\ (\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta) &= (x_R \oplus \alpha_{L'}) \boxplus (y_R \oplus \beta_{L'}) \boxplus c_L \ ||\ (x_L \oplus \alpha_{R'}) \boxplus (y_L \oplus \beta_{R'})\end{aligned}$$

This means that the equation

$$(\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta) \ \oplus\ \overleftarrow{x \boxplus y} = \Delta \tag{3}$$

can be rewritten as a system of equations

$$\begin{cases} (x_R \oplus \alpha_{L'}) \boxplus (y_R \oplus \beta_{L'}) \boxplus c_L \ \oplus \ x_R \boxplus y_R = \Delta_{L'} & \text{(4c)} \\ (x_L \oplus \alpha_{R'}) \boxplus (y_L \oplus \beta_{R'}) \ \oplus \ x_L \boxplus y_L \boxplus c_R = \Delta_{R'} & \text{(4d)} \\ c_R = \mathbb{1}_{x_R + y_R \geq 2^{n-k}} & \text{(4e)} \\ c_L = \mathbb{1}_{(x_L \oplus \alpha_{R'}) + (y_L \oplus \beta_{R'}) \geq 2^k} & \text{(4f)} \end{cases}$$

By taking Equations (4c), (4d) modulo 2 (or by Lemma 1 and Lemma 2), we can deduce the only possible values of the carries c_L, c_R from the differences:

$$c_L = (\alpha_{L'} \oplus \beta_{L'} \oplus \Delta_{L'})_0 = \alpha_k \oplus \beta_k \oplus \Delta_k = \chi_k$$
$$c_R = (\alpha_{R'} \oplus \beta_{R'} \oplus \Delta_{R'})_0 = \alpha_0 \oplus \beta_0 \oplus \Delta_0 = \chi_0$$

Furthermore, we can rewrite Equation (4d) to have the same structure as Equation (4c) using the change of variables $\tilde{x_L} = x_L \oplus \alpha_{R'}$ and $\tilde{y_L} = y_L \oplus \beta_{R'}$. This leads to an equivalent system

$$\begin{cases} (x_R \oplus \alpha_{L'}) \boxplus (y_R \oplus \beta_{L'}) \boxplus \chi_k \ \oplus \ x_R \boxplus y_R = \Delta_{L'} \\ \chi_0 = \mathbb{1}_{x_R + y_R \geq 2^{n-k}} \\ (\tilde{x_L} \oplus \alpha_{R'}) \boxplus (\tilde{y_L} \oplus \beta_{R'}) \boxplus \chi_0 \ \oplus \ \tilde{x_L} \boxplus \tilde{y_L} = \Delta_{R'} \\ \chi_k = \mathbb{1}_{\tilde{x_L} + \tilde{y_L} \geq 2^k} \end{cases}$$

Now, we use the fact that α, β, γ and thus χ are all fixed. Therefore, the system splits into two *independent* parts of the following form (one on x_R, y_R and one on $\tilde{x_L}, \tilde{y_L}$), with vectors over $\mathbb{F}_2^m$, m being k or $n-k$:

$$\begin{cases} (x \oplus \boldsymbol{\alpha}) \boxplus (y \oplus \boldsymbol{\beta}) \boxplus (\boldsymbol{\alpha}_0 \oplus \boldsymbol{\beta}_0 \oplus \boldsymbol{\Delta}_0) \oplus x \boxplus y = \boldsymbol{\Delta} \\ \mathbb{1}_{x+y \geq 2^m} = \mathbf{w} \end{cases} \tag{5}$$

with two instantiations:

1. $m = n-k$, $\boldsymbol{\alpha} = \alpha_{L'}, \boldsymbol{\beta} = \beta_{L'}, \boldsymbol{\Delta} = \Delta_{L'}, \mathbf{w} = \chi_0$.
2. $m = k$, $\boldsymbol{\alpha} = \alpha_{R'}, \boldsymbol{\beta} = \beta_{R'}, \boldsymbol{\Delta} = \Delta_{R'}, \mathbf{w} = \chi_k$.

Denote by $T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, \mathbf{w})$ the number of solutions to Equation (5). Note that Equation (5) is essentially a generalization of Lemma 1 and Lemma 2 to a restriction of the output carries of the counted solutions to the differential equation. We conclude with the following proposition and we will focus on expressing T_m in the following sections.

Proposition 1. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ define an RX-differential with the notations from this section. Then we have*

$$\Pr_{x,y}[(\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta) \ \oplus \ \overleftarrow{x \boxplus y} = \Delta] =$$
$$T_{n-k}(\alpha_{L'}, \beta_{L'}, \Delta_{L'}, \chi_0) \ \times \ T_k(\alpha_{R'}, \beta_{R'}, \Delta_{R'}, \chi_k)$$

2.3 Validity criteria for RX-differentials

As a consequence of Proposition 1, we deduce the following theorem (criteria) for an RX-differential to have nonzero probability.

Theorem 1. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be fixed constants and $u, v \in \mathbb{F}_2^n$ be defined as*

$$u = (I \oplus \mathrm{SHL})(\alpha \oplus \beta \oplus \Delta)$$
$$v = \mathrm{SHL}((\alpha \oplus \Delta) \vee (\beta \oplus \Delta))$$

Then, the equation

$$(\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta) \ \oplus \ \overleftarrow{x \boxplus y} = \Delta$$

has at least one solution (x, y) *if and only if :*

- $u \preccurlyeq v$ *when* $k = 0$
- $u_i \leq v_i$ *for all* $i \neq 0, k$ *when* $k \neq 0$

which means that the RX-propagation $[(\alpha, k), (\beta, k)] \xrightarrow{\boxplus} (\Delta, k)$ *is valid (has nonzero probability) if and only if the above condition is satisfied. Overall, this is the same criterion as in Lemma 1 but ignoring the bits of index 0 and* k *when* $k \neq 0$*.*

Proof. When $k = 0$, this is directly Lemma 1, so we consider only $k > 0$.

If the probability is nonzero (i.e. there is at least one solution), then there is at least one solution to Equations (4c), (4d). Thus, in particular, from Lemmas 1 and 2, we have the expected criteria. Note that the full-word SHL operation differs from two separate k-bit and $(n-k)$-bit SHL operations by not removing the overflowing bit of the k-bit output. However, this bit does not affect the condition as it is excluded from consideration by the criteria $i \neq k$.

For the other direction, assume that the above criteria hold. From Proposition 1, then whatever the value of the XOR of the LSB of $(\alpha_{L'}, \beta_{L'}, \Delta_{L'})$ (resp. of $(\alpha_{R'}, \beta_{R'}, \Delta_{R'})$), we have a solution to Equation (4c) (resp. 4d). However, we still need to show that there is always a solution that satisfies the corresponding condition on the output carry of $x + y$.

Let (x_R, y_R) and (x_L, y_L) be any such solution to 4c and 4d, respectively, and let c_R, c_L be their respective outgoing carry. Let $x'_R = x_R \oplus 10\ldots0$, i.e. x_R but with the MSB flipped, and y'_R, x'_L, y'_L dbe efined analogously. Consider the four following tuples (x_R, y_R), (x'_R, y_R), (x_R, y'_R), (x'_R, y'_R). Then all four of these are also solutions to 4c. Indeed, in the modular addition, the MSB behaves linearly for differentials, because $x \oplus 10\ldots0 = x \boxplus 10\ldots0$ and so this bit flip propagates through modular additions with probability 1 (including the possible c_L/c_R carry addition). Since flipping the MSB is the same as adding a 1-bit difference, all four of these are also solutions to 4c. Moreover, at least one of the four tuples will be such that both MSBs are (0,0) (necessarily leading to the outgoing carry being 0), and similarly, at least one of them will be such that both MSBs are (1,1) (necessarily leading to the outgoing carry being 1). The same can be applied to (x_L, y_L) with x'_L and y'_L.

Thus, we always have at least one solution that leads to the outgoing carry being 0, and one solution leading to the outgoing carry being 1, which means that there will always be at least one solution satisfying the corresponding condition on $\mathbb{1}_{x+y \geq 2^m}$. Hence we can conclude that if the criterion holds, then we always have at least one solution, i.e. a nonzero probability.

□

2.4 Solving the subcase system

Returning back to computing the probability of RX-transition, we need to evaluate the probability of a system of the form

$$\begin{cases} (x \oplus \alpha) \boxplus (y \oplus \beta) \boxplus (\alpha_0 \oplus \beta_0 \oplus \Delta_0) \ \oplus \ x \boxplus y = \Delta \\ x + y \ \texttt{op} \ 2^n \end{cases} \tag{6}$$

for both cases of the comparison `op` (either $<$ or $\geq$).

Note that instead of directly computing the probability, we will instead count the number of solutions, as it is slightly simpler (and we can easily derive the probability from

the solution count). Moreover, we can focus on the case where op is $<$, since the two cases sum up to the probability of the basic differential transition (given by Lemma 1 and Lemma 2).

For $\alpha, \beta, \Delta \in \mathbb{F}_2^n$, we will denote by $\mathrm{XDS}_n(\alpha, \beta, \Delta)$ and $\mathrm{XDS}_n^+(\alpha, \beta, \Delta)$ the sets

$$\mathrm{XDS}_n(\alpha, \beta, \Delta) = \{(x, y) \in \mathbb{F}_2^{2n} \text{ s.t. } (x \oplus \alpha) \boxplus (y \oplus \beta) \oplus x \boxplus y = \Delta\}$$
$$\mathrm{XDS}_n^+(\alpha, \beta, \Delta) = \{(x, y) \in \mathbb{F}_2^{2n} \text{ s.t. } (x \oplus \alpha) \boxplus (y \oplus \beta) \boxplus 1 \oplus x \boxplus y = \Delta\}$$

When α, β, Δ are clear from the context, we will shorten it to simply XDS_n and XDS_n^+. We also denote $\mathrm{xdp}_n(\alpha, \beta, \Delta) = \#\mathrm{XDS}_n(\alpha, \beta, \Delta)/2^{2n}$ (analogously $\mathrm{xdp}_n^+ = \#\mathrm{XDS}_n^+/2^{2n}$), similarly shortened to xdp_n (resp. xdp_n^+) when appropriate.

First, it turns out that we can easily deduce the size of XDS_n^+ if we can compute the size of XDS_n, as stated in the following proposition.

Proposition 2. *With the above notations, and $\bar{\alpha} = \alpha \oplus 1 \ldots 1$ (and respectively for $\bar{\beta}$ and $\bar{\Delta}$) we have*

$$\#\mathrm{XDS}_n^+(\alpha, \beta, \Delta) = \#\mathrm{XDS}_n(\bar{\alpha}, \bar{\beta}, \bar{\Delta})$$

Proof. Using the identity $X \boxplus Y \boxplus 1 = \overline{\bar{X} \boxplus \bar{Y}}$, we can write

$$\begin{aligned}
\mathrm{XDS}_n^+(\alpha, \beta, \Delta) &= \{(x, y) \in \mathbb{F}_2^{2n} \text{ s.t. } (x \oplus \alpha) \boxplus (y \oplus \beta) \boxplus 1 \oplus x \boxplus y = \Delta\} \\
&= \{(x, y) \in \mathbb{F}_2^{2n} \text{ s.t. } \overline{\overline{(x \oplus \alpha)} \boxplus \overline{(y \oplus \beta)}} \oplus x \boxplus y = \Delta\} \\
&= \{(x, y) \in \mathbb{F}_2^{2n} \text{ s.t. } \overline{(x \oplus \bar{\alpha}) \boxplus (y \oplus \bar{\beta})} \oplus x \boxplus y = \Delta\} \\
&= \{(x, y) \in \mathbb{F}_2^{2n} \text{ s.t. } (x \oplus \bar{\alpha}) \boxplus (y \oplus \bar{\beta}) \oplus x \boxplus y = \bar{\Delta}\} \\
&= \mathrm{XDS}_n(\bar{\alpha}, \bar{\beta}, \bar{\Delta})
\end{aligned}$$

□

As such, we can focus on only evaluating the number of solutions to the system

$$\begin{cases} (x \oplus \alpha) \boxplus (y \oplus \beta) \oplus x \boxplus y = \Delta \\ x + y < 2^n \end{cases}$$

for some given $\alpha, \beta, \Delta \in \mathbb{F}_2^n$, and deduce the other cases from that.

First, we begin with a core lemma that we will use repeatedly.

Lemma 3. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be fixed valid differences, and let $\alpha', \beta', \Delta' \in \mathbb{F}_2$ be so that the differences $\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta$ are also valid. Then,*

$$\#\mathrm{XDS}_{n+1}(\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta) = \begin{cases} 4 \times \#\mathrm{XDS}_n(\alpha, \beta, \Delta) & \alpha_{n-1} = \beta_{n-1} = \Delta_{n-1} \\ 2 \times \#\mathrm{XDS}_n(\alpha, \beta, \Delta) & \textit{otherwise} \end{cases}$$

Proof. From Lemma 1, we know that $\mathrm{xdp}_n(\alpha, \beta, \Delta) = 2^{-d}$ where d is the amount of indices i such that $(\alpha_i, \beta_i, \Delta_i)$ are not all equal, excluding the MSB (i.e., $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1})$). Thus when considering $\mathrm{XDS}_{n+1}(\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta)$, we do not need to consider whether or not $(\alpha', \beta', \Delta')$ are all equal, however, we do need to check $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1})$.

If $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1})$ are not all equal, we have $\mathrm{xdp}_{n+1}(\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta) = 2^{-(d+1)} = \frac{1}{2}\mathrm{xdp}_n(\alpha, \beta, \Delta)$. Thus we get

$$\begin{aligned}
\mathrm{xdp}_{n+1} = \frac{1}{2}\mathrm{xdp}_n \Leftrightarrow & \frac{\#\mathrm{XDS}_{n+1}}{2^{2(n+1)}} = \frac{1}{2} \times \frac{\#\mathrm{XDS}_n}{2^{2n}} \\
\Leftrightarrow & \#\mathrm{XDS}_{n+1} = 2 \times \#\mathrm{XDS}_n
\end{aligned}$$

Otherwise if $\alpha_{n-1} = \beta_{n-1} = \Delta_{n-1}$, we have $\mathrm{xdp}_{n+1}(\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta) = 2^{-d} = \mathrm{xdp}_n(\alpha, \beta, \Delta)$. Thus we get

$$\begin{aligned}\mathrm{xdp}_{n+1} = \mathrm{xdp}_n \Leftrightarrow & \frac{\#\mathrm{XDS}_{n+1}}{2^{2(n+1)}} = \frac{\#\mathrm{XDS}_n}{2^{2n}} \\ \Leftrightarrow & \#\mathrm{XDS}_{n+1} = 4 \times \#\mathrm{XDS}_n\end{aligned}$$

□

The following three lemmas are also needed to prove the main result of this subsection.

Lemma 4. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be fixed valid differences, and let $\alpha', \beta', \Delta' \in \mathbb{F}_2$ be so that the differences $\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta$ are also valid. Let $x, y \in \mathbb{F}_2^n$ and $x', y' \in \mathbb{F}_2$ be any solution to the corresponding differential equation on the $n+1$ bits, that is,*

$$[((x'||x) \oplus (\alpha'||\alpha)) \boxplus ((y'||y) \oplus (\beta'||\beta))] \oplus [(x'||x) \boxplus (y'||y)] = (\Delta'||\Delta)$$

Furthermore, let $c \in \mathbb{F}_2$ be the outgoing carry of $x + y$, that is, $c = 0$ when $x + y < 2^n$ and $c = 1$ otherwise. Similarly, let $\bar{c} \in \mathbb{F}_2$ be the outgoing carry of $(x \oplus \alpha) + (y \oplus \beta)$. Then $\alpha' \oplus \beta' \oplus \Delta' = c \oplus \bar{c}$.

Proof. We can split the differential equation between the MSB and the last n bits as

$$\begin{aligned}&[((x'||x) \oplus (\alpha'||\alpha)) \boxplus ((y'||y) \oplus (\beta'||\beta))] \oplus [(x'||x) \boxplus (y'||y)] = (\Delta'||\Delta) \\ \Leftrightarrow &\begin{cases} [(x \oplus \alpha) \boxplus (y \oplus \beta)] \oplus [x \boxplus y] = \Delta \\ [(x' \oplus \alpha') \boxplus (y' \oplus \beta') \boxplus \bar{c}] \oplus [x' \boxplus y' \boxplus c] = \Delta' \end{cases}\end{aligned}$$

Note that the second equation is over one single bit, and as such, the modular addition is equivalent to the XOR operation. Thus we get

$$\begin{aligned}& [(x' \oplus \alpha') \boxplus (y' \oplus \beta') \boxplus \bar{c}] \oplus [x' \boxplus y' \boxplus c] = \Delta' \\ \Leftrightarrow\ & [(x' \oplus \alpha') \oplus (y' \oplus \beta') \oplus \bar{c}] \oplus [x' \oplus y' \oplus c] = \Delta' \\ \Leftrightarrow\ & \alpha' \oplus \beta' \oplus \Delta' = c \oplus \bar{c}\end{aligned}$$

□

Lemma 5. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be fixed valid differences, and let $\alpha', \beta', \Delta' \in \mathbb{F}_2$ be so that the differences $\alpha'||\alpha, \beta'||\beta, \Delta'||\Delta$ are also valid. Then we cannot have*

$$(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = (000) \ \wedge \ \alpha' \oplus \beta' \oplus \Delta' = 1$$

nor

$$(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = (111) \ \wedge \ \alpha' \oplus \beta' \oplus \Delta' = 0$$

Proof. Denote $\tilde{\alpha} = \alpha'||\alpha$, $\tilde{\beta} = \beta'||\beta$ and $\tilde{\Delta} = \Delta'||\Delta$. Then by Lemma 1, since $\tilde{\alpha}, \tilde{\beta}, \tilde{\Delta}$ is a valid differential, it must verify

$$(\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta}) \oplus \mathrm{SHL}(\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta}) \preccurlyeq \mathrm{SHL}((\tilde{\alpha} \oplus \tilde{\Delta}) \vee (\tilde{\beta} \oplus \tilde{\Delta}))$$

In particular, for the MSB, we must have

$$(\alpha' \oplus \beta' \oplus \Delta') \oplus (\alpha_{n-1} \oplus \beta_{n-1} \oplus \Delta_{n-1}) \leq (\alpha_{n-1} \oplus \Delta_{n-1}) \vee (\beta_{n-1} \oplus \Delta_{n-1})$$

If $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = (000)$, we have $(\alpha_{n-1} \oplus \Delta_{n-1}) \vee (\beta_{n-1} \oplus \Delta_{n-1}) = 0$ and $(\alpha_{n-1} \oplus \beta_{n-1} \oplus \Delta_{n-1}) = 0$, thus we need to have $\alpha' \oplus \beta' \oplus \Delta' \preccurlyeq 0$ i.e. $\alpha' \oplus \beta' \oplus \Delta' = 0$.

Similarly, if $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = (111)$, we have $(\alpha_{n-1} \oplus \Delta_{n-1}) \vee (\beta_{n-1} \oplus \Delta_{n-1}) = 0$ and $(\alpha_{n-1} \oplus \beta_{n-1} \oplus \Delta_{n-1}) = 1$, thus we need to have $\alpha' \oplus \beta' \oplus \Delta' \oplus 1 \preccurlyeq 0$ i.e. $\alpha' \oplus \beta' \oplus \Delta' = 1$ □

Lemma 6. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be fixed valid differences. Moreover, let $c \in \mathbb{F}_2$ be the outgoing carry of $x + y$, i.e. $c = 0$ when $x + y < 2^n$ and $c = 1$ otherwise. Similarly, let $\bar{c} \in \mathbb{F}_2$ be the outgoing carry of $(x \oplus \alpha) + (y \oplus \beta)$. Let $\mathbb{A}, \mathbb{B}, \mathbb{B}_{01}, \mathbb{B}_{10}$ be the sets defined as*

$$\mathbb{A} = \{(x, y) \in \mathrm{XDS}_n \text{ s.t. } c = \bar{c}\} \qquad \mathbb{B}_{01} = \{(x, y) \in \mathrm{XDS}_n \text{ s.t. } c = 0 \wedge \bar{c} = 1\}$$
$$\mathbb{B} = \{(x, y) \in \mathrm{XDS}_n \text{ s.t. } c \neq \bar{c}\} \qquad \mathbb{B}_{10} = \{(x, y) \in \mathrm{XDS}_n \text{ s.t. } c = 1 \wedge \bar{c} = 0\}$$

where $\mathbb{B} = \mathbb{B}_{01} \cup \mathbb{B}_{10}$. Then:

- *if $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1})$ are not all equal, we have $\#\mathbb{A} = \#\mathbb{B} = \frac{1}{2}\#\mathrm{XDS}_n$ and $\#\mathbb{B}_{01} = \#\mathbb{B}_{10} = \frac{1}{4}\#\mathrm{XDS}_n$*
- *if $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = 000$, we have $\#\mathbb{A} = \#\mathrm{XDS}_n$ and $\#\mathbb{B} = 0$.*
- *if $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = 111$, we have $\#\mathbb{A} = 0$, $\#\mathbb{B} = \#\mathrm{XDS}_n$, and $\#\mathbb{B}_{01} = \#\mathbb{B}_{10} = \frac{1}{2}\#\mathrm{XDS}_n$.*

Proof. Let $\mathbb{X}_0$ and $\mathbb{X}_1$ be the two sets

$$\mathbb{X}_0 = \mathrm{XDS}_{n+1}(0||\alpha, 0||\beta, 0||\Delta) \quad \mathbb{X}_1 = \mathrm{XDS}_{n+1}(0||\alpha, 0||\beta, 1||\Delta)$$

For any $((x'||x), (y'||y)) \in \mathbb{X}_0$, by Lemma 4 we have $c \oplus \bar{c} = 0 \oplus 0 \oplus 0 = 0$, and thus $\mathbb{X}_0 \subseteq \mathbb{F}_2^2 \times \mathbb{A}$. Similarly, we also have $\mathbb{X}_1 \subseteq \mathbb{F}_2^2 \times \mathbb{B}$.

On the other hand, let $(x, y) \in \mathbb{A}$. Then from Lemma 1 we can easily see that any $(x', y') \in \mathbb{F}_2^2$ verifies the equation

$$[((x'||x) \oplus (0||\alpha)) \boxplus ((y'||y) \oplus (0||\beta))] \oplus [(x'||x) \boxplus (y'||y)] = (\Delta'||\Delta)$$

where here $\Delta' = c \oplus \bar{c} = 0$ by Lemma 4. Thus we have $((x'||x), (y'||y)) \in \mathrm{XDS}_{n+1}(0||\alpha, 0||\beta, 0||\Delta) = \mathbb{X}_0$, and as such, $\mathbb{F}_2^2 \times \mathbb{A} \subseteq \mathbb{X}_0$ Again, similarly we also have $\mathbb{F}_2^2 \times \mathbb{B} \subseteq \mathbb{X}_1$. Thus, we can conclude that $\mathbb{F}_2^2 \times \mathbb{A} = \mathbb{X}_0$ and $\mathbb{F}_2^2 \times \mathbb{B} = \mathbb{X}_1$.

Then by using Lemma 3:

- If $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) \notin \{000, 111\}$, we have $\#\mathbb{X}_0 = 2 \times \#\mathrm{XDS}_n(\alpha, \beta, \Delta)$. Thus, since $\#\mathbb{X}_0 = \#(\mathbb{F}_2^2 \times \mathbb{A}) = 4 \times \#\mathbb{A}$, we have $\#\mathbb{A} = \frac{1}{2}\mathrm{XDS}_n(\alpha, \beta, \Delta)$. Similarly, we also have $\#\mathbb{B} = \frac{1}{2}\mathrm{XDS}_n(\alpha, \beta, \Delta)$. Moreover, by symmetry, for any $(x, y) \in \mathbb{B}_{01}$, we have $(x', y') = (x \oplus \alpha, y \oplus \beta) \in \mathbb{B}_{10}$, thus $\#\mathbb{B}_{01} = \#\mathbb{B}_{10} = \frac{1}{2}\#\mathbb{B} = \frac{1}{4}\#\mathrm{XDS}_n$.
- If $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = 000$, then by Lemma 5 we have $\#\mathbb{B} = 0$. Since $\mathbb{A}$ and $\mathbb{B}$ form a partition of XDS_n, we thus have $\#\mathbb{A} = \#\mathrm{XDS}_n$.
- Similarly, if $(\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1}) = 111$ then we have $\#\mathbb{A} = 0$ and thus $\#\mathbb{B} = \#\mathrm{XDS}_n$. Again by symmetry, $\mathbb{B}_{01}$ and $\mathbb{B}_{10}$ are of the same size, thus $\#\mathbb{B}_{01} = \#\mathbb{B}_{10} = \frac{1}{2}\#\mathrm{XDS}_n$

□

Now we can finally prove the number of solutions to Equation (6) when $c = 0$. Denote

$$R_n(\alpha, \beta, \Delta) = \#\{(x, y) \in \mathrm{XDS}_n(\alpha, \beta, \Delta) \text{ s.t. } x + y < 2^n\} \tag{7}$$

then we have the following proposition

Proposition 3. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$ be fixed valid differences, and let $\alpha', \beta', \Delta' \in \mathbb{F}_2$ be so that the differences $\tilde{\alpha} = \alpha'||\alpha$, $\tilde{\beta} = \beta'||\beta$ and $\tilde{\Delta} = \Delta'||\Delta$ are also valid. Also denote $\delta_{n-1} = (\alpha_{n-1}, \beta_{n-1}, \Delta_{n-1})$ and $\chi' = \alpha' \oplus \beta' \oplus \Delta'$. Then $R_{n+1}(\tilde{\alpha}, \tilde{\beta}, \tilde{\Delta})$ verifies the recurrence relation*

$$R_{n+1}(\tilde{\alpha}, \tilde{\beta}, \tilde{\Delta}) = \begin{cases} 2R_n(\alpha, \beta, \Delta) \text{ if } \delta_{n-1} \notin \{000, 111\} \text{ and } \chi = 0 \\ \#\mathrm{XDS}_n(\alpha, \beta, \Delta) \text{ if } \delta_{n-1} \notin \{000, 111\} \text{ and } \chi = 1 \\ \#\mathrm{XDS}_n(\alpha, \beta, \Delta) + 2R_n(\alpha, \beta, \Delta) \text{ if } \delta_{n-1} = 000 \text{ and } \chi = 0 \\ 2 \times \#\mathrm{XDS}_n(\alpha, \beta, \Delta) \text{ if } \delta_{n-1} = 111 \text{ and } \chi = 1 \end{cases}$$

Moreover, we have

$$R_n(\alpha,\beta,\Delta) = \begin{cases} \frac{\#\mathrm{XDS}_n(\alpha,\beta,\Delta)}{2} & \text{if } \alpha\oplus\beta\oplus\Delta \neq 0 \\ \frac{\#\mathrm{XDS}_n(\alpha,\beta,\Delta)}{2} + 2^{n-1} & \text{if } \alpha\oplus\beta\oplus\Delta = 0 \end{cases}$$

Proof. First, as shown in Lemma 5, we cannot have $(\delta_{n-1} = 000 \wedge \chi = 1)$ nor $(\delta_{n-1} = 111 \wedge \chi = 0)$. Moreover, we also cannot have $\alpha\oplus\beta\oplus\Delta = 1\ldots1$ from Lemma 1 as it would be an invalid differential. We can show all of the proposition in a single induction proof.

The base case is easy to verify computationally, so we only focus on the induction step. In the rest of this proof, for any given $(x,y) \in \mathbb{F}_2^{2n}$, we denote $c \in \mathbb{F}_2$ the corresponding outgoing carry of $x+y$, i.e. $c=0$ when $x+y < 2^n$ and $c=1$ otherwise and $\bar{c} \in \mathbb{F}_2$ the outgoing carry of $(x\oplus\alpha)+(y\oplus\beta)$. Similarly, for any given $(x'||x, y'||y) \in \mathbb{F}_2^{n+1}$, we denote $c' \in \mathbb{F}_2$ the corresponding outgoing carry of $x'||x+y'||y$, i.e. $c'=0$ when $x'||x+y'||y < 2^{n+1}$ and $c'=1$ otherwise. Our goal is to count how many $(x'||x, y'||y) \in \mathbb{F}_2^{n+1}$ are there such that $c'=0$ *and* $(x'||x, y'||y) \in \mathrm{XDS}_{n+1}$.

In general, we can observe the following (recall that c' is the outgoing carry of 1-bit addition $x'+y'+c$):

- If $(x',y') = (1,1)$, c' is always 1 so no solution of that form exist.
- If $(x',y') = (0,0)$, c' is always 0, so depending on the value of χ, we can use Lemma 6 to evaluate how many solutions we have of this form for (x,y).
- If $(x',y') = (0,1)$, we have $c'=c$, so we will use Lemma 6 according to the value of χ along with the additional constraint that $c=0$.
- If $(x',y') = (1,0)$, by symmetry we have the same number of solutions as the previous case, so we just multiply the previous solution count by 2.

Thus, depending on the value of δ_{n-1} and χ, we have the following.

If $\delta_{n-1} \notin \{000, 111\}$ and $\chi = 0$ By Lemma 4, since $\chi = \alpha'\oplus\beta'\oplus\Delta' = 0$, we have $c=\bar{c}$. When $(x',y') = (0,0)$, we can take any $(x,y) \in \{(x,y) \in \mathrm{XDS}_n \wedge c=\bar{c}\}$, thus $\frac{\#\mathrm{XDS}_n}{2}$ solutions. When $(x',y') = (0,1)$, we can take any $(x,y) \in \{(x,y) \in \mathrm{XDS}_n \wedge c=\bar{c}=0\}$, which by Lemma 6 is of size

$$\begin{aligned} &\#\{(x,y) \in \mathrm{XDS}_n \wedge c=\bar{c}=0\} \\ =&\#\{(x,y) \in \mathrm{XDS}_n \wedge c=0\} - \#\{(x,y) \in \mathrm{XDS}_n \wedge c=0 \wedge \bar{c}=1\} \\ =&R_n - \frac{\#\mathrm{XDS}_n}{4} \end{aligned}$$

Thus in total the number of solutions over $n+1$ bits is

$$R_{n+1} = \frac{\#\mathrm{XDS}_n}{2} + 2\times\left(R_n - \frac{\#\mathrm{XDS}_n}{4}\right) = 2R_n$$

If $\alpha\oplus\beta\oplus\Delta = 0$, by induction, $R_n = \frac{\#\mathrm{XDS}_n}{2} + 2^{n-1}$. Since $\delta_{n-1} \notin \{000, 111\}$, by Lemma 3 we know that $\#\mathrm{XDS}_{n+1} = 2\times\#\mathrm{XDS}_n$, thus we have

$$R_{n+1} = 2R_n = 2\left(\frac{\#\mathrm{XDS}_n}{2} + 2^{n-1}\right) = \frac{\#\mathrm{XDS}_{n+1}}{2} + 2^n$$

as expected for the induction since here we would have $\tilde{\alpha}\oplus\tilde{\beta}\oplus\tilde{\Delta} = 0$.

If $\alpha\oplus\beta\oplus\Delta \neq 0$, by induction, $R_n = \frac{\#\mathrm{XDS}_n}{2}$. Again by Lemma 3, we get

$$R_{n+1} = 2R_n = 2\left(\frac{\#\mathrm{XDS}_n}{2}\right) = \frac{\#\mathrm{XDS}_{n+1}}{2}$$

as expected for the induction since here we would have $\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta} \neq 0$.

If $\delta_{n-1} \notin \{000, 111\}$ and $\chi = 1$ By Lemma 4, since $\chi = \alpha' \oplus \beta' \oplus \Delta' = 1$, we have $c \neq \bar{c}$. When $(x', y') = (0, 0)$, we can take any $(x, y) \in \{(x, y) \in \mathrm{XDS}_n \wedge c \neq \bar{c}\}$, thus $\frac{\#\mathrm{XDS}_n}{2}$ solutions. When $(x', y') = (0, 1)$, we can take any $(x, y) \in \{(x, y) \in \mathrm{XDS}_n \wedge c \neq \bar{c} \wedge c = 0\}$, which by Lemma 6 is of size $\frac{\#\mathrm{XDS}_n}{4}$.

Thus in total the number of solutions over $n+1$ bits is

$$R_{n+1} = \frac{\#\mathrm{XDS}_n}{2} + 2 \times \left(\frac{\#\mathrm{XDS}_n}{4}\right) = \#\mathrm{XDS}_n$$

Moreover, since $\alpha' \oplus \beta' \oplus \Delta' = 1$, we always have $\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta} \neq 0$, so we only have one case to consider. In addition, $\delta_{n-1} \notin \{000, 111\}$ so by Lemma 3, we have

$$R_{n+1} = \#\mathrm{XDS}_n = \frac{\#\mathrm{XDS}_{n+1}}{2}$$

If $\delta_{n-1} = 000$ and $\chi = 0$ By Lemma 4, since $\chi = \alpha' \oplus \beta' \oplus \Delta' = 0$, we have $c = \bar{c}$. When $(x', y') = (0, 0)$, we can take any $(x, y) \in \{(x, y) \in \mathrm{XDS}_n \wedge c = \bar{c}\}$, thus $\#\mathrm{XDS}_n$ solutions. When $(x', y') = (0, 1)$, we can take any $(x, y) \in \{(x, y) \in \mathrm{XDS}_n \wedge c = \bar{c} = 0\}$, which by Lemma 6 is of size

$$\begin{aligned}
&\#\{(x, y) \in \mathrm{XDS}_n \wedge c = \bar{c} = 0\} \\
=&\#\{(x, y) \in \mathrm{XDS}_n \wedge c = 0\} - \#\{(x, y) \in \mathrm{XDS}_n \wedge c = 0 \wedge \bar{c} = 1\} \\
=&R_n - 0 \quad \text{(Because } \#\mathbb{B} = 0 \text{ in Lemma 6)}
\end{aligned}$$

Thus in total the number of solutions over $n+1$ bits is $R_{n+1} = \#\mathrm{XDS}_n + 2R_n$.

If $\alpha \oplus \beta \oplus \Delta = 0$, by induction, $R_n = \frac{\#\mathrm{XDS}_n}{2} + 2^{n-1}$. Moreover, since $\delta_{n-1} = 000$, by Lemma 3 we know that $\#\mathrm{XDS}_{n+1} = 4 \times \#\mathrm{XDS}_n$, thus we have

$$R_{n+1} = \#\mathrm{XDS}_n + 2R_n = 2\#\mathrm{XDS}_n + 2^n = \frac{\#\mathrm{XDS}_{n+1}}{2} + 2^n$$

as expected for the induction since here we would have $\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta} = 0$.

If $\alpha \oplus \beta \oplus \Delta \neq 0$, by induction, $R_n = \frac{\#\mathrm{XDS}_n}{2}$. Again by Lemma 3, we get

$$R_{n+1} = \#\mathrm{XDS}_n + 2R_n = \#\mathrm{XDS}_n + 2\left(\frac{\#\mathrm{XDS}_n}{2}\right) = 2\#\mathrm{XDS}_n = \frac{\#\mathrm{XDS}_{n+1}}{2}$$

as expected for the induction since here we would have $\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta} \neq 0$.

If $\delta_{n-1} = 111$ and $\chi = 1$ By Lemma 4, since $\chi = \alpha' \oplus \beta' \oplus \Delta' = 1$, we have $c \neq \bar{c}$. When $(x', y') = (0, 0)$, we can take any $(x, y) \in \{(x, y) \in \mathrm{XDS}_n \wedge c \neq \bar{c}\}$, thus $\#\mathrm{XDS}_n$ solutions. When $(x', y') = (0, 1)$, we can take any $(x, y) \in \{(x, y) \in \mathrm{XDS}_n \wedge c \neq \bar{c} \wedge c = 0\}$, which by Lemma 6 is of size $\frac{\#\mathrm{XDS}_n}{2}$. Thus in total the number of solutions over $n+1$ bits is

$$R_{n+1} = \#\mathrm{XDS}_n + 2 \times \left(\frac{\#\mathrm{XDS}_n}{2}\right) = 2 \times \#\mathrm{XDS}_n$$

Moreover, since $\alpha' \oplus \beta' \oplus \Delta' = 1$, we always have $\tilde{\alpha} \oplus \tilde{\beta} \oplus \tilde{\Delta} \neq 0$, so we only have one case to consider. In addition, $\delta_{n-1} = 111$ so by Lemma 3, we have

$$R_{n+1} = 2 \times \#\mathrm{XDS}_n = \frac{\#\mathrm{XDS}_{n+1}}{2}$$

□

Combining Proposition 2 and Proposition 3, we can deduce the following proposition for $R_n^+(\alpha, \beta, \Delta) = \#\{(x, y) \in \mathrm{XDS}_n^+(\alpha, \beta, \Delta) \text{ s.t. } x + y < 2^n\}$.

Proposition 4. *With the previous notations, we have*

$$R_n^+(\alpha,\beta,\Delta) = \begin{cases} \frac{\#\mathrm{XDS}_n^+(\alpha,\beta,\Delta)}{2} & \text{if } \alpha\oplus\beta\oplus\Delta \neq 1\ldots1 \\ \frac{\#\mathrm{XDS}_n^+(\alpha,\beta,\Delta)}{2} + 2^{n-1} & \text{if } \alpha\oplus\beta\oplus\Delta = 1\ldots1 \end{cases}$$

Proof. This is straightforward from the previous propositions:

$$\begin{aligned} R_n^+(\alpha,\beta,\Delta) &= \#\{(x,y)\in \mathrm{XDS}_n^+(\alpha,\beta,\Delta) \text{ s.t. } x+y<2^n\} \\ &= \#\{(x,y)\in \mathrm{XDS}_n(\bar\alpha,\bar\beta,\bar\Delta) \text{ s.t. } x+y<2^n\} \\ &= \begin{cases} \frac{\#\mathrm{XDS}_n(\bar\alpha,\bar\beta,\bar\Delta)}{2} \text{ if } \bar\alpha\oplus\bar\beta\oplus\bar\Delta\neq 0 \\ \frac{\#\mathrm{XDS}_n(\bar\alpha,\bar\beta,\bar\Delta)}{2} + 2^{n-1} \text{ if } \bar\alpha\oplus\bar\beta\oplus\bar\Delta = 0 \end{cases} \\ &= \begin{cases} \frac{\#\mathrm{XDS}_n^+(\alpha,\beta,\Delta)}{2} \text{ if } \alpha\oplus\beta\oplus\Delta\neq 1\ldots1 \\ \frac{\#\mathrm{XDS}_n^+(\alpha,\beta,\Delta)}{2} + 2^{n-1} \text{ if } \alpha\oplus\beta\oplus\Delta = 1\ldots1 \end{cases} \end{aligned}$$

□

2.5 Combining the results and the main theorem

Corollary 1. *Let $\alpha,\beta,\Delta\in\mathbb{F}_2^n$ be valid differences such that either* $\mathrm{XDS}_n(\alpha,\beta,\Delta)$ *or* $\mathrm{XDS}_n^+(\alpha,\beta,\Delta)$ *is not empty. Let d be the amount of indices i such that $(\alpha_i,\beta_i,\Delta_i)$ are not all equal (i.e. different from 000 and 111)* excluding *the MSB. Then, in both cases we can write $R_n(\alpha,\beta,\Delta)$ (or $R_n^+(\alpha,\beta,\Delta)$ depending on which is nonzero) as a single expression*

$$2^{2n-d-1} + \mathbb{1}_{\alpha\oplus\beta\oplus\Delta\in\{0\ldots0,1\ldots1\}}\times 2^{n-1}$$

Proof. From Lemma 1, for a valid (α,β,Δ), we have $\#\mathrm{XDS}_n(\alpha,\beta,\Delta) = 2^{2n-d}$. Similarly, from Lemma 2, for a valid (α,β,Δ), we have $\#\mathrm{XDS}_n^+(\alpha,\beta,\Delta) = 2^{2n-d}$. Moreover, as for Proposition 1, if the differences are valid for XDS_n, then we need to have $\alpha_0\oplus\beta_0\oplus\Delta_0 = 0$ (otherwise the probability/solution count will be 0 anyway). Thus, one can still write

$$R_n(\alpha,\beta,\Delta) = 2^{2n-d-1} + \mathbb{1}_{\alpha\oplus\beta\oplus\Delta\in\{0\ldots0,1\ldots1\}}\times 2^{n-1}$$

for a valid difference since if we have $\alpha\oplus\beta\oplus\Delta = 1\ldots1$ in this case, the probability/solution count is going to be zero. Similarly, we can still write the same thing for $R_n^+(\alpha,\beta,\Delta)$ for a valid difference since if we have $\alpha\oplus\beta\oplus\Delta = 0\ldots0$, the probability/solution count is going to be zero. □

Thus, combining all of these results, we can finally conclude with the following theorem to evaluate the probability of

$$[(\overleftarrow{x}\oplus\alpha)\boxplus(\overleftarrow{y}\oplus\beta)]\oplus\overleftarrow{x\boxplus y} = \Delta$$

Theorem 2 (Main)**.** *For a valid differential (α,β,Δ) and with the notation introduced at the beginning of this section, as well as $\chi = \alpha\oplus\beta\oplus\Delta$. We have that the probability of the above equation to hold can be expressed as*

$$T'_k(\alpha_{L'},\beta_{L'},\Delta_{L'},\chi_0)\times T'_{n-k}(\alpha_{R'},\beta_{R'},\Delta_{R'},\chi_k)$$

where

$$T'_m(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\Delta},\mathbf{w}) = 2^{-d-1} + \mathbb{1}_{\boldsymbol{\alpha}\oplus\boldsymbol{\beta}\oplus\boldsymbol{\Delta}\in\{0\ldots0,1\ldots1\}}\times\left((-1)^{\mathbf{w}}\times 2^{-m-1}\right) \tag{8}$$

where d is the number of indices i such that $(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i,\boldsymbol{\Delta}_i)$ are not all equal (i.e. different from 000 and 111) excluding *the MSB.*

Proof. By Proposition 1, the statement of the theorem is equivalent to the claim that $T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, \mathbf{w})$ from the proposition can be computed as T'_m of Equation (8), assuming that it is nonzero. From the definitions of T_m and R_m, R^+_m, the fact that only one of the latter two can be nonzero, and the assumption that the differential is valid, we can deduce that

$$T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, 0) = (R_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}) + R^+_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}))/2^{2m}$$

which by Corollary 1 is equal to

$$2^{-d-1} + \mathbb{1}_{\boldsymbol{\alpha}\oplus\boldsymbol{\beta}\oplus\boldsymbol{\Delta}\in\{0\ldots0,1\ldots1\}} \times 2^{-m-1}$$

As mentioned in the beginning of Subsection 2.4, the case $T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, 1)$ (corresponding to the $\mathrm{xdp}_m/\mathrm{xdp}^+_m$ conditioned on the output carry being equal to 1) can be computed as the complement of $T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, 0)$ from the xdp or xdp^+, both of which for valid transitions are given by 2^{-d}. It follows that

$$T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, 1) = 2^{-d} - T_m(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, 0) = 2^{-d-1} - \mathbb{1}_{\boldsymbol{\alpha}\oplus\boldsymbol{\beta}\oplus\boldsymbol{\Delta}\in\{0\ldots0,1\ldots1\}} \times 2^{-m-1}$$

from where the expression T'_m for T_m follows.

□

2.6 Exact probability of RX-differentials through constant addition

For completeness of the RX-trail searching framework, it is necessary to consider RX-differential transitions through modular addition with a public constant. This amounts to counting the number of solutions $x \in \mathbb{F}_2^n$ to the equation

$$(\overleftarrow{x} \oplus \delta) \boxplus a \;\oplus\; \overleftarrow{x \boxplus a} = \Delta \tag{9}$$

where $\delta, \Delta, a \in \mathbb{F}_2^n$ are fixed.

As we shall see, the overlapping parts of the rotated constants vary depending on whether $k < n/2$ or $k > n/2$. To avoid considering the two cases, we show that they are equivalent up to an appropriate rotation of differences. The following proposition combined with the fact that $x \ggg k = x \lll (n-k)$ allows us to focus only on the case $k \leq n/2$.

Proposition 5. *Let $\delta, \Delta, a \in \mathbb{F}_2^n$ be fixed, together with the rotation amount k. Then, the number of solutions $x \in \mathbb{F}_2^n$ to each of the two following equations is equal (note that both left/right rotations use the same k):*

$$(\overleftarrow{x} \oplus \delta) \boxplus a \;\oplus\; \overleftarrow{x \boxplus a} = \Delta$$
$$(\overrightarrow{x} \oplus \overrightarrow{\delta}) \boxplus a \;\oplus\; \overrightarrow{x \boxplus a} = \overrightarrow{\Delta}$$

Proof. Solutions to the two equations are in bijection by $x' = \overleftarrow{x} \oplus \delta, x = \overrightarrow{x' \oplus \delta}$:

$$\begin{aligned}
&(\overleftarrow{\overrightarrow{x' \oplus \delta}} \oplus \delta) \boxplus a \;\oplus\; \overleftarrow{\overrightarrow{(x' \oplus \delta)} \boxplus a} = \Delta \\
\Leftrightarrow\;& x' \boxplus a \;\oplus\; \overleftarrow{\overrightarrow{(x' \oplus \delta)} \boxplus a} = \Delta \\
\Leftrightarrow\;& \overrightarrow{x' \boxplus a} \;\oplus\; \overrightarrow{(x' \oplus \delta)} \boxplus a = \overrightarrow{\Delta} \\
\Leftrightarrow\;& \overrightarrow{x' \boxplus a} \;\oplus\; (\overrightarrow{x'} \oplus \overrightarrow{\delta}) \boxplus a = \overrightarrow{\Delta}
\end{aligned}$$

□

From now on, we assume $1 \leq k \leq n/2$. Adapting the approach from Subsection 2.2, let $x = x_L||x_R$ with $x_L \in \mathbb{F}_2^k$ and $x_R \in \mathbb{F}_2^{n-k}$, and let $\delta = \delta_{L'}||\delta_{R'}, \Delta = \Delta_{L'}||\Delta_{R'}$ with $\delta_{L'}, \Delta_{L'} \in \mathbb{F}_2^{n-k}$ and $\delta_{R'}, \Delta_{R'} \in \mathbb{F}_2^k$. This partition is illustrated in Figure 2. Furthermore, let $a = a_L||a_M||a_{R'} \in \mathbb{F}_2^k \times \mathbb{F}_2^{n-2k} \times \mathbb{F}_2^k$. Then, we can rewrite Equation (9) as a system of equations:

$$\begin{cases} [(x_R \oplus \delta_{L'}) \boxplus (a_L||a_M) \boxplus c_R] \ \oplus \ [x_R \boxplus (a_M||a_{R'})] = \Delta_{L'} & \text{(10c)} \\ [(x_L \oplus \delta_{R'}) \boxplus a_{R'}] \ \oplus \ [x_L \boxplus a_L \boxplus c_L] = \Delta_{R'} & \text{(10d)} \\ c_R = 0 \quad \text{if } (x_L \oplus \delta_{R'}) + a_{R'} < 2^k, \quad 1 \text{ otherwise} & \text{(10e)} \\ c_L = 0 \quad \text{if } x_R + (a_M||a_{R'}) < 2^{n-k}, \quad 1 \text{ otherwise} & \text{(10f)} \end{cases}$$

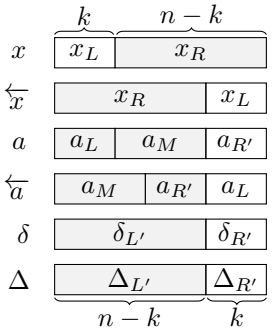

Figure 2: Splitting of the words in the equation of the RX-differential through modular addition with a constant. Parts involved in Equation (10c) are shaded.

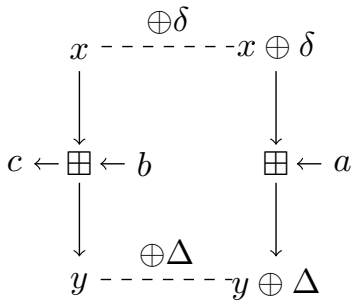

Figure 3: Differential over modular addition with two constants.

Observe that, as in the case of Proposition 1, by the linearity of the LSBs of the addition, the only possible values for the carries c_R, c_L can be deduced solely from the differences and constants:

$$c_R = (\delta_{L'} \oplus (a_L||a_M) \oplus a_{R'} \oplus \Delta_{L'})_0 \tag{11}$$
$$c_L = (\delta_{R'} \oplus a_{R'} \oplus a_L \oplus \Delta_{R'})_0 \tag{12}$$

Therefore, the system of equations above can be described by two independent systems (one on x_L and one on x_R) of the following form, parameterized by $a, b, \delta, \Delta \in \mathbb{F}_2^m, c \in \mathbb{F}_2$ (see Figure 3):

$$\begin{cases} (x \oplus \delta) \boxplus a \ \oplus \ x \boxplus b = \Delta \\ c = 0 \quad \text{if } x \boxplus b < 2^n, \quad 1 \text{ otherwise} \end{cases} \tag{13}$$

where $m = n-k, \delta = \delta_{L'}, \Delta = \Delta_{L'}, a = (a_L||a_M) \boxplus c_R, b = a_M||a_{R'}, c = c_L$ (using Eq. 11), $x = x_R$ for Equation (10c), and $m = k, \delta = \delta_{R'}, \Delta = \Delta_{R'}, a = a_L \boxplus c_L, b = a_{R'}, c = c_R$ (using Eq. 12), $x = x_L \oplus \delta_{R'}$ for Equation (10d) (note that we substitute $x = x_L \oplus \delta_{R'}$ to match the carry expression from Equation (10e)).

The first equation is similar to the usual (non-rotational) differential transition through modular addition with a constant, which was studied before in several works [Mac01, ARS+20, ARS+22]. The differences with our problem are that the two constants added to x and to $x \oplus \delta$ may in principle be different (e.g., $a = a_L||a_M, b = a_M||a_{R'}$ for Equation (10c)), and we have a constraint on the output carry of one of the additions. This makes the problem rather complicated. Note that even the restricted case considered in the previous works does not have a known compact expression, only a recursive state-machine-based

computation, which was nonetheless implemented in SMT solvers [ARS+20, ARS+22]. Finally, we remark that, based on our experiments, Equation (13) may have a number of solutions that is not expressible as a sum or difference of two powers of two (in contrast to the case of Corollary 1). We conclude that a general compact expression for the number of solutions of such systems may not exist or may be more complex. However, it is possible to efficiently compute this number by a general dynamic programming / state-machine-based approach, which we describe below. This method is similar to the directed acyclic graph approach from [MVCP10].

2.6.1 Computing probability of Equation (13)

Let us assume that $n \geq 1$ is fixed, together with the vectors $a, b, \delta, \Delta \in \mathbb{F}_2^n$. Define $c = c(x, b) = (c_n, \ldots, c_0)$ to be the vector of carries in the addition $x \boxplus b$ and $c' = (c'_n, \ldots, c'_0)$ to be the vector of carries in the addition $(x \oplus \delta) \boxplus a$:

$$c_i = \begin{cases} 0 & i = 0 \\ \mathrm{maj}(x_{i-1}, b_{i-1}, c_{i-1}) & 1 \leq i \leq n \end{cases} \tag{14}$$

$$c'_i = \begin{cases} 0 & i = 0 \\ \mathrm{maj}(x_{i-1} \oplus \delta_{i-1}, a_{i-1}, c'_{i-1}) & 1 \leq i \leq n \end{cases} \tag{15}$$

where $\mathrm{maj} : \mathbb{F}_2^3 \to \mathbb{F}_2$ is the majority function. Note that $c = c_n || [x \boxplus b \ \oplus \ x \oplus b]$ with c_n equal to 1 if and only if $x + b \geq 2^n$, and, similarly, $c' = c'_n || (x \oplus \delta) \boxplus a \ \oplus \ x \oplus \delta \oplus a]$. Using this notation, the differential condition of Equation (13) becomes

$$\delta_i \oplus \Delta_i \oplus a_i \oplus b_i \oplus c_i \oplus c'_i = 0 \tag{16}$$

In order to compute the number of solutions to the system, we define intermediate counts for each reduction of the equation to $\mathbb{F}_2^i$. Note that the differential constraint affects the intermediate counts for $i \leq n-1$, while for $i = n$ there is no constraint. This is exactly the same situation as in Lemma 1 and Lemma 2.

For $C \in \mathbb{F}_2$ define $T_i^C = T_i^C(\delta, \Delta, a, b)$ as

$$T_i^C = \begin{cases} 1 - C & i = 0 \\ \#\{x \in \mathbb{F}_2^i \text{ s.t. } (x \oplus \delta) \boxplus a \ \oplus \ x \boxplus b = \Delta, c_i(x, b) = C, & \\ \qquad c_i(x, b) \oplus c'_i(x, b) = \delta_i \oplus \Delta_i \oplus a_i \oplus b_i\} & 1 \leq i \leq n-1 \\ \#\{x \in \mathbb{F}_2^n \text{ s.t. } (x \oplus \delta) \boxplus a \ \oplus \ x \boxplus b = \Delta, c_n(x, b) = C\} & i = n \end{cases} \tag{17}$$

where $(x \oplus \delta) \boxplus a \ \oplus \ x \boxplus b = \Delta$ is computed over $\mathbb{F}_2^i$. Note that T_n^0 and T_n^1 correspond to the number of solutions to Equation (13) for the cases $c = 0$ and $c = 1$ respectively.

Theorem 3. *Let $n \geq 2$ and $1 \leq i \leq n-1$. Then, the counts T_i^0, T_i^1 satisfy the linear recursion*

$$\begin{pmatrix} T_i^0 \\ T_i^1 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{b_{i-1}} \times M_\rho \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{b_{i-1}} \times \begin{pmatrix} T_{i-1}^0 \\ T_{i-1}^1 \end{pmatrix}$$

with $M_\rho \in \mathbb{Z}^{2\times 2}$ given by

$$M_{000} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \quad M_{001} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad M_{010} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \quad M_{011} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$M_{100} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \quad M_{101} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad M_{110} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \quad M_{111} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and $\rho \in \mathbb{F}_2^3$ *given by*

$$\rho = (\delta_{i-1} \oplus a_{i-1} \oplus b_{i-1},\ \Delta_{i-1} \oplus \delta_{i-1},\ \Delta_i \oplus \delta_i \oplus a_i \oplus b_i \oplus a_{i-1} \oplus b_{i-1})$$

Furthermore,

$$\begin{pmatrix} T_n^0 \\ T_n^1 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{b_{n-1}} \times \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{b_{n-1}} \times \begin{pmatrix} T_{n-1}^0 \\ T_{n-1}^1 \end{pmatrix} \tag{18}$$

Proof. Fix i with $1 \le i \le n-1$. Let $x' = x_{i-1}||x \in \mathbb{F}_2^i$ be a solution counted by $T_i^{c_i(x')}$. Note that x must be counted by $T_x^{c_{i-1}(x)}$ (here, we use the fact that the condition in the definition of T_i^C is equivalent to the differential equation described in Equation (16)).

We first show that, for each x, the distribution of such extensions x' (across T_i^0, T_i^1 and outside of both) is determined solely by $c_{i-1}(x)$ and fixed values b_{i-1} and ρ as defined in the theorem. First, note that c_i can be computed directly using Equation (14) from x_{i-1}, b_{i-1} and c_{i-1}. The differential constraint (Equation (16)) can be rewritten using $\rho = (\rho_0, \rho_1, \rho_2)$ as

$$c_i' = c_i \oplus \rho_2 \oplus a_{i-1} \oplus b_{i-1} = c_i \oplus \rho_2 \oplus \rho_0 \oplus \delta_{i-1}$$

On the other hand, it must equal its definition from Equation (15):

$$\begin{aligned} c_i' &= \mathrm{maj}(x_{i-1} \oplus \delta_{i-1},\ a_{i-1},\ c_{i-1}') \\ &= \mathrm{maj}(x_{i-1} \oplus \delta_{i-1},\ \rho_0 \oplus b_{i-1} \oplus \delta_{i-1},\ c_{i-1} \oplus a_{i-1} \oplus b_{i-1} \oplus \delta_{i-1} \oplus \Delta_{i-1}) \\ &= \mathrm{maj}(x_{i-1} \oplus \delta_{i-1},\ \rho_0 \oplus b_{i-1} \oplus \delta_{i-1},\ c_{i-1} \oplus \rho_0 \oplus \rho_1 \oplus \delta_{i-1}) \\ &= \delta_{i-1} \oplus \mathrm{maj}(x_{i-1},\ \rho_0 \oplus b_{i-1},\ \rho_0 \oplus \rho_1 \oplus c_{i-1}) \end{aligned}$$

where we used (in order) definition of T_{i-1}, definition of ρ, and the fact that $\mathrm{maj}(y + \mathtt{111}) = 1 + \mathrm{maj}(y)$ for all $y \in \mathbb{F}_2^3$. By combining the two expressions of c_i', we get an equivalent to the differential constraint

$$c_i \oplus \rho_0 \oplus \rho_2 = \mathrm{maj}(x_{i-1}, \rho_0 \oplus b_{i-1}, \rho_0 \oplus \rho_1 \oplus c_{i-1}) \tag{19}$$

which shows the claim about the distribution (over both values of $x_{i-1} \in \mathbb{F}_2$).

We now prove the effect of b_{i-1} on the transition matrix. It is easy to see that, in the theorem's statement, the swap matrices present when $b_{i-1} = 1$ simply flip the input and the output carries c_{i-1} and c_i. To prove this, observe that the condition in Equation (19) when $b_{i-1} = 1$ is equivalent to the same condition with $b_{i-1} = 0$ and c_i, c_{i-1}, x_{i-1} flipped (again using the aforementioned property of maj). Note that flipping x_{i-1} does not change the distribution.

The transition of the counts T_{i-1}^0, T_{i-1}^1 into T_i^0, T_i^1 can be compactly described by the matrices M_ρ, which are defined and directly computed as

$$M_{\rho;y,z} = \#\{x_{i-1} \in \mathbb{F}_2 \text{ s.t. Eq. 19 holds with } b_{i-1} = 0, c_{i-1} = z, c_i = y\} \tag{20}$$

Finally, for the expression of T_n^0, T_n^1, the only difference is the absence of the differential condition, which means that we have to count both $\rho = (\rho_0, \rho_1, 0)$ and $\rho = (\rho_0, \rho_1, 1)$ (note that for $i = n$ we still have well-defined ρ_0 and ρ_1), since this corresponds to the partitioning of the set of all possible extensions x' according to possible "next" difference bits (i.e., at index n). This can also be seen by considering the two possible values of Δ_i. Then, it is left to observe by computation that $M_{yz0} + M_{yz1} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ for all $y, z \in \mathbb{F}_2$. □

This theorem can be of independent interest, since it describes the differential transition through modular addition with two different constants. For example, it may be useful in differential cryptanalysis of modular addition with different domain separation constants.

Summary of the method to compute the probability of RX-differential through constant addition We now summarize how to compute the probability of Equation (9). First, if $k > n/2$, then, by Proposition 5, we can replace the parameters by $\delta' = \delta \ggg k$, $\Delta' = \Delta \ggg k$, and $k' = n - k$, without changing the number of solutions. Then, compute the carries c_R, c_L from differences using Equations (11), (12). Count the number of solutions to each of the two subsystems of the form given in Equation (13) using Theorem 3. Finally, multiply the two numbers of solutions and divide by 2^n to compute the desired probability. We implemented and extensively verified this approach in practice.

2.7 Experimental verification

In order to ensure the correctness of our theorems, we have performed extensive computational verifications.

For our main result, Theorem 2 (and the validity criteria Theorem 1), we performed the following tests:

1. For all $2 \leq n \leq 8$, all rotational amounts $1 \leq k \leq n-1$, all 2^{2n} possible input differences, we computed the precise distribution of output differences by enumerating all 2^{2n} possible inputs and computing the corresponding output differences. For every differential $(\alpha, \beta) \to \delta$, we verified whether validity criteria correctly tell if it has zero or nonzero matching inputs. For every differential with at least one matching input, we verified that the number (divided by 2^{2n}) matches the theorem's claim. This requires 2^{4n} time per (n, k) pair.

2. For all $2 \leq n \leq 12$, all rotational amounts $1 \leq k \leq n-1$, we sampled 100 random input differences and performed the same verification as in the first step (for all output differences).

For the same ranges of parameters (exhaustive for $n \leq 8$ and random samples for $n \leq 12$) and using similar methods, we tested formulas for XDS/XDS$^+$ (Lemma 1, Lemma 2) and for R_n/R_n^+ (Proposition 3, Proposition 4). Furthermore, for these parameters, we verified that Theorem 3 matches the definition in (17), as well as the complete method for computing the exact probability of an RX-differential through constant addition (summarized at the end of Subsection 2.6) matches direct exhaustive probability computation.

Finally, for each $2 \leq n \leq 30$, we chose 1 000 000 random valid n-bit differential transitions and verified Proposition 2, Lemma 3, Lemma 4, Lemma 5, based on theoretical definitions. These experiments also support our intermediate statements, which are of independent interest.

3 Impact of the Correction on the Previous Formula

In this section, we study the difference between the previous approximate formula for the RX-probability [AL16] for the rotation $k = 1$. The case of [HXW22] for the rotations $k > 1$ will be studied in the applications section. We study the first case in detail due to several papers building upon this result [LWRA17, RLA17, XLSL19, Ran22]. As we show, the error factor for some transitions ranges from 1/2 to 3/2, that is, the correct transition probability may be twice lower or 50% larger than claimed.

First, we reproduce the original theorem behind RX-differential transitions through modular addition for rotation $k = 1$.

Theorem 4 ([AL16], under assumptions)**.** *Let $n \geq 2$ be an integer and fix $k = 1$. With the same notation as in Theorem 2, let*

$$\chi = \alpha \oplus \beta \oplus \Delta = (\chi_{L'} || \chi_{R'}) \in (\mathbb{F}_2^{n-1} \times \mathbb{F}_2^n)$$
$$\nu = (\alpha \oplus \beta) \vee (\alpha \oplus \Delta) = (\nu_{L'} || \nu_{R'}) \in (\mathbb{F}_2^{n-1} \times \mathbb{F}_2^n).$$

Then,

$$\begin{aligned}\Pr_{x,y\in\mathbb{F}_2^n}\left[\overleftarrow{x \boxplus y} \;\oplus\; (\overleftarrow{x} \oplus \alpha) \boxplus (\overleftarrow{y} \oplus \beta) = \Delta\right] \\ = \mathbb{1}_{(I\oplus \mathrm{SHL})(\chi_{L'})\oplus 1 \preccurlyeq \mathrm{SHL}(\nu_{L'})} \cdot 2^{-\mathrm{wt}(\mathrm{SHL}(\nu_{L'}))} \cdot 2^{-3} \\ + \mathbb{1}_{(I\oplus \mathrm{SHL})(\chi_{L'}) \preccurlyeq \mathrm{SHL}(\nu_{L'})} \cdot 2^{-\mathrm{wt}(\mathrm{SHL}(\nu_{L'}))} \cdot 2^{-1.415}\end{aligned}$$

Here, SHL *operates on* $(n-1)$*-bit words, so that the most significant bit is removed.*

Remark 1. We set $a_1 = b_1 = \Delta_1 = 0$ to avoid redundancy and $a_2 = \alpha, b_2 = \beta, \Delta_2 = \Delta$ in the original notation of [AL16].

Although the authors of [AL16] write "*Our goal is to estimate the transition probability*" before the theorem, and approximate the probabilities of outgoing carries by 1/2 "*for large* n", we will show that for certain RX-differential transitions the error in the probability can reach a factor of 2, which certainly requires a more precise statement. However, surprisingly, outside of this class of transitions, the theorem yields exact values despite using approximations.

Theorem 5. *Theorem 4 holds exactly when* $\chi_{L'} \notin \{0\dots0, 1\dots1\}$. *Furthermore, in the remaining cases, the correct probability can be obtained by multiplying the probability from Theorem 4 by the factor*

$$1 + (-1)^{\chi_0} 2^{d+1-n}$$

which ranges between its extreme values 1/2 (when $\chi_0 = 1$ *and* $d = n-2$*) and 3/2 (when* $\chi_0 = 0$ *and* $d = n-2$*).*

Proof. Consider Theorem 2 with $k = 1$. It is easy to see that the factor T_k is equal to $1/2+(-1)^{\chi_1}\times 2^{-2}$, which is equal to 1/4 and 3/4, respectively, for the cases $\chi_1 = 1$ or $\chi_1 = 0$. The factor T_{n-k} is equal to 2^{-d-1} with $d = \mathrm{wt}(\mathrm{SHL}(\nu_{L'}))$, whenever $\chi_{L'} \notin \{0\dots0, 1\dots, 1\}$. We obtain the correct probabilities 2^{-d-3} and $2^{-d-3}\cdot 3$ for the cases $\chi_1 = 1$ and $\chi_1 = 0$, respectively. Since the least significant bit of $\mathrm{SHL}(\nu_{L'})$ is equal to 0, these cases correspond precisely to the two indicators in the expression of Theorem 4, and thus the two theorems agree.

We now study the effect of the missing term when $\chi_{L'} \in \{0\dots0, 1\dots, 1\}$. Dividing the correct probability from Theorem 2 by the probability from Theorem 4 yields the ratio between the correct T_{n-k} and T_{n-k} with the missing second term, equal to

$$\frac{2^{-d-1} + (-1)^{\chi_0} 2^{-(n-1)-1}}{2^{-d-1}} = 1 + (-1)^{\chi_0} 2^{d+1-n}$$

as claimed. When d takes its maximum value $n-2$ (which does not contradict the condition on $\chi_{L'}$ or χ_0), we get the values 1/2 or 3/2. □

Remark 2. The incorrect probabilities with maximal correction factor are exactly equal to 2^{-n-1}, i.e., they are the least probable transitions. The correction factor decreases to 1 as the probability increases.

Remark 3. To our understanding, the issue in the statement of [AL16] comes from the fact that, despite considering the 4 cases of the *incoming* carries into the two chunks and estimating their probabilities (similarly to our Subsection 2.2), the authors computed the differential transition probabilities directly, without conditioning the *outgoing* carries. In other words, they do not consider the equations Equation (4e) and Equation (4f) in the equation system that decomposes the problem. In the case of balanced outgoing carries, only half of the counted solutions to the classic differential equations would satisfy the outgoing carry constraint, which is what we observe.

Remark 4. The discrepancy can also be seen from running the original theorem verification script referenced in [AL16][1]. Note that the script only counts experimentally the number of solutions for fixed input/output differences. By setting $a_1 = b_1 = \Delta_1 = 0$ (redundant differences) and $a_2 = \texttt{0x2556} \lll 7$, $b_2 = \texttt{0x0b21}$, $\Delta_2 = \texttt{0x2e76}$, the program outputs $1\,523\,712 = 2^{20.54}$ values (probability $2^{20.54-32} = 2^{-11.46}$), which could not be output by their theorem (it has either an integral weight or a fractional part -0.415).

Example 1. Let $n = 16, k = 1$. Consider the RX-differential transition

$$\begin{aligned}\alpha &= \texttt{1110000000010011}\\ \beta &= \texttt{0001001110100001}\\ \Delta &= \texttt{0000110001001101}\\ \chi &= \texttt{1111111111111111}\\ \nu &= \texttt{1111111111111110}\end{aligned}$$

We have $d = \mathrm{wt}(\mathrm{SHL}(\nu_{L'})) = n - 2 = 14$. Theorem 4 claims probability $2^{-d} \cdot 2^{-3} = 2^{-17}$, while the correct probability by Theorem 2 is equal to $(2^{-d-1} - 2^{-n}) \cdot 1/4 = 2^{-18}$. Exhaustive verification confirms $2^{32-18} = 2^{14}$ solutions.

We now explore the question of how many RX-differential transitions are affected by the error.

Proposition 6. *Let $n > k = 1$. There exist at least $16 \cdot 3^{n-2}$ valid RX-differential transitions for which the correction factor from Theorem 5 is equal to 1/2 (and the same amount when it is equal to 3/2).*

Proof. Fix $\chi_{L'} = 0 \ldots 0 \in \mathbb{F}_2^{n-1}$ and $\nu_{L'}$ equal to 1 in all positions except maybe the most significant bit. Then, the validity criteria are satisfied and the correction factor applies. We show how to construct at least $8 \cdot 3^{n-2}$ transitions that fit this constraint. For each position i, $1 \le i \le n-2$, the triple $(\alpha_i, \beta_i, \Delta_i)$ must sum to 0 and not all three must be equal. Exactly the 3 permutations of $(0, 1, 1)$ satisfy the constraints. For the most significant bit $i = n - 1$ we do not have the constraint on ν_{n-1} (but χ_{n-1} still must be 0), adding another solution $(0, 0, 0)$. Finally, observe that any assignment to $(\alpha_0, \beta_0, \Delta)$ is valid and only affects the value of χ_0 that, in turn, affects the sign of the term (4 assignments per each sign). We obtain $4 \times 3^{n-2} \times 4$ assignments. The case of $\chi_{L'} = 1 \ldots 1$ is fully symmetric and double the number of constructed transitions. □

Remark 5. This proposition covers the least probable transitions (with $d = n-2$), but there are many other affected transitions (with $\chi_{L'} \in \{0 \ldots 0, 1 \ldots 1\}$). As mentioned above, the correction factor tends to 1 as d decreases and the transition probability increases.

We conclude that $\Omega(3^n)$ out of 8^n transitions are affected, which is an exponentially large number but also still an exponentially small fraction of all transitions. The following proposition completes the comparison by counting all *valid* RX-differential transitions.

Proposition 7. *The number of valid RX-differential transitions for $n > k \ge 1$ is equal to $64 \times 7^{n-2}$.*

Proof. The proof is based on the validity criteria Theorem 1 and induction on the i for i-bit suffixes of differences, $1 \le i \le n$. We will maintain two counters $(c_{i,0}, c_{i,1})$, one for the number of valid i-bit suffixes with $\nu_{i-1} = 0$ (all 3 bits are equal) and one for the case $\nu_{i-1} = 1$ (not all 3 bits equal).

[1] http://homes.esat.kuleuven.be/~tashur/Rotational_Cryptanalysis_in_the_Presence_of_Constants.zip

(base case) For $i = 1$, all suffixes are valid and we have 2 and 6 transitions, respectively, for the two values of ν_{i-1}.

(induction $i \to i+1 \leq k$) We now study how the valid suffixes of length i can be extended to length $i+1$. Note that due to the use of SHL in the criteria, ν_{i-1} determines whether there will be any restriction in position i (length $i+1$)). The restriction consists in one XOR condition on the sum of the most significant bits at position i, meaning that it exactly halves the number of possible candidates, both for cases, independently of ν_i (for both cases of ν_i the XOR condition halves the number of candidates). Thus, each of the $c_{i,0}$-counted suffixes contributes 1 and 3 to counters $c_{i+1,0}, c_{i+1,1}$ respectively, and each of the $c_{i,1}$-counted suffixes contributes 2 and 6 to counters $c_{i+1,0}, c_{i+1,1}$ respectively:

$$\begin{pmatrix} c_{i+1,0} \\ c_{i+1,1} \end{pmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix} \times \begin{pmatrix} c_{i,0} \\ c_{i,1} \end{pmatrix} \tag{21}$$

(induction $i = k \to i = k+1$) This case is similar to the base case since there are no constraints. We get $c_{k+1,0} = 2(c_{k,0} + c_{k,1})$ and $c_{k+1,0} = 6(c_{k,0} + c_{k,1})$.

(induction $i > k \to i+1$) This case is the same as the induction $i \to i+1 \leq k$.

Observe that the ratio $c_{i,1} = 3c_{i,0}$ is maintained both through the base step and through the induction steps. Furthermore, the sum of the two counters starts at 8 in the base step, and is multiplied by 7 in the induction cases excluding $i = k \to i = k+1$, in which case it is multiplied by 8. It follows that the final count is equal to $8 \cdot 8 \cdot 7^{n-2}$. □

As a result of independent interest, we obtain an average probability of a valid RX-differential transition.

Corollary 2. *Let $n \geq 2$ and let $(\alpha, \beta, \Delta) \in \mathbb{F}_2^n$ be an RX-differential transition through modular addition chosen uniformly at random among all valid (nonzero probability) transitions. Then, the expected probability of the chosen transition is equal to*

$$\frac{49}{64} \cdot \left(\frac{4}{7}\right)^n$$

Proof. For each pair of input differences (α, β), the sum of the probabilities of transitions (α, β, δ) over all valid output differences δ is equal to 1. Therefore, the sum of probabilities of all valid transitions is equal to the total number of input differences, which is 2^{2n}. It follows that the average probability is 2^{2n} divided by the number of valid transitions, which is $64 \cdot 7^{n-2}$, which can be written as in the statement. □

Example 2. The average trail probability for $n = 16$ is equal to $2^{-13.3}$ and for $n = 32$ is equal to $2^{-26.22}$.

Impact on published results After a quick search we did not find any published RX-differential trail that is noticeably affected by the error. Low-probability transitions are unlikely to have the difference XOR equal to all-zeros or all-ones, while high-probability sparse transitions might hit the all-zero differences' XOR but would not have noticeable correction (2^{d+1-n} being very small).

On the other hand, all published searches for optimal trails based on the theorem from [AL16] can no longer be claimed to be optimal, since they might be missing a better trail affected by the correction.

Experimental verification We verified experimentally the claims in this section. For small values of $n \leq 12$, the experimental (exhaustive) number of solutions to the RX-differential equation always matches our theorem and is related to the output of the theorem of [AL16] exactly as Theorem 5 predicts. We also verified the construction in Proposition 6 for the same values of n, and the values in Proposition 7, Corollary 2 are verified for $n \leq 9$. All verification programs are provided in the code repository.

4 MILP-based Search for RX-trails

Automated search for cryptanalytic attacks is a very active branch of research. In this section, we show that our probability formula from Theorem 2 can be modeled by a compact MILP (Mixed Integer Linear Programming) system. This allows efficient automated search for best RX-differential trails, for any rotation amount k.

4.1 MILP model for the validity criteria (Theorem 1)

Theorem 1 allows us to characterize in which cases the modular addition of an RX-difference (α, k) with an RX-difference (β, k) propagates with a nonzero probability to an RX-difference (Δ, k). The goal now is to be able to describe this characterization using linear inequalities in order to fit them into an MILP model. We denote by $\mathbf{z} = \mathbf{XOR}(\mathbf{x}, \mathbf{y})$ the process of modeling the relation $z = x \oplus y$, which can be done in 4 inequalities as shown in Proposition 11 in Appendix I.

To model the propagation through the modular addition, one could simply unroll the whole expression of Theorem 1, which leads to a lot of inequalities and extra variables. Instead, consider the following boolean function $f : \mathbb{F}_2^6 \to \mathbb{F}_2$

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \begin{cases} 1 & \text{if } (x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6) \\ & \quad \leq ((x_4 \oplus x_6) \vee (x_5 \oplus x_6)) \\ 0 & \text{otherwise,} \end{cases}$$

then we want to guarantee that for $1 \leq i < n$, $f(\alpha_i, \beta_i, \Delta_i, \alpha_{i-1}, \beta_{i-1}, \Delta_{i-1}) = 1$. We notice that f evaluates to 0 in only the eight following points

$$(1,0,0,0,0,0), (0,1,0,0,0,0), (0,0,1,0,0,0), (1,1,1,0,0,0),$$
$$(0,1,1,1,1,1), (1,0,1,1,1,1), (1,1,0,1,1,1), (0,0,0,1,1,1)$$

thus we can model $f(x_1, x_2, x_3, x_4, x_5, x_6) = 1$ with a set of 8 inequalities (each removing one point) with standard MILP modeling techniques (see Proposition 13 in Appendix I). We denote this system by $\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6)$, and thus we can model the propagation of RX-differences as follows.

Proposition 8. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$, $0 \leq k < n$ and $\mathbf{a}, \mathbf{b}, \boldsymbol{\delta}$ be vectors of n binary variables modeling α, β, and Δ, respectively. Then, we can model the relation $[(\alpha, k), (\beta, k)] \xrightarrow{\boxplus} (\delta, k)$ with the set of inequalities*

$$\begin{cases} \boldsymbol{\delta}_0 = \mathbf{XOR}(\mathbf{a}_0, \mathbf{b}_0) \\ \mathbf{f}(\mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\delta}_i, \mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \boldsymbol{\delta}_{i-1}) \quad 1 \leq i < n \end{cases}$$

If $k > 0$, the first constraint and the constraint for $i = k$ are omitted.

When the weight of the differential is still needed (see Section 5), we can still use only 8 inequalities per bit, although of a more complicated shape and larger coefficients. These 8 inequalities were found using the tool from [Udo21], we denote them by $\mathbf{g}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7)$ and for completeness they are given in Proposition 14 in Appendix I. This improves the state-of-the-art of 13 inequalities per bit from [FWG+16].

Proposition 9. *Let $\alpha, \beta, \Delta, \eta \in \mathbb{F}_2^n$, $0 \leq k < n$ and $\mathbf{a}, \mathbf{b}, \boldsymbol{\delta}, \boldsymbol{\eta}$ be vectors of n binary variables modeling α, β, Δ and η, respectively. Then we can model the relations $[(\alpha, k), (\beta, k)] \xrightarrow{\boxplus} (\delta, k)$ and $\eta = \mathrm{SHL}((\alpha \oplus \beta) \vee (\alpha \oplus \delta)$ with the set of inequalities*

$$\begin{cases} \boldsymbol{\delta}_0 = \mathbf{XOR}(\mathbf{a}_0, \mathbf{b}_0) \\ \boldsymbol{\eta}_0 = 0 \\ \mathbf{g}(\mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\delta}_i, \mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \boldsymbol{\delta}_{i-1}, \boldsymbol{\eta}_i) \quad 1 \leq i < n \end{cases}$$

Proof. By computation, it can be checked that $\mathbf{g}(\mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\delta}_i, \mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \boldsymbol{\delta}_{i-1}, \boldsymbol{\eta}_i) \subseteq \{0,1\}^7$ holds if and only if $\boldsymbol{\eta}_i$ matches the definition of η_i (i.e. $\boldsymbol{\eta}_i = 0$ if and only if $\mathbf{a}_{i-1} = \mathbf{b}_{i-1} = \boldsymbol{\delta}_{i-1}$) and $\mathbf{a}_i \oplus \mathbf{b}_i \oplus \boldsymbol{\delta}_i = \mathbf{a}_{i-1} \oplus \mathbf{b}_{i-1} \oplus \boldsymbol{\delta}_{i-1}$ whenever $\boldsymbol{\eta}_i = 0$. □

4.2 MILP model for the T_n function

First, recall the expression for T_n:

$$T_n(\alpha, \beta, \Delta, v) = 2^{-d-1} + \mathbb{1}_{\alpha\oplus\beta\oplus\Delta\in\{0\dots0,1\dots1\}} \times \left((-1)^v \times 2^{-n-1}\right)$$

where $\alpha, \beta, \Delta \in \mathbb{F}_2^n$, $v \in \mathbb{F}_2$ and $d = \mathrm{wt}(\mathrm{SHL}((\alpha \oplus \beta) \vee (\alpha \oplus \Delta))$. In the MILP model, α, β, Δ are represented as vectors of binary variables, v is a binary variable, and the T_n output is represented by a real variable $w = -\log_2 T_n(\alpha, \beta, \Delta, v)$. We also introduce the following auxiliary variables:

- binary vector $\chi = \alpha \oplus \beta \oplus \Delta \in \mathbb{F}_2^n$;
- binary variable $a = \mathbb{1}_{\alpha\oplus\beta\oplus\Delta\in\{0\dots0,1\dots1\}} = \mathbb{1}_{\chi\in\{0\dots0,1\dots1\}} \in \mathbb{F}_2$;
- binary vector $\eta = \mathrm{SHL}((\alpha \oplus \beta) \vee (\alpha \oplus \Delta)$;
- integer variable $d = \sum_{i=0}^{n-1} \eta_i$;
- real variables $\ell_0 = \log_2(1 + 2^{-n-d})$, $\ell_1 = \log_2(1 - 2^{-n-d})$;
- real variable $\ell = v\ell_1 + (1-v)\ell_0$.

Then, the weight w can be expressed as

$$\begin{aligned} w &= -\log_2 T_n(\alpha, \beta, \Delta, v) && (22)\\ &= -\log_2(2^{-d-1} + a(-1)^v 2^{-n-1}) && (23)\\ &= d + 1 - \log_2(1 + a(-1)^v 2^{-n+d}) && (24)\\ &= d + 1 - a\log_2(1 + (-1)^v 2^{-n+d})) && (25)\\ &= d + 1 - a\ell. && (26) \end{aligned}$$

Finally, since a is binary, the multiplication can be rewritten using a big-M multiplexing system:

$$\begin{cases} w - aM \;\le\; d+1 \;\le\; w + aM \\ w - (1-a)M \;\le\; d+1-\ell \;\le\; w + (1-a)M \end{cases}$$

where M is a sufficiently large constant (e.g., $M = 2n$ suffices). This system of inequalities is equivalent to $w = d + 1$ when $a = 0$ and $w = d + 1 - \ell$ when $a = 1$. Thus, it is sufficient for modeling the variables d, a and ℓ.

Model for an "all-equal" variable Let $x_1, \dots, x_m, y$ be binary variables. The goal is to model the constraint "$y = 1$ if and only if $x_1 = \dots = x_m$", which we denote by $\mathrm{eq}(x_1, \dots, x_m, y)$. In particular, this is needed to compute the variable d in the Lipmaa-Moriai theorem and in our expression T_n. This case only requires $m = 3$ and was already solved in [FWG+16], although combined with the validity criteria (requiring 13 inequalities). However, we also need the case of a larger m to model the a variable. More precisely, we need $m = n - k$ or $m = k$ for rotational cryptanalysis. We propose a generic model with 4 inequalities valid for any number of variables m.

Proposition 10. *Let $x_1, \ldots, x_m, y$ be binary variables. The set of solutions to the following system is precisely characterized by "$y = 1$ if and only if $x_1 = \ldots = x_m$":*

$$\begin{cases} (x_1 + \ldots + x_{m-1}) - (m-1)x_m + (m-1)(1-y) \ \geq \ 0 \\ -(x_1 + \ldots + x_{m-1}) + (m-1)x_m + (m-1)(1-y) \ \geq \ 0 \\ x_1 + \ldots + x_m + y \ \geq \ 1 \\ -(x_1 + \ldots + x_m) + y \ \geq \ -m + 1 \end{cases}$$

Proof. The first two inequalities are trivially satisfied when $y = 0$, and are equivalent to $(x_1 + \ldots + x_{m-1}) = (m-1)x_m$ when $y = 1$, which is only possible when $x_1 = \ldots = x_m$. In other words, they model the implication $y = 1 \ \Rightarrow \ x_1 = \ldots = x_m$. Similarly, the last two inequalities model the implication $y = 0 \ \Rightarrow 1 \leq x_1 + \ldots + x_m \leq m - 1$. □

Model for η and d This variables are standard in modeling the differential transitions through modular addition, as they appear in the Lipmaa-Moriai theorem [LM01]. Equivalently, these can be modeled by $n - 1$ constraints

$$\{\mathrm{eq}(\alpha_i, \beta_i, \Delta_i, 1 - \eta_i)\}_{0 \leq i \leq n-2} \ \wedge \ d = \eta_0 + \ldots + \eta_{n-1}$$

Model for χ and a The vector χ can be modeled as the element-wise XOR of 3 vectors by standard techniques using 8 inequalities (see Proposition 12), and the variable a is characterized by $\mathrm{eq}(\chi_0, \ldots, \chi_{n-1}, a)$.

Model for the logarithms The logarithm function can be modeled using a so-called piecewise-linear function constraint (PWL), which essentially allows a lookup-table-style constraint for an integer variable with small domain. For all values of $d \in \{0, \ldots, n-1\}$, we precompute $\ell_0(d) = \log_2(1 + 2^{-n-d})$ and $\ell_1(d) = \log_2(1 - 2^{-n-d})$, and these tables form the PWL. Note that these functions are convex which may potentially improve model performance[2]. Finally, the variable ℓ can be modeled using a big-M multiplexing constraint:

$$\begin{cases} \ell - vM \ \leq \ \ell_0 \ \leq \ \ell + vM \\ \ell - (1-v)M \ \leq \ \ell_1 \ \leq \ \ell + (1-v)M \end{cases}$$

which is equivalent to $\ell = \ell_0$ when $v = 0$ and $\ell = \ell_1$ when $v = 1$.

The case $n = 1$ When $n = 1$, everything is a lot simpler and we do not need a PWL constraint. Indeed in that case, we always have $a = 1$ and $d = 0$, and thus we always have $w = 1 - \ell$. By precomputing $\ell_0 = \log_2(1 + 2^{-n})$ and $\ell_1 = \log_2(1 - 2^{-n})$, we can use the same kind of multiplexing constraint as in the previous paragraph to have $w = 1 - \ell_0$ when $v = 0$ and $w = 1 - \ell_1$ when $v = 1$.

5 Applications

We modeled several ciphers using the sets of inequalities from the previous section and ran these experiments for different values of the rotation k. Note that in our experiments, solving these models was quite slow even when using a 96-core server. As a result, we made the following two changes.

Before actually searching for the best trail (in terms of probability), we first solve a heuristic model where instead of maximizing the probability (i.e. minimizing $-\log(p)$), we

[2] See https://www.gurobi.com/documentation/current/refman/objectives.html#subsubsection:PiecewiseObj

minimize the total number of "not-all-equal" (NEQ) variables in the RX-differentials over modular additions. Indeed, if we look at the formula for the T_n function from Theorem 2, we can observe that it is essentially equal to 2^{-d-1} with a small deviation, where d is the number of bits that are "not-all-equal" in the RX-differential over the modular addition. What we are essentially doing here is just approximating the probability of each RX-differential propagation through the modular addition only using this quantity d and ignoring the other terms. This results in a much simpler model to compute the objective, and we use that to quickly obtain a "reasonably" good initial trail. Note that we do not need to completely optimize this heuristic model, as (contrary to what one could expect), minimizing this quantity doesn't necessarily lead to a trail with the best probability. However, this allows use to get a solution that can be used as a starting point for the *complete* model (whose goal is now to find a trail with the best probability), and in practice, in the same amount of time, the heuristic model gave us higher quality (i.e. better probability) solutions than solving the complete model from scratch.

Armed with this heuristic model to find a good starting point for the second model, we expected this to be enough to solve the complete model to optimality. Unfortunately, the complete model is still (in most cases) rather hard to solve to optimality and we had to limit how much time we spent on each case so that we are able to give results with multiple parameters (in particular, multiple values of k). In case of Alzette, we decided to limit it to 1 hour for the heuristic model (which was surprisingly efficient to get decent solutions already) and then 2 additional hours for the complete model to try to improve the probability to something better than what the heuristic model found. For the first constant c_0 and all 16 rotations we allowed 4 hours for the complete model. For other targets (such as Speck and Salsa) we used 5 + 55 minutes split. Note that we are quite limited in time due to the large number of instances to be solved: for example, for Alzette we consider 128 different instances (8 constants $\times$ 16 rotation amounts).

For an example of an evolution of bounds with respect to time, see Figure 4 depicting the case of $\mathsf{Alzette}_{c_0}$ and $k = 6$. Here, the best known solution $d = 29$ was obtained after only 4 minutes of search, and spending 1 more hour did not improve the NEQ heuristic (only slightly improved the lower bound). Although for a few instances the improvements happened near the end of the time limit, we could not afford to spend more time on all constants due to the large number of instances. All the corresponding evolution graphs are available in the code repository.

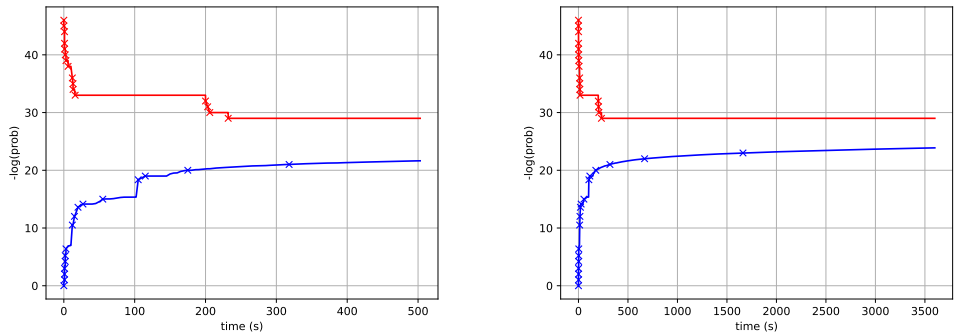

Figure 4: Evolution of lower (blue) and upper (red) bounds on the NEQ-weight of the trail, on $\mathsf{Alzette}_{c_0}$ with rotation $k = 6$. Left: first 500 seconds, right: 1 hour. Crosses denote new integer lower bounds.

Verifying validity In all our experiments, we also searched for a pair of plaintext that would verify the trail, so that we are at least sure that the trail is not actually impossible. To do so, we followed the same idea as [SRB21], creating an MILP model that models the transformation of the two plaintexts, with the added constraints that they need to follow the trail resulting from the previous model. For Alzette and Speck, we only did this verification at the end of the probability optimization and it found a pair of plaintexts most of the time. For Salsa, we had to perform more modifications due to the specific structure of trails for Salsa, see Section 5.2.

5.1 Alzette

Alzette is a 64-bit ARX-based S-box proposed by Beierle et al. [BBdS+20a] and part of the SPARKLE [BBdS+20b] permutation. We recall the structure of the Alzette permutations in Appendix A.

Due to its large size, one cannot expect to compute the usual properties of an S-box like the DDT in reasonable time, and the authors explicitly welcome tools to examine its specific properties. Alzette depends on a 32-bit constant, each constant defining a new permutation, and we focused on the eight constants carefully chosen by the designers, which are the ones used in SPARKLE. The resulting permutation when using constant c is denoted $\mathsf{Alzette}_c$.

In [HXW22], the authors give an (erroneous) formula to compute the probability of RX-differentials through the modular addition and gave some results on Alzette. In particular, they claimed to have found a 4-round trail over $\mathsf{Alzette}_{c_0}$ with $c_0 =$ 0xb7e15162 with probability $2^{-37.66}$, with the input/output differences being

$$(\texttt{0x0841fa08}, \texttt{0x0420f900}) \rightarrow (\texttt{0xdc33f048}, \texttt{0xbfee1412})$$

and without giving the intermediate values for the trail. Using our MILP modeling we were able to find (at least) one trail matching this input/output, and evaluating the probability of this trail with [HXW22] formula leads to a probability of $2^{-37.50}$ (thus it might not be this exact trail they found, although it has a slightly higher probability already). However, when evaluating the probability of this trail with our (correct) formula, the probability ends up being $2^{-36.66}$.

Moreover, in [HXW22, Tab.3], the authors gave the best probability they found over 4 rounds for $\mathsf{Alzette}_{c_0}$ with $k \in \{1, 8, 15, 16\}$. In Table 11 (Appendix D), we give the best trails we were able to find for every value of k and in particular, we can observe that we found trails with much better probabilities than claimed in [HXW22]. We give the comparison for $k \in \{1, 8, 15, 16\}$ in Table 2. Note that for our trails, the probability shown here was the best one we could find in a few hours of calculations, but was not proven to be optimal by the solver. The table also reports the experimental probability of distinguishers. These numbers can differ from trail probabilities due to dependence of consecutive rounds, clustering effect (small, see below), and measurement error (also expected to be small). Thus, any strong discrepancy has to come from the rounds' inter-dependence. Overall, we give the results for the best trails we were able to find for each of the 8 specified constants for Alzette and for every value of k between 1 and 16 (half the word size) in Table 11 through 18 in Appendix D.

On the clustering effect During our experimental verifications, we computed both the probability of the differential (input and output RX-difference match) and the probability of the trail (all intermediate RX-differences match). Among all experimental verifications, we observed a very small clustering effect (i.e., encryption pairs satisfying the input/output RX-difference but not all of the intermediate ones). For the cases given in Table 2, we observed no clustering for the trails given for $k = 1$ and $k = 8$; for $k = 15$ we found 46

encryption pairs following the trail and 2 encryption pairs only satisfying the differential; for $k = 16$ we found 293 and 9 pairs respectively. For the latter two cases, we have the experimental differential probability $2^{-36.42}$ and $2^{-33.76}$ versus the experimental trail probability $2^{-36.48}$ and $2^{-33.81}$. Technical details and verification logs are available in the supporting code of this work.

Table 2: Comparison of the probability of the best trail found by [HXW22] and our work, for $\mathsf{Alzette}_{c_0}$ with $k \in \{1, 8, 15, 16\}$. The values given are the negated base-2 logarithms of the probabilities. The last column refers to the 4-round distingusher experimental probability over 2^{42} pairs.

k	[HXW22]	Ours (trail)	Ours (exper.)
1	37.66	33.66	35.26
8	46.98	40.00	38.30
15	43.00	37.00	36.42
16	44.00	34.00	33.76

An example with $k \neq 1$ being stronger For five out of eight constants, the best trails among those we found are given for $k \neq 1$. This shows that rotation amounts $k \neq 1$ often provide superior trails compared to $k = 1$. In particular, for the constant $c_7 = \texttt{0xc2b3293d}$, the best trails we found for $\mathsf{Alzette}_{c_7}$ have probability $2^{-37.66}$ for $k = 1$ and probability $2^{-32.44}$ for $k = 2$ (best over all k). The CASCADA tool [Ran22, RR22] also found the trail for $k = 1$ within 5 hours on a single core and claims that its optimal[3].

We performed experimental verification of the aforementioned trail and observed probabilities $2^{-32.79}$ for the case of $k = 2$ and $2^{-37.83}$ for the case of $k = 1$ (over 2^{42} encrypted pairs each), matching well the trail probability. On the other hand, we also observed cases in which the experimental probability decreased due to dependencies between rounds. The involved trails, together with reproducible verification code and logs, are provided in the supporting code.

5.2 Salsa

Salsa [Ber08b] is a stream cipher that is the precursor to Chacha [Ber08a]. It has an internal state of 512 bits, split into 16 32-bit words which are organized in a 4×4 array. This internal state is updated through the use of the so-called "Quarter Round" function $QR(a, b, c, d)$ defined as described in Appendix B.

Note that the initial state has four constant words, placed on the diagonal of the state. This means that the first round is quite special, as one word in each QR function will be fixed to a constant. Due to this, as well as to the large state, we decided to focus only on the first round. By doing so, each QR function in the first round is independent, and we can focus on each one by one.

Note that since we have this constant word, searching for trails becomes a bit more complicated. First, we need to ensure that the constant word is actually used to build the trail. However due to the complexity of modeling the modular addition with a constant (see Section 2.6) we did not model the modular addition with a constant as such. Instead, we fixed the RX-difference to the appropriate difference for this constant. For example, for the first QR function, the first word is fixed to the constant $\texttt{0x61707865}$, thus in the model for the first QR function, we fix the RX-difference of the first word of the input to $\texttt{0x61707865} \oplus (\texttt{0x61707865} \lll k)$ depending on the rotation offset k considered. Note

[3]The performance of CASCADA varied significantly for different Alzette constants: for some constants it takes 20 minutes to find the optimal trail, for some constants it didn't find the best trail after a full day of computations.

that this results in a search space that is bigger than it should be for trails, so the resulting probabilities are only an estimation of the actual probability.

Moreover, because of this constant, finding a pair of matching plaintexts for a given trail turned out to be very unreliable if we only did this verification after optimizing the probability, as many trails end up being invalid. To counter that, we implemented the search for a pair of matching plaintexts *each time* a trail was found by the solver. This is made possible by the *Callback* functionality of the Gurobi solver, which allows the user to execute some code during the solving process *without interrupting it*. Thus each time the solver found a candidate trail during the optimization process, we ran the search for a matching pair of plaintexts. If no such pair exists, we deem this trail invalid and add constraints (called *Lazy Constraints* in Gurobi) to indicate to the solver that this solution was in fact, not a valid solution. This does slow down the solving process significantly but means that the resulting trails and probabilities are likely to be much more accurate. We give the results for each QR in Table 7 through 10 in Appendix C.

In the end, since the QR functions of the first round are independent, we can easily derive the probability of the best trail we can obtain for each k based on the best trails we found for each quarter round by simply multiplying the probabilities (i.e. summing the logs). While not all the best trails we found were proven to be optimal, we still also have lower bounds on the best possible value for the log of the probability. This results in the probabilities and respective lower bounds given in Table 3, and we can observe that the best probability is obtained with $k = 8$.

Table 3: Best trails we found on the first round of Salsa, as well as the best lower bounds proven by the solver.

k	bestLB	-log2(prob)	k	bestLB	-log2(prob)
1	99.5563	112.6405	9	101.6015	108.8935
2	100.5827	114.5341	10	104.1313	116.9068
3	107.6998	116.7500	11	106.0595	110.9481
4	100.9024	105.4544	12	108.4113	118.9285
5	102.3382	116.4515	13	100.7455	109.9718
6	107.4087	116.6232	14	100.0059	113.9945
7	109.5945	112.6569	15	104.8363	113.9892
8	95.4763	103.7891	16	104.4756	119.9856

5.3 Speck

The Speck family of block ciphers was designed by the NSA for lightweight software applications [BSS+15]. It can be parameterized by n, the word size, and m the number of key words, leading to Speck$2n/mn$ with block size $2n$ and key size mn. We searched for good trail for up to 6 rounds on Speck and, as it is a block cipher, the resulting trails are in the *related-key* model, where we allow RX-differences both in the plaintext and in the key.

Due to the nature of Speck as a block cipher, when searching for trails over r rounds, we can exploit results on $r' < r$ rounds to help the solver find solutions and/or prove lower bounds. Indeed, assume that we found some lower bounds for r' rounds (either for the heuristic on the number of not-all-equal bits, or on the actual search for the probability), then we can derive Matsui-like bounds for any r' consecutive rounds within those r rounds. This means that if we got a lower bound $b_{r'}$ for r' rounds, we can add the constraints

$$\sum_{j=1}^{r'} o_{i+j} \leq b_{r'} \quad \text{for } i \in \{1, \ldots, r - r'\}$$

where o_i is the quantity we are optimizing on for one round (i.e. for the heuristic, the number of not-all-equal bits in one round, and for the probability, the overall probability within one round).

This means that we first fix k, then iterate on the number of rounds so that we are able to exploit earlier results for the same k.

Unfortunately, due to complexity of the key schedule (and the increased state size), we were able to search for trails only for Speck32/32 for up to 6 rounds and for every $k \in \{1, \ldots, 8\}$, as well as for Speck48/48 for up to 6 rounds for every $k \in \{1, \ldots, 9\}$. Note that these two parameter sets were not listed in the [BSS+15] paper due to small key size, but they are defined by the general structure (e.g., 16-bit words with a 2-word key). Due to the time limitations, we were not able to have results on Speck48/48 for $k \in \{10, 11, 12\}$ nor for bigger variants of Speck. The probability for the best trail found for each case is given in Table 19 through 28 in Appendix E and F.

5.4 Observations on the results

Overall, our experiments showcase several interesting behaviors. We have clear evidence that the choice of the rotation k to build the RX-differences is not as clear cut as it seemed to be in previous works. Indeed, we have several examples where $k > 1$ gives better results, and there doesn't seem to be a clear rule on the evolution of the best trails when k changes. Moreover, the heuristic of only minimizing the number of not-all-equal bits seemed surprisingly efficient. In a lot of cases, the probability optimization did not go much further than the best trail that the heuristic was able to find, although we did make the choice of limiting the total time for each experiment so that we would be able to have results for every value of k. Thus using this heuristic as a simpler, relatively quick way to get a good trail for a given cipher seems to be a good idea.

5.5 Comparison with previous trail search techniques and distinguishers

Overall, our modeling is not always fast performing. Due to the main focus of the paper, its advantages are correctness/precision and applicability to rotation amounts $k \geq 2$. Application of advanced modeling strategies (e.g., ones similar to [LWRA17]) and alternative modeling frameworks (SMT/SAT/CP) is out of scope of this work. Nonetheless, we provide a brief comparison with previous work.

Remark 6. We remind that, as shown in Section 3, existing search models even for the rotation $k = 1$ can not claim optimality since both expressions used inside models and expressions used to compute the probability of the trail are not correct for a class of transitions.

Alzette We implemented Alzette in the CASCADA tool [RR22], which implements the technique from [LWRA17]. The tool does not support parallelized search so it was run on a single core of an i7-1185G7 3.00GHz CPU. It managed to find an "optimal" trail (with respect to the previous formula and a precision level) for all eight 4-round Alzette instances. However, the performance varied extremely depending on the constant and the weight of the best trail. Comparison with our results for $k = 1$ and best over all k is given in Table 4.

Speck As mentioned in Subsection 5.3, we only managed to search for related-key trails in reduced versions of Speck (32/32 and 48/48) and only for a small number of rounds (up to 6). Furthermore, in these cases, the best found trails were with rotation amount $k = 1$. Thus, we can not directly compare our results to those of [LWRA17], who managed to search for related-key RX-trails in versions ranging from Speck-32/64 (12 rounds) up to

Table 4: Comparison of the trail search using the CASCADA tool [RR22] (modeling technique from [LWRA17]), [HXW22] and our work, for all $\mathsf{Alzette}_{c_i}$ instances. Optimals (as claimed by the respective tool) are in bold. "wt" (weight) stands for negative base-2 logarithm of the probability. † The first tool/technique does not use the exact formula and therefore its actual optimality is not guaranteed. Times for our results reported are based on time when the solution was found, per single k (full search was limited to 5 hours for $i = 0$ and 3 hours for $1 \leq i \leq 7$).

	[RR22, LWRA17] † ($k = 1$)		This work ($k = 1$)		This work ($k > 1$)		[HXW22] ($k = 1$)	[HXW22] ($k > 1$)
c_i	wt	time	wt	time	wt	time	wt	wt
c_0	**33.66**	14 min	33.66	9 min	33.93	1.2 hr	37.66	43.00
c_1	**31.66**	15 min	31.66	80 min	33.01	16 min	38.66	-
c_2	**37.66**	17 hr	37.66	1 hr	34.00	12 min	52.66	-
c_3	**38.66**	2.1 hr	38.66	7 min	32.75	1.8 hr	45.66	-
c_4	**35.66**	18 min	35.66	5 min	33.00	37 min	45.66	-
c_5	**32.66**	9 min	33.66	40 min	30.89	13 min	44.66	-
c_6	**30.66**	2 min	30.66	10 min	32.97	60 min	40.66	-
c_7	**37.66**	4.7 hr	37.66	46 min	32.45	5 s	49.66	-

Speck-128/256 (13 rounds). Adaptation of their search strategies and SMT frameworks to our formula is left as an open question.

6 Malzette: A Malicious Alzette-based Permutation

In this section, we illustrate the importance of RX-differential analysis with rotation $k > 1$ by designing a 12-round Alzette-based permutation with maliciously chosen *dense* constants, such that there exists a strong RX-differential trail allowing practical distinguisher for the permutation, whereas standard arguments against differential/linear cryptanalysis are still reassuring.

6.1 Design procedure

We fix $k = 3$ to showcase our probability formula: lower k leads to lower probabilities, while odd k leads to better "obfuscation" of constants (see below). First, we generate a list of high-probability RX-differentials through modular addition with $d = 0$ in the 3-bit parts $\alpha_{R'}, \beta_{R'}, \Delta_{R'}$ of the differences and $d \in \{0, 1\}$ in the 29-bit parts $\alpha_{L'}, \beta_{L'}, \Delta_{L'}$ of the differences. We keep only those differentials that have reasonably balanced numbers of 0s and 1s, e.g. here is an RX-transition (α, β, Δ) with $d = 1$ in the 29-bit part with probability $2^{-2.83}$:

$$\alpha = \texttt{00000000000000011111111111111100}$$
$$\beta = \texttt{10000000000000011111111111111000}$$
$$\Delta = \texttt{10000000000000111111111111111100}$$

The second step is to find a chain of such differentials with high total probability and random-looking intermediate constants. While standard Alzette only XORs a constant on the left branch, this is insufficient for our purposes. Indeed, there is no linear diffusion on the left branch of Alzette, which essentially would force us to use the same transition in every round. Instead, we extend the constant addition to the both branches, and change the constant after every round.

Let (α, β, Δ) and $(\alpha', \beta', \Delta')$ be RX-differentials transitions through modular addition in rounds $i-1$ and i of Alzette respectively. By propagating them through the linear layer, we obtain conditions on the constants (c_L, c_R) added between the two rounds:

$$c_L \oplus \overleftarrow{c_L} = \Delta \oplus \alpha'$$
$$c_R \oplus \overleftarrow{c_R} = (\beta' \ggg r_i) \oplus (\beta \ggg r_{i-1}) \oplus (\Delta \ggg s_{i-1})$$

Note that the linear map $c \mapsto c \oplus \overleftarrow{c}$ is 2-to-1 whenever $\gcd(k, n) = 1$, i.e., k is odd. Therefore, we have high chances to find preimages c_L, c_R satisfying the above equations. Furthermore, this inversion helps with obfuscation of constants, since the vectors $c_L \oplus \overleftarrow{c_L}$ and $c_R \oplus \overleftarrow{c_R}$ in our case typically consist of a few runs of ones and zeroes, the property which is destroyed by the inversion of $c \mapsto c \oplus \overleftarrow{c}$.

Using these tools, we can find high probability RX-trails using randomized recursive search. In addition, we pose constraints on the constants, such as not having a runs of 3 or more digits in the hexadecimal representations, not containing the zero hexadecimal digit, and not repeating the constants across rounds.

6.2 Resulting Malicious Permutations

We propose two instances of permutations, $\mathsf{Malzette}_1$ and $\mathsf{Malzette}_2$, based on 3-wise iteration of Alzette (with its usual rotation amounts) with full-state malicious constants XORed after every ARX round. The designers of Alzette claim that such 12-round Alzette has no differential trails of probability 2^{-54} or better, which is in fact only a lower bound: the exact optimal trail probability is not known. For the correlation of linear trails, the lower bound is known only for 8 rounds, and which is 2^{-17} (yielding 2^{-19} or even lower for 12 rounds, lower bounding a distinguisher complexity by 2^{38}). It is reasonable to expect the actual optimal trails to have very low probability/correlation, since their weights typically grow very fast with the number of rounds. The difficulty of finding such trails adds to the hardness of mounting a standard attack on such a permutation. On the other hand, our instances have RX-differential trails over full 12 rounds with probabilities $2^{-24.86}$ and $2^{-29.41}$ respectively. The former has sparse and structured constants, while the latter has sufficiently random-looking dense constants, both sets given in Table 5. Note that the last round's constants do not affect the security and were chosen completely at random. Of course, the constants reveal their structure after applying the map $c \mapsto c \oplus (c \lll 3)$, namely, the long runs of ones and zeroes, which makes such a trapdoor detectable by a specific test.

We now provide full RX-differential trails and matching inputs. Denote $\mathsf{Malzette}_i$, $i \in \{1, 2\}$ by $F_i : (\mathbb{F}_2^{32})^2 \to (\mathbb{F}_2^{32})^2$ and set the rotation amount $k = 3$. Then, there exists an RX-differential trail $(\delta_L, \delta_R) \to (\Delta_L, \Delta_R)$ through F_i, satisfied by an input (x_L, x_R). That is,

$$(y_L, y_R) = F_i(x_L, x_R)$$
$$(\overleftarrow{y_L} \oplus \Delta_L, \overleftarrow{y_R} \oplus \Delta_R) = F_i(\overleftarrow{x_L} \oplus \delta_L, \overleftarrow{x_R} \oplus \delta_R)$$

For $\mathsf{Malzette}_1$, the differential and the satisfying input are given by

$$(\delta_L, \delta_R) = \texttt{7ffffff8} : \texttt{3ffffffe}$$
$$(\Delta_L, \Delta_R) = \texttt{b989d417} : \texttt{0347a2a9}$$
$$(x_L, x_R) = \texttt{618e890e} : \texttt{37326dd5}$$

For $\mathsf{Malzette}_2$, the differential and the satisfying input is given by

$$(\delta_L, \delta_R) = \texttt{7fff0003} : \texttt{ffffc001}$$
$$(\Delta_L, \Delta_R) = \texttt{dd2bc54b} : \texttt{078b106c}$$
$$(x_L, x_R) = \texttt{3759c889} : \texttt{0bd01faa}$$

Table 5: Round constants (in hexadecimal) used in malicious Malzette instances and the associated trapdoor RX-trail probabilities.

Round	Malzette_1 Constants	$\log_2$(prob)	Malzette_2 Constants	$\log_2$(prob)
1	00000000:4e381c1c	-2.19	1c71c924:249cad47	-2.83
2	2aaaaaaa:36dbe492	-2.19	49249c71:1249871c	-1.83
3	7fffffff:1236db6c	-1.83	6db6c71c:5b127ffe	-3.19
4	55555555:0763638e	-1.83	38e39249:152ad249	-1.83
5	2aaaaaaa:1b6d4949	-2.19	638e36db:649cad55	-2.83
6	55555555:638ef1c7	-1.83	1c71c7ff:471c9492	-1.83
7	00000000:47638e39	-2.19	36db6d55:63f1c71d	-2.83
8	2aaaaaaa:5236b6db	-2.19	471c7249:36a4ff1c	-2.19
9	55555555:4e381c1c	-1.83	4924938e:5b6c8e47	-3.19
10	7fffffff:638eb1c7	-2.19	2aab6db6:71c736db	-1.83
11	7fffffff:47638e39	-2.19	6db638e3:55b9c71d	-2.83
12	3f2bb31e:b6c004cc	-2.19	fb3d2330:b6da4b61	-2.19
Total		-24.86		-29.41

More satisfying inputs can be found easily. The full trails are given in Table 6.

Table 6: RX-differential ($k = 3$) trails for malicious Malzette instances. (α, β, δ) denote the input and output RX-differences of the modular addition in the respective round. The corresponding full 12-round RX-differentials are 7ffffff8:3ffffffe → b989d417:0347a2a9 and 7fff0003:ffffc001 → dd2bc54b:078b106c respectively.

Round	Malzette_1 (α, β, δ)	$\log_2$(pr.)	Malzette_2 (α, β, δ)	$\log_2$(pr.)
1	7ffffff8:7ffffffc:fffffffc	-2.19	7fff0003:ffff8003:7fff8007	-2.83
2	fffffffc:7ffffffc:7ffffff8	-2.19	80000003:00000003:80000007	-1.83
3	00000003:80000003:80000007	-1.83	80007ffc:80007ffc:0000fff8	-3.19
4	00000003:80000003:80000007	-1.83	00000007:80000003:80000003	-1.83
5	7ffffff8:7ffffffc:fffffffc	-2.19	7fff0003:7fff8003:ffff8007	-2.83
6	80000007:80000003:00000003	-1.83	80000007:00000003:80000003	-1.83
7	fffffffc:7ffffffc:7ffffff8	-2.19	7ffff804:7ffff000:fffff804	-2.83
8	7ffffff8:7ffffffc:fffffffc	-2.19	7ffffff8:7ffffffc:fffffffc	-2.19
9	80000007:80000003:00000003	-1.83	80001fff:00000ffb:80000ffb	-3.19
10	fffffffc:7ffffffc:7ffffff8	-2.19	80000007:80000003:00000003	-1.83
11	fffffffc:7ffffffc:7ffffff8	-2.19	7ff00004:fff80000:7ff80004	-2.83
12	fffffffc:7ffffffc:7ffffff8	-2.19	7fffffc:7ffffff8:fffffffc	-2.19
Total		-24.86		-29.41

Experimental verification We have also verified the distinguishers experimentally and provide the valid pairs of inputs. In the experimental verification, on encryption of 2^{40} random pairs (per each of the two versions), we hit 33 522 and 854 correct pairs, which correspond to probabilities $2^{-24.97}$ and $2^{-30.26}$ respectively. This supports our trail analysis, although the second variant seems to have slightly lower probability (by less than a factor of 2), which must have happened due to dependency of rounds. Finally, we remark that all of these differential matches followed the full trails precisely (whenever input/output differences were satisfied). In other words, we did not observe any clustering effect at all. The probable explanation is the special structure of the trapdoored trail, which does not

allow deviations.

Remark 7. Although we described a permutation, it is easy to convert the design into a block cipher and the permutation distinguishers into a related-key RX-differential distinguisher, and add some rounds for a key recovery. This should suffice as a proof-of-concept of importance of RX-differential analysis.

Acknowledgments

The work was supported by the Luxembourg National Research Fund's (FNR) and the German Research Foundation's (DFG) joint project APLICA (C19/IS/13641232). The authors thank the anonymous reviewers for the fruitful discussions and suggestions. The authors also thank Marius Lombard-Platet for the idea of Proposition 10.

References

[AAE+14] Ange Albertini, Jean-Philippe Aumasson, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Malicious hashing: Eve's variant of SHA-1. In *Selected Areas in Cryptography*, volume 8781 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2014. 544

[AL16] Tomer Ashur and Yunwen Liu. Rotational cryptanalysis in the presence of constants. *IACR Trans. Symmetric Cryptol.*, 2016(1):57–70, 2016. 542, 543, 545, 558, 559, 560, 561

[ARS+20] Seyyed Arash Azimi, Adrián Ranea, Mahmoud Salmasizadeh, Javad Mohajeri, Mohammad Reza Aref, and Vincent Rijmen. A bit-vector differential model for the modular addition by a constant. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 385–414. Springer, 2020. 555, 556

[ARS+22] Seyyed Arash Azimi, Adrián Ranea, Mahmoud Salmasizadeh, Javad Mohajeri, Mohammad Reza Aref, and Vincent Rijmen. A bit-vector differential model for the modular addition by a constant and its applications to differential and impossible-differential cryptanalysis. *Des. Codes Cryptogr.*, 90(8):1797–1855, 2022. 555, 556

[AY15] Riham AlTawy and Amr M. Youssef. Watch your constants: malicious Streebog. *IET Inf. Secur.*, 9(6):328–333, 2015. 544

[BBdS+20a] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Alzette: A 64-bit ARX-box (feat. CRAX and TRAX). In *CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 419–448. Springer, 2020. 543, 544, 566

[BBdS+20b] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Lightweight AEAD and hashing using the Sparkle permutation family. *IACR Trans. Symmetric Cryptol.*, 2020(S1):208–261, 2020. 566

[Ber08a] Daniel J Bernstein. ChaCha, a variant of Salsa20. In *Workshop record of SASC*, volume 8 (1), pages 3–5. Citeseer, 2008. 567

[Ber08b] Daniel J. Bernstein. The Salsa20 family of stream ciphers. In *The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008. 567

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *DAC*, pages 175:1–175:6. ACM, 2015. 568, 569, 587

[CZX+23] Siwei Chen, Mingming Zhu, Zejun Xiang, Runqing Xu, Xiangyong Zeng, and Shasha Zhang. Rotational-XOR differential rectangle cryptanalysis on Simon-like ciphers. In *CT-RSA*, volume 13871 of *Lecture Notes in Computer Science*, pages 305–330. Springer, 2023. 542

[FWG+16] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. MILP-based automatic search algorithms for differential and linear trails for Speck. In *FSE*, volume 9783 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2016. 562, 563

[HLK+13] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Donggeon Lee. LEA: A 128-bit block cipher for fast encryption on common processors. In *WISA*, volume 8267 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2013. 587

[HXW22] Mingjiang Huang, Zhen Xu, and Liming Wang. On the probability and automatic search of rotational-XOR cryptanalysis on ARX ciphers. *Comput. J.*, 65(12):3062–3080, 2022. 543, 558, 566, 567, 570

[KN10] Dmitry Khovratovich and Ivica Nikolic. Rotational cryptanalysis of ARX. In *FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 333–346. Springer, 2010. 542

[KNR10] Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational rebound attacks on reduced Skein. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2010. 543

[LM01] Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2001. 545, 564

[LWRA17] Yunwen Liu, Glenn De Witte, Adrián Ranea, and Tomer Ashur. Rotational-XOR cryptanalysis of reduced-round SPECK. *IACR Trans. Symmetric Cryptol.*, 2017(3):24–36, 2017. 542, 543, 558, 569, 570

[Mac01] Alexis Warner Machado. Differential probability of modular addition with a constant operand. Cryptology ePrint Archive, Paper 2001/052, 2001. 555

[Mor15] Pawel Morawiecki. Malicious Keccak. Cryptology ePrint Archive, Paper 2015/1085, 2015. 544

[MVCP10] Nicky Mouha, Vesselin Velichkov, Christophe De Cannière, and Bart Preneel. The differential analysis of S-functions. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 36–56. Springer, 2010. 556

[NW97] Roger M Needham and David J Wheeler. TEA extensions. *Report (Cambridge University, Cambridge, UK, 1997)*, 1997. 587

[Ran22] Adrián Ranea. Cascada 1.0, April 2022. 543, 558, 567

[RLA17] Adrián Ranea, Yunwen Liu, and Tomer Ashur. An easy to use tool for rotational-XOR cryptanalysis of ARX block ciphers. *Proceedings of the Romanian Academy, Series A*, 18(special issue):307–316, 2017. 558

[RR22] Adrián Ranea and Vincent Rijmen. Characteristic automated search of cryptographic algorithms for distinguishing attacks (CASCADA). *IET Inf. Secur.*, 16(6):470–481, 2022. 567, 569, 570

[Sch13] Ernst Schulte-Geers. On CCZ-equivalence of addition mod 2^n. *Des. Codes Cryptogr.*, 66(1-3):111–127, 2013. 545

[SRB21] Sadegh Sadeghi, Vincent Rijmen, and Nasour Bagheri. Proposing an MILP-based method for the experimental verification of difference-based trails: application to SPECK, SIMECK. *Des. Codes Cryptogr.*, 89(9):2113–2155, 2021. 566

[ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *EUROCRYPT (3)*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, 2017. 587

[Udo21] Aleksei Udovenko. MILP modeling of boolean functions by minimum number of inequalities. Cryptology ePrint Archive, Paper 2021/1099, 2021. 562

[XLSL19] Wenqian Xin, Yunwen Liu, Bing Sun, and Chao Li. Improved cryptanalysis on SipHash. In *CANS*, volume 11829 of *Lecture Notes in Computer Science*, pages 61–79. Springer, 2019. 558

A Specification of the Alzette permutations

The permutation for Alzette is built as four consecutive rounds, where the round function $f_c(x, y, r, s)$ is defined as the procedure

$$\begin{aligned} x &\leftarrow x \boxplus (y \ggg r) \\ y &\leftarrow y \oplus (x \ggg s) \\ x &\leftarrow x \oplus c \end{aligned}$$

The rotation offsets r and s differ between each round, but the same constant c is used in all four. In the end, for a given constant c, the permutation of $(x, y) \in \mathbb{F}_2^{2n}$ is defined by the procedure

$$\begin{aligned} &f_c(x, y, 31, 24) \\ &f_c(x, y, 17, 17) \\ &f_c(x, y, 0, 31) \\ &f_c(x, y, 24, 16) \end{aligned}$$

The resulting permutation when using constant c is denoted $\mathsf{Alzette}_c$. The following eight randomly chosen constants are provided by the designers:

$$\begin{aligned} (c_0, \ldots, c_7) = (&\texttt{0xb7e15162}, \texttt{0xbf715880}, \texttt{0x38b4da56}, \texttt{0x324e7738}, \\ &\texttt{0xbb1185eb}, \texttt{0x4f7c7b57}, \texttt{0xcfbfa1c8}, \texttt{0xc2b3293d}). \end{aligned}$$

B Specification of the Salsa permutation

The permutation used in Salsa is built using the "Quarter Round" function $QR(a, b, c, d)$ defined by the procedure

$$\begin{aligned} b &\leftarrow b \oplus ((a \boxplus d) \lll 7) \\ c &\leftarrow c \oplus ((b \boxplus a) \lll 9) \\ d &\leftarrow d \oplus ((c \boxplus b) \lll 13) \\ a &\leftarrow a \oplus ((d \boxplus c) \lll 18) \end{aligned}$$

This function is then applied to each column (on odd rounds) or each row (on even rounds) depending on the round index.

C Results on Salsa

Table 7: Best solution found in 1h (for each k) for Salsa's first QR. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**23**	**23**	35	24	**29.6601**	**29.6601**
2	**28**	**28**	34.7123	28	**34.7123**	**34.7123**
3	**27**	**27**	35.8528	27	**34.3382**	**34.3382**
4	**22**	**22**	30.0177	22	**29.5031**	**29.5031**
5	**22**	**22**	29.9112	22	**29.5714**	**29.5714**
6	**26**	**26**	33.9109	26	**33.9109**	**33.9109**
7	**28**	**28**	35.9832	28	**35.8024**	**35.8024**
8	**16**	**16**	23.9888	16	**23.9663**	**23.9663**
9	**20**	**20**	28	20	**28**	**28**
10	**21**	**21**	29	21	**29**	**29**
11	**26**	**26**	33.993	26	33.9863	33.9873
12	**25**	**25**	32.9877	25	**32.9657**	**32.9657**
13	**26**	**26**	34.0004	26	**33.9717**	**33.9717**
14	**22**	**22**	29.9999	22	**29.9957**	**29.9957**
15	**27**	**27**	35	27	34	34.9943
16	**24**	**24**	32.0013	24	31.9925	31.9943

Table 8: Best solution found in 1h (for each k) for Salsa's second QR. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	21	5	26.6601	21	20.8261	26.6601
2	18	5	24.7123	18	16.743	24.4493
3	17	5	24.8528	17	20.7964	24.4902
4	13	5	20.5727	13	**19.9877**	**19.9877**
5	18	6	26.093	18	20.8095	26.093
6	17	5	24.9105	17	19.5046	24.9105
7	14	5	21.944	14	**21.944**	**21.944**
8	17	5	24.972	17	20.8401	24.8845
9	15	5	22.9916	15	19.9761	22.9608
10	19	6	26.9887	19	21.4414	26.9083
11	14	6	21.9979	14	21.415	21.9965
12	19	5	27	19	22.4012	27
13	15	5	23.0007	15	21.5869	23.0005
14	18	5	25.9988	18	20.3224	25.9988
15	16	5	24.0001	16	19.7989	24
16	20	6	28	20	21.6438	27.9999

Table 9: Best solution found in 1h (for each k) for Salsa's third QR. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	21	5	28.2451	22	20.41	27.6602
2	17	5	26.6601	17	19.6781	25.9232
3	20	5	27.6601	20	21.2449	26.6013
4	17	5	25.5715	17	19.9332	24.4852
5	22	5	29.7807	22	20.9115	29.7413
6	17	5	24.9105	17	21.0579	24.8665
7	17	5	24.9888	17	21.9264	24.9888
8	17	5	24.9775	17	20.7091	24.9775
9	18	5	26.0113	18	21.6674	25.9747
10	20	5	28	20	20.6914	28
11	17	5	24.9986	16	19.6939	24
12	18	6	25.9982	18	20.0798	25.9982
13	**17**	**17**	25.0002	17	17.1869	24.9996
14	24	5	31.9999	21	20.6885	29
15	16	5	24.0003	16	20.0425	24
16	20	6	27.9989	20	20.2327	27.9971

Table 10: Best solution found in 1h (for each k) for Salsa's fourth QR. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**23**	**23**	28.6601	23	**28.6601**	**28.6601**
2	**23**	**23**	30.7712	23	**29.4493**	**29.4493**
3	**24**	**24**	31.3203	24	**31.3203**	**31.3203**
4	**23**	**23**	31.5715	23	**31.4784**	**31.4784**
5	**23**	**23**	31.2843	23	**31.0458**	**31.0458**
6	**25**	**25**	33.0284	25	**32.9353**	**32.9353**
7	**22**	**22**	30.034	22	**29.9217**	**29.9217**
8	**22**	**22**	30	22	**29.9608**	**29.9608**
9	**24**	**24**	31.9831	24	**31.958**	**31.958**
10	**25**	**25**	33.024	25	**32.9985**	**32.9985**
11	**23**	**23**	31	23	**30.9643**	**30.9643**
12	**25**	**25**	33	25	**32.9646**	**32.9646**
13	**20**	**20**	28.0014	20	**28**	**28**
14	**21**	**21**	29.0004	21	28.9993	29
15	**23**	**23**	30.9986	23	**30.9949**	**30.9949**
16	**24**	**24**	31.9999	24	30.6066	31.9943

D Results on Alzette

For completeness, we provide search results for Alzette for all constants proposed by the designers (used in the Sparkle permutations). The structure of Alzette is recalled in Appendix A. As usual, "wt" refers to the negated base-2 logarithm of the probability. For the constant c_0, the time limits were set to 1 hour (NEQ heuristic) and 4 hours (exact probability). For the other constants, the time limits were set respectively to 1 and 2 hours.

Table 11: Best solution found in 5h (for each k) for Alzette with the constant c_0 = `0xb7e15162`. Best probability/NEQ heuristic per column is underlined.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	27	21	35.83	28	32.04	<u>33.66</u>
2	28	22	34.71	28	33.03	34.71
3	31	24	38.85	31	31.08	38.80
4	30	22	37.83	29	32.76	36.74
5	32	23	39.91	32	32.51	39.87
6	29	23	37.07	29	33.02	37.00
7	<u>26</u>	21	<u>33.98</u>	<u>26</u>	31.77	33.93
8	32	26	40.02	32	33.84	40.00
9	29	22	36.99	29	33.13	36.90
10	31	24	39.01	31	34.20	38.98
11	31	25	39.00	31	34.09	39.00
12	28	20	36.00	28	29.98	36.00
13	31	24	38.99	31	32.11	38.99
14	30	21	38.00	30	32.17	38.00
15	29	21	37.00	29	32.67	37.00
16	<u>26</u>	15	34.00	<u>26</u>	33.32	34.00

Table 12: Best solution found in 3h (for each k) for Alzette with the constant c_1 = `0xbf715880`. Best probability/NEQ heuristic per column is underlined.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	<u>26</u>	21	<u>31.66</u>	26	29.47	<u>31.66</u>
2	28	20	34.71	28	31.98	34.71
3	28	25	35.75	28	32.66	35.66
4	27	21	34.18	27	32.73	34.10
5	30	26	37.96	30	31.55	37.87
6	31	20	38.96	31	30.66	38.96
7	30	17	38.03	27	30.14	35.00
8	27	25	34.98	27	32.38	34.96
9	<u>26</u>	18	34.01	<u>25</u>	30.29	33.01
10	30	22	37.99	30	32.11	37.99
11	34	23	42.00	34	33.28	42.00
12	31	23	39.00	31	33.76	39.00
13	32	21	40.00	28	31.58	36.00
14	36	23	44.00	36	33.81	43.99
15	32	22	40.00	27	32.13	35.00
16	30	13	38.00	30	30.78	38.00

Table 13: Best solution found in 3h (for each k) for Alzette with the constant c_2 = `0x38b4da56`. Best probability/NEQ heuristic per column is underlined.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	32	20	39.25	32	32.44	37.66
2	31	23	38.45	31	31.37	38.45
3	28	19	35.32	28	29.65	35.32
4	27	24	35.02	27	31.51	35.02
5	28	21	35.87	28	31.66	35.87
6	28	19	35.91	28	30.26	35.79
7	32	22	39.90	31	31.06	38.81
8	30	21	37.95	30	30.97	37.95
9	<u>26</u>	18	<u>34.00</u>	<u>26</u>	29.63	<u>34.00</u>
10	31	21	38.97	31	31.03	38.97
11	33	22	41.00	32	32.05	40.00
12	31	21	39.00	29	30.64	37.00
13	27	21	35.00	27	28.19	35.00
14	28	20	36.00	28	30.57	36.00
15	29	19	37.00	29	31.50	37.00
16	<u>26</u>	24	<u>34.00</u>	<u>26</u>	32.16	<u>34.00</u>

Table 14: Best solution found in 3h (for each k) for Alzette with the constant c_3 = `0x324e7738`. Best probability/NEQ heuristic per column is underlined.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	33	25	38.66	33	33.16	38.66
2	31	23	37.71	31	28.77	37.71
3	<u>26</u>	19	<u>33.34</u>	<u>26</u>	30.63	<u>32.75</u>
4	29	24	37.01	29	31.46	36.68
5	29	16	37.19	27	28.03	34.33
6	31	21	38.89	31	31.79	38.89
7	34	22	41.98	31	32.48	38.90
8	29	19	36.99	27	30.89	34.99
9	28	21	35.97	28	30.43	35.97
10	28	22	36.00	28	31.50	36.00
11	32	22	40.00	32	30.40	40.00
12	31	20	39.00	27	29.87	35.00
13	31	21	39.00	28	32.79	36.00
14	28	18	36.00	28	31.01	36.00
15	29	19	37.00	29	30.44	37.00
16	33	26	41.00	33	35.01	41.00

Table 15: Best solution found in 3h (for each k) for Alzette with the constant c_4 = `0xbb1185eb`. Best probability/NEQ heuristic per column is underlined. † indicates that no plaintext pair was found for the resulting trail.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	29	25	37.83	30	33.36	35.66
2	34	21	42.19	34	30.63	41.45
3	33	25	40.34	33	34.38	40.34
4	29	21	36.92	29	30.64	36.60
5	27	20	34.91	27	30.16	34.91
6	28	18	35.85	28	29.72	35.85
7	28	22	35.97	28	33.00	35.97
8	31	24	38.99	31	33.32	38.96
9	30	23	38.00	29	32.33	36.99
10	25	23	33.00	25	31.55	33.00
11	32	21	40.00	29	31.14	37.00
12	32	24	40.00	32	33.04	40.00
13	33	22	41.00	33	31.16	40.99
14	36	25	44.00	36	34.31	44.00
15	30	20	38.00	29	31.25	37.00
16	33	14	41.00	31	31.60	39.00†

Table 16: Best solution found in 3h (for each k) for Alzette with the constant c_5 = `0x4f7c7b57`. Best probability/NEQ heuristic per column is underlined. † indicates that no plaintext pair was found for the resulting trail.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	28	21	35.25	28	30.47	33.66
2	28	22	34.71	28	32.04	34.71
3	30	20	38.08	30	32.05	36.92
4	29	24	36.83	29	33.00	36.66
5	27	25	34.87	27	31.75	34.87
6	30	21	37.80	30	32.63	37.80
7	23	21	31.01	23	30.62	30.89†
8	29	23	36.97	29	32.23	36.92
9	34	25	42.00	31	33.39	39.01
10	32	23	40.01	32	33.04	39.99
11	34	23	42.00	34	33.80	42.00
12	23	17	31.00	23	29.13	31.00
13	28	21	36.00	28	30.97	35.98
14	29	25	37.00	29	32.66	37.00
15	33	22	41.00	31	32.22	39.00
16	33	16	41.00	32	32.59	40.00

Table 17: Best solution found in 3h (for each k) for Alzette with the constant c_6 = `0xcfbfa1c8`. Best probability/NEQ heuristic per column is underlined.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	<u>25</u>	23	<u>30.66</u>	<u>25</u>	30.50	<u>30.66</u>
2	29	21	35.71	29	30.23	35.71
3	30	26	38.02	29	33.08	36.19
4	31	22	38.94	31	32.77	38.57
5	28	27	35.87	28	32.52	35.74
6	27	20	34.98	27	30.86	34.79
7	29	20	37.06	29	30.47	37.01
8	<u>25</u>	19	32.99	<u>25</u>	30.85	32.97
9	28	20	35.99	28	29.40	35.98
10	28	21	36.00	28	31.46	35.99
11	34	23	42.00	34	32.94	41.99
12	36	26	44.00	36	34.08	44.00
13	28	19	36.00	28	30.72	36.00
14	34	21	42.00	31	33.12	39.00
15	27	20	35.00	27	30.85	35.00
16	30	22	38.00	30	29.23	38.00

Table 18: Best solution found in 3h (for each k) for Alzette with the constant c_7 = `0xc2b3293d`. Best probability/NEQ heuristic per column is underlined.

	NEQ optimization			Probability optimization		
k	nb NEQ	best LB	wt	nb NEQ	best LB	wt
1	31	24	39.83	32	31.48	37.66
2	<u>25</u>	20	<u>32.45</u>	<u>25</u>	29.89	<u>32.45</u>
3	30	23	36.60	30	33.95	36.60
4	30	21	38.20	30	33.19	37.66
5	31	23	38.70	31	32.26	38.62
6	34	23	42.00	34	32.61	41.91
7	34	20	41.99	29	31.77	37.01
8	28	18	35.98	28	29.87	35.98
9	29	23	36.99	29	33.37	36.98
10	26	19	33.99	26	29.12	33.97
11	32	22	40.00	31	30.37	38.99
12	<u>25</u>	20	33.00	<u>25</u>	28.95	33.00
13	32	22	40.00	32	32.02	39.99
14	27	21	35.00	27	32.37	35.00
15	30	23	38.00	30	31.99	38.00
16	32	16	40.00	32	30.93	40.00

E Results on Speck32/32

Since large state and complex key schedule slow down MILP model optimization significantly, for illustration purposes, we applied the 2-word key schedule of Speck (32-bit key) to the 32-bit version (Speck specification does not use it for 32/48/64-bit block sizes because of the small key size).

Table 19: Best solution found in 1h (for each k) for Speck32/32 over 2 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**0**	**0**	4.245	0	**4.24458**	**4.24458**
2	**0**	**0**	5.0339	0	**5.03365**	**5.03365**
3	**0**	**0**	5.4897	0	**5.4897**	**5.4897**
4	**0**	**0**	5.7366	0	**5.73655**	**5.73655**
5	**0**	**0**	5.8647	0	**5.8647**	**5.8647**
6	**0**	**0**	5.9287	0	**5.92867**	**5.92867**
7	**0**	**0**	5.9579	0	**5.95787**	**5.95787**
8	**0**	**0**	5.9662	0	**5.96625**	**5.96625**

Table 20: Best solution found in 1h (for each k) for Speck32/32 over 3 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**0**	**0**	7.0751	0	**7.07497**	**7.07497**
2	**0**	**0**	8.3899	0	**8.38992**	**8.38992**
3	**0**	**0**	9.1495	0	**9.14949**	**9.14949**
4	**0**	**0**	9.5609	0	**9.56092**	**9.56092**
5	**0**	**0**	9.7745	0	**9.77451**	**9.77451**
6	**0**	**0**	9.8811	0	**9.88112**	**9.88112**
7	**0**	**0**	9.9298	0	**9.92979**	**9.92979**
8	**0**	**0**	9.9437	0	**9.94375**	**9.94375**

Table 21: Best solution found in 1h (for each k) for Speck32/32 over 4 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**3**	**3**	17.4898	3	**14.9046**	**14.9046**
2	**5**	**5**	21.8035	5	**21.8035**	**21.8035**
3	**6**	**6**	24.2396	6	**23.8246**	**23.8246**
4	**6**	**6**	25.3798	6	**25.3797**	**25.3797**
5	**6**	**6**	25.6646	6	**25.6646**	**25.6646**
6	**3**	**3**	19.9269	3	**19.879**	**19.879**
7	**5**	**5**	22.9498	5	**22.8767**	**22.8767**
8	**4**	**4**	21.8988	4	**21.8988**	**21.8988**

Table 22: Best solution found in 1h (for each k) for Speck32/32 over 5 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	10	5	32.3197	10	**32.3183**	**32.3183**
2	11	6	35.1609	11	**35.1577**	**35.1577**
3	17	7	49.6868	16	36.0789	45.6151
4	16	7	47.4163	16	39.3567	47.053
5	14	6	43.8708	13	**42.6489**	**42.6489**
6	13	6	42.7863	11	**38.7913**	**38.7913**
7	12	6	41.9813	13	**39.7992**	**39.7992**
8	10	5	35.8431	10	**35.8431**	**35.8431**

Table 23: Best solution found in 1h (for each k) for Speck32/32 over 6 rounds. Optimals in bold. † indicates that no plaintext pair was found for the resulting trail.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	16	5	47.1505	16	37.7968	46.148
2	22	6	62.8049	21	40.1124	61.3688†
3	37	8	91.0227	26	38.4284	71.2579†
4	26	7	71.9424	26	39.9844	69.4866†
5	29	8	78.1388	24	39.7644	66.7442†
6	20	6	58.8804	20	38.9773	57.4848†
7	16	7	52.8763	16	38.0515	52.7888†
8	18	6	53.8655	18	34.887	53.8655

F Results on Speck48/48

Table 24: Best solution found in 1h (for each k) for Speck48/48 over 2 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**0**	**0**	4.2451	0	**4.24508**	**4.24508**
2	**0**	**0**	5.0342	0	**5.03421**	**5.03421**
3	**0**	**0**	5.4902	0	**5.49022**	**5.49022**
4	**0**	**0**	5.7376	0	**5.7376**	**5.7376**
5	**0**	**0**	5.8668	0	**5.8668**	**5.8668**
6	**0**	**0**	5.9329	0	**5.93288**	**5.93288**
7	**0**	**0**	5.9663	0	**5.96629**	**5.96629**
8	**0**	**0**	5.9831	0	**5.98306**	**5.98306**
9	**0**	**0**	5.9914	0	**5.99142**	**5.99142**

Table 25: Best solution found in 1h (for each k) for Speck48/48 over 3 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**0**	**0**	7.0752	0	**7.07503**	**7.07503**
2	**0**	**0**	8.3903	0	**8.39036**	**8.39036**
3	**0**	**0**	9.1504	0	**9.15037**	**9.15037**
4	**0**	**0**	9.5627	0	**9.56267**	**9.56267**
5	**0**	**0**	9.778	0	**9.77802**	**9.77802**
6	**0**	**0**	9.8881	0	**9.88813**	**9.88813**
7	**0**	**0**	9.9438	0	**9.94381**	**9.94381**
8	**0**	**0**	9.9718	0	**9.97177**	**9.97177**
9	**0**	**0**	9.9857	0	**9.9857**	**9.9857**

Table 26: Best solution found in 1h (for each k) for Speck48/48 over 4 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	**3**	**3**	17.4902	4	**16.9053**	**16.9053**
2	**6**	**6**	23.8054	6	**22.8054**	**22.8054**
3	**5**	**5**	22.8284	5	**22.8284**	**22.8284**
4	**6**	**6**	25.3927	6	**24.3927**	**24.3927**
5	**6**	**6**	25.7807	6	**25.6905**	**25.6905**
6	**5**	**5**	23.8436	5	**23.8436**	**23.8436**
7	**4**	**4**	21.9443	4	**21.9217**	**21.9217**
8	**5**	**5**	22.9718	5	**22.9604**	**22.9604**
9	**5**	**5**	23.9741	5	**23.9741**	**23.9741**

Table 27: Best solution found in 1h (for each k) for Speck48/48 over 5 rounds. Optimals in bold.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	16	3	46.9052	12	32.031	37.3203
2	12	6	38.8986	12	29.2115	38.8985
3	46	5	109.2153	18	27.935	48.8284
4	39	6	95.7376	17	31.3449	49.2226
5	45	6	98.0041	20	31.0771	52.4737
6	27	5	71.0231	16	30.903	46.8914
7	31	4	75.1617	17	34.8509	48.8342
8	18	5	52.005	15	32.4578	45.9715
9	17	5	50.9659	17	30.3652	49.954

Table 28: Best solution found in 1h (for each k) for Speck48/48 over 6 rounds. Optimals in bold. † indicates that no plaintext pair was found for the resulting trail.

	NEQ optimization			Probability optimization		
k	nb NEQ	bestLB	-log2(prob)	nb NEQ	bestLB	-log2(prob)
1	50	4	117.9052	20	31.9818	57.7352
2	58	7	134.0162	30	36.3996	71.3134
3	54	6	125.1504	44	34.2616	101.091†
4	56	7	133.6069	38	36.0866	92.5626†
5	55	7	128.8478	36	36.5846	88.7425
6	63	6	147.9776	34	35.9255	87.1285
7	50	5	113.0005	29	34.6834	76.8324
8	45	6	104	28	40.7458	73.9549
9	55	6	131.9971	41	34.9591	94.9315†

G Impossible RX-Differentials

In this section, we explore briefly the possibility of using our results (namely, Theorem 1) for searching for *impossible* RX-differentials. Due to low effectiveness of this approach, it was not included in the main body of the paper.

G.1 Search for Single-Key Impossible RX-Differentials

Unlike regular differentials, in RX-differential trails the key addition has an impact on the trail. Indeed, for an input RX-difference (δ, k), the RX-difference after the addition (with an XOR) of some round key $k \in \mathbb{F}_2^n$ will be $(\delta \oplus k \oplus \overleftarrow{k}, k)$. When looking for RX-differential trails, this often results in only a certain subset of keys to allow for a specific trail, thus resulting in a weak-key model. However, if we were able to find some RX-differentials that are impossible for *any* key, then this would make RX-differentials useful in the context of the single-key model.

We used MILP to model and search for impossible RX-differentials for any key, and thus we also have to add constraints to model the key-schedule *in value*. These constraints are very similar to the ones used to model the propagation of RX-differentials, and one just has to be careful that the key addition must be done according to the propagation rule given in the previous paragraph (see Proposition 15). Then, we search for impossible RX-differentials where the weight of the input and output RX-difference is 1 (i.e. n^2 differences). The main reason for this is that to exploit impossible differentials in an attack, one usually needs a *truncated* impossible differential, i.e. a subspace of differences that cannot propagate to another subspace of differences through the cipher. Thus, weight 1 differences are the minimum requirement to have any chance at finding such an impossible truncated differential, hence our focus on those. This is inspired by a similar approach from [ST17]. Note that there is a minimal number of rounds required for this idea to work. Indeed, assume that the master key is of size m and that each round key is of size n, then we need to consider at least r rounds such that $rn > m$, otherwise there are not enough constraints on the key schedule and round keys.

We applied this search to the ciphers `LEA` [HLK+13], `Speck` [BSS+15] and `XTEA` [NW97], most of them having several variants in key size and/or block size. Unfortunately, for all of them, we were never able to find such impossible RX-differentials, even when only considering input/output differences of weight 1, and the smallest amount of rounds possible (e.g. only 3 rounds on `Speck2n/2n`). Thus for any input (resp. output) difference Δ (resp. ∇), both of weight 1, we always found at least one key such that there exists an RX-difference trail from Δ to ∇.

G.2 Search for Related-Key Impossible RX-Differentials

Since our search in the single-key model was not successful, we shifted out focus to look for *related-key* impossible RX-differentials. In this model, the attacker can query an encryption oracle not only using the secret key K, but also using another key K' that is related to K in some way (obviously without knowing either K or K', but still knowing the relation between the two). In the context of RX-differentials, this means that the attacker can also make queries using a key $K' = \overleftarrow{K} \oplus \Delta^K$ for some Δ^K that the attacker can choose. This gives a lot more power to the attacker, but also expands the search space significantly.

We searched for such impossible RX-differentials in a related-key model on the `Speck32/32` and `Speck64/64` block ciphers, with block and key size 32 and 64 bits respectively. To make the search finish in a reasonable time, we only focused on RX-differences in the key with weight 1, i.e. $w(\Delta^K) = 1$.

For `Speck32/32`, again even with the smallest amount of rounds possible, no impossible RX-differentials were found for any of the possible Δ^K such that $w(\Delta^K) = 1$. For

`Speck64/64`, we found no impossible RX-differentials on 4 rounds with the same constraint on Δ^K , however on 3 rounds we got several of them. The list is too long to write down in this paper, but to give an idea, the case where we found the highest amount of impossible differentials was with $k = 1$, with $\Delta^K_{39} = 1$ (and other bits at 0). In this case, we found 1112 pairs (Δ, ∇) of weight 1 input/output differentials that were impossible. For $k = 2$, the case with the highest amount of such pairs to be impossible is also with $\Delta^K_{39} = 1$, leading to 920 impossible pairs. We also observed that as k increases, the number of impossible weight 1 RX-differentials goes down, to even 0 (i.e. not weight (1,1) impossible differentials) for $k \geq 13$, see Table 29 in Appendix H.

Moreover, while we could not search for every combination to build an impossible truncated RX-differential, we were still able to find an *almost truncated* impossible differential. This means that the almost the whole set of differentials is impossible.

Indeed, take $k = 1$, with $\Delta^K_{39} = 1$ (and other bits at 0), $\mathcal{S}_{in} = \{32, 33, 35, 36\}$ and $\mathcal{S}_{out} = \{15, 16, 19, 20\}$. Then for any input (resp. output) RX-differential Δ (resp. ∇) such that $\Delta_i = 0$ for $i \notin \mathcal{S}_{in}$ (resp. $\nabla_i = 0$ for $i \notin \mathcal{S}_{out}$), (Δ, ∇) is an impossible RX-differential over 3 rounds *except* for the case $(\Delta, \nabla) = (0, 0)$. Note that this last case in non-trivial in the case of RX-differentials, as an RX-difference of 0 does not mean that the plaintexts are equal, but rather that $\overleftarrow{x} = x'$. Unfortunately, due to time constraints, we were not able to explore the search space more for larger/better cases.

G.3 About the Alzette S-box

As mentioned in Section 5.1, Alzette is a 64-bit ARX-based S-box. Similarly to the single-key case (which here would be more akin to a known-key case), we investigated the existence of weight-(1,1) impossible RX-differentials, for each of these constants. For 4 of these constants, namely `0xb7e15162`, `0x38b4da56`, `0xbb1185eb` and `0xc2b3293d`, there is no weight-(1,1) impossible RX-differentials. For the constant `0xbf715880`, there are only 2 weight-(1,1) impossible RX-differentials over all values of k, and these two are obtained with $k = 1$. Similarly, for the constant `0x324e7738`, there are only 3 weight-(1,1) impossible RX-differentials, all obtained with $k = 3$.

Something more interesting happens with the constant `0xcfbfa1c8`, where the only case with impossible RX-differentials is $k = 3$. There are 6 such impossible RX-differentials, and 4 of them lead to an interesting structure. These 4 are built with input difference Δ such that the one bit set to 1 is Δ_{34}, and the output difference is ∇ where the bit set to 1 is $\nabla_i, i \in \{12, 21, 37, 60\}$. We looked at the truncated RX-differential $\{34\} \to \{12, 21, 37, 60\}$, which thus consists of 32 trails[4], and out of them, 8 are impossible, which are the 4 weight-(1,1) mentioned previously, as well as the 4 weight-(1,3) impossible RX-differentials that can be built with this set of indices. While not resulting in a truncated differential per se, it is interesting to observe that all RX-differentials of weight (1,1) and (1,3) are impossible, while all the others are possible. Finally for the last studied constant `0x4f7c7b57`, in addition to 3 other impossible RX-differentials that do not seem to belong to any structure, we found a similar structure as in the previous constant. Here for $k = 7$, if we look at the truncated RX-differential $\{34, 35\} \to \{12, 21, 37, 60\}$, so 64 trails in total, 16 of them are impossible, and those are again exactly all the trails of weight (1,1) and (1,3).

H Results for Impossible RX-differentials on Speck64/64

[4]Recall that the case of a zero RX-difference in non-trivial compared to regular differentials, thus we also need to examine all differentials $0 \to \nabla$

Table 29: Best cases of impossible RX-differentials over 3 rounds of `Speck64/64` for each value of γ. The "# of Δ^K" column indicates how many weight-1 RX-differentials over the key Δ^K lead to at least one impossible RX-difference, the best case column indicates which Δ^K leads to the highest number of weight (1-1) impossible RX-differentials.

k	# of Δ^K	Best Case	k	# of Δ^K	Best Case
1	64	1112 imp.diff. with $\Delta^K_{39}=1$	9	45	145 imp.diff. with Δ^K_2
2	64	920 imp.diff. with $\Delta^K_{39}=1$	10	42	44 imp.diff. with Δ^K_2
3	62	659 imp.diff. with $\Delta^K_1=1$	11	24	24 imp.diff. with Δ^K_2
4	61	592 imp.diff. with $\Delta^K_2=1$	12	4	4 imp.diff. with Δ^K_3
5	58	508 imp.diff. with $\Delta^K_2=1$	13	0	-
6	57	437 imp.diff. with $\Delta^K_2=1$	14	0	-
7	55	333 imp.diff. with $\Delta^K_2=1$	15	0	-
8	51	218 imp.diff. with $\Delta^K_4=1$	16	0	-

I Other MILP Propagation Rules

These are known from the previous works, but are given for completeness.

Proposition 11. *Let $x, y, z \in \mathbb{F}_2$ with $z = x \oplus y$, and $\mathbf{x}, \mathbf{y}, \mathbf{z}$ be binary variables modeling x, y and z respectively. Then we can model the relation $z = x \oplus y$ with the set of inequalities*

$$\mathbf{z} = \mathbf{XOR}(\mathbf{x}, \mathbf{y}) \Leftrightarrow \begin{cases} \mathbf{x} + \mathbf{y} + (1 - \mathbf{z}) \geq 1 \\ \mathbf{x} + (1 - \mathbf{y}) + \mathbf{z} \geq 1 \\ (1 - \mathbf{x}) + \mathbf{y} + \mathbf{z} \geq 1 \\ (1 - \mathbf{x}) + (1 - \mathbf{y}) + (1 - \mathbf{z}) \geq 1 \end{cases}$$

Proposition 12. *Let $w, x, y, z \in \mathbb{F}_2$ with $z = x \oplus y \oplus w$, and $\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z}$ be binary variables modeling w, x, y and z respectively. Then we can model the relation $z = x \oplus y \oplus w$ with the set of inequalities*

$$\mathbf{z} = \mathbf{XOR}(\mathbf{x}, \mathbf{y}, \mathbf{w}) \Leftrightarrow \begin{cases} \mathbf{w} + \mathbf{x} + \mathbf{y} + (1 - \mathbf{z}) \geq 1 \\ (1 - \mathbf{w}) + \mathbf{x} + \mathbf{y} + \mathbf{z} \geq 1 \\ \mathbf{w} + \mathbf{x} + (1 - \mathbf{y}) + \mathbf{z} \geq 1 \\ (1 - \mathbf{w}) + \mathbf{x} + (1 - \mathbf{y}) + (1 - \mathbf{z}) \geq 1 \\ \mathbf{w} + (1 - \mathbf{x}) + \mathbf{y} + \mathbf{z} \geq 1 \\ (1 - \mathbf{w}) + (1 - \mathbf{x}) + \mathbf{y} + (1 - \mathbf{z}) \geq 1 \\ \mathbf{w} + (1 - \mathbf{x}) + (1 - \mathbf{y}) + (1 - \mathbf{z}) \geq 1 \\ (1 - \mathbf{w}) + (1 - \mathbf{x}) + (1 - \mathbf{y}) + \mathbf{z} \geq 1 \end{cases}$$

Proposition 13. *Let $\alpha, \beta, \Delta \in \mathbb{F}_2^n$, $0 \leq k < n$ and $\mathbf{a}, \mathbf{b}, \boldsymbol{\delta}$ be vectors of n binary variables modeling α, β and Δ, respectively. Then we can model the relation $[(\alpha, k), (\beta, k)] \xrightarrow{\boxplus} (\delta, k)$ with the set of inequalities*

$$\begin{cases} \boldsymbol{\delta}_0 = \mathbf{XOR}(\mathbf{a}_0, \mathbf{b}_0) \\ \mathbf{f}(\mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\delta}_i, \mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \boldsymbol{\delta}_{i-1}) \quad 1 \leq i < n \end{cases}$$

where $\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6)$ *if given by*

$$\begin{cases} (1-\mathbf{x_1}) + \mathbf{x_2} + \mathbf{x_3} + \mathbf{x_4} + \mathbf{x_5} + \mathbf{x_6} \geq 1 \\ \mathbf{x_1} + (1-\mathbf{x_2}) + \mathbf{x_3} + \mathbf{x_4} + \mathbf{x_5} + \mathbf{x_6} \geq 1 \\ \mathbf{x_1} + \mathbf{x_2} + (1-\mathbf{x_3}) + \mathbf{x_4} + \mathbf{x_5} + \mathbf{x_6} \geq 1 \\ (1-\mathbf{x_1}) + (1-\mathbf{x_2}) + (1-\mathbf{x_3}) + \mathbf{x_4} + \mathbf{x_5} + \mathbf{x_6} \geq 1 \\ \mathbf{x_1} + (1-\mathbf{x_2}) + (1-\mathbf{x_3}) + (1-\mathbf{x_4}) + (1-\mathbf{x_5}) + (1-\mathbf{x_6}) \geq 1 \\ (1-\mathbf{x_1}) + \mathbf{x_2} + (1-\mathbf{x_3}) + (1-\mathbf{x_4}) + (1-\mathbf{x_5}) + (1-\mathbf{x_6}) \geq 1 \\ (1-\mathbf{x_1}) + (1-\mathbf{x_2}) + \mathbf{x_3} + (1-\mathbf{x_4}) + (1-\mathbf{x_5}) + (1-\mathbf{x_6}) \geq 1 \\ \mathbf{x_1} + \mathbf{x_2} + \mathbf{x_3} + (1-\mathbf{x_4}) + (1-\mathbf{x_5}) + (1-\mathbf{x_6}) \geq 1 \end{cases}$$

If $k > 0$*, the first constraint and the constraint for* $i = k$ *are omitted.*

Proposition 14. *Let* $\alpha, \beta, \Delta, \eta \in \mathbb{F}_2^n$*,* $0 \leq k < n$ *and* $\mathbf{a}, \mathbf{b}, \boldsymbol{\delta}, \boldsymbol{\eta}$ *be vectors of* n *binary variables modeling* α, β, Δ *and* η*, respectively. Then we can model the relations* $[(\alpha, k), (\beta, k)] \xrightarrow{\boxplus} (\delta, k)$ *and* $\eta = \mathrm{SHL}((\alpha \oplus \beta) \vee (\alpha \oplus \delta)$ *with the set of inequalities*

$$\begin{cases} \boldsymbol{\delta}_0 = \mathbf{XOR}(\mathbf{a}_0, \mathbf{b}_0) \\ \boldsymbol{\eta}_0 = 0 \\ \mathbf{g}(\mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\delta}_i, \mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \boldsymbol{\delta}_{i-1}, \boldsymbol{\eta}_i) \quad 1 \leq i < n \end{cases}$$

where $\mathbf{g}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7) \subseteq \mathbb{F}_2^7$ *is given by inequalities*

$$\begin{cases} 2\mathbf{x}_1 - 2\mathbf{x}_2 - 2\mathbf{x}_3 + 6\mathbf{x}_4 - \mathbf{x}_5 - 7\mathbf{x}_6 + 8\mathbf{x}_7 & \geq -4 \\ \mathbf{x}_1 - \mathbf{x}_2 + \mathbf{x}_3 + 4\mathbf{x}_4 + 4\mathbf{x}_5 + 4\mathbf{x}_6 - 3\mathbf{x}_7 & \geq 0 \\ -\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + 4\mathbf{x}_4 + 4\mathbf{x}_5 - 7\mathbf{x}_6 + 8\mathbf{x}_7 & \geq 0 \\ -3\mathbf{x}_1 + 3\mathbf{x}_2 - 3\mathbf{x}_3 + \mathbf{x}_4 - 2\mathbf{x}_5 - 2\mathbf{x}_6 + 4\mathbf{x}_7 & \geq -6 \\ 3\mathbf{x}_1 + 3\mathbf{x}_2 + 3\mathbf{x}_3 - 2\mathbf{x}_4 + \mathbf{x}_5 - 2\mathbf{x}_6 + 4\mathbf{x}_7 & \geq 0 \\ \mathbf{x}_1 + \mathbf{x}_2 - \mathbf{x}_3 - 3\mathbf{x}_4 - 3\mathbf{x}_5 + 7\mathbf{x}_6 + 7\mathbf{x}_7 & \geq 0 \\ -\mathbf{x}_1 - \mathbf{x}_2 - \mathbf{x}_3 - 3\mathbf{x}_4 - 3\mathbf{x}_5 + 7\mathbf{x}_6 + 7\mathbf{x}_7 & \geq -2 \\ -\mathbf{x}_1 - \mathbf{x}_2 + \mathbf{x}_3 - 4\mathbf{x}_4 - 4\mathbf{x}_5 - 4\mathbf{x}_6 - 3\mathbf{x}_7 & \geq -13 \end{cases}$$

Proposition 15. *Let* (δ, γ) *be an RX-differential over* n *bits, and for* $c \in \mathbb{F}_2^n$*, denote* $\oplus_c(x) = x \oplus c$*. Then we have the following propagation, with probability 1*

$$(\delta, \gamma) \xrightarrow{\oplus_c} (\delta \oplus c \oplus \overleftarrow{c}, \gamma)$$

Proof. Let (x, x') be a pair with RX-difference (δ, γ), i.e. $(x, x') = (x, \overleftarrow{x} \oplus \delta)$, and $(y, y') = (x \oplus c, x' \oplus c)$. Then we obviously have $y = x \oplus c$ along with

$$\begin{aligned} y' &= x' \oplus c \\ &= \overleftarrow{x} \oplus \delta \oplus c \\ &= \overleftarrow{y \oplus c} \oplus \delta \oplus c \\ &= \overleftarrow{y} \oplus \overleftarrow{c} \oplus \delta \oplus c \end{aligned}$$

Thus $(y, y') = (y, \overleftarrow{y} \oplus (\overleftarrow{c} \oplus \delta \oplus c))$ is a pair satisfying the RX-differential $(\delta \oplus c \oplus \overleftarrow{c}, \gamma)$. □

Hence the propagation $(a, \gamma) \xrightarrow{\oplus_c} (b, \gamma)$ for the addition with a constant c is easily modeled with

$$\left\{ \mathbf{b}^i = \mathbf{XOR}(\mathbf{a}^i, c_i, c_{(i-\gamma) \bmod n}) \quad 0 \leq i < n \right.$$

Proposition 16. *Let* (δ_1, γ) *and* (δ_2, γ) *be two RX-differential over* n *bits, and denote* $\oplus(x, y) = x \oplus y)$*, then we have the following propagation, with probability 1*

$$[(\delta_1, \gamma), (\delta_2, \gamma)] \xrightarrow{\oplus} (\delta_1 \oplus \delta_2, \gamma)$$

Proof. Let (x, x') (resp. (y, y')) be a pair with RX-difference (δ_1, γ) (resp. (δ_2, γ)) , i.e. $(x, x') = (x, \overleftarrow{x} \oplus \delta_1)$ (resp. $(y, y') = (y, \overleftarrow{y} \oplus \delta_2)$). Denote $(z, z') = (x \oplus y, x' \oplus y')$. Then we obviously have $z = x \oplus y$ and

$$\begin{aligned} z' &= x' \oplus y' \\ &= \overleftarrow{x} \oplus \delta_1 \oplus \overleftarrow{y} \oplus \delta_2 \\ &= \overleftarrow{x \oplus y} \oplus \delta_1 \oplus \delta_2 \\ &= \overleftarrow{z} \oplus \delta_1 \oplus \delta_2 \end{aligned}$$

Then $(z, z') = (z, \overleftarrow{z} \oplus (\delta_1 \oplus \delta_2))$ is a pair satisfying the RX-differential $(\delta_1 \oplus \delta_2, \gamma)$. □

Hence the propagation $[(a, \gamma), (b, \gamma)] \xrightarrow{\oplus} (\delta, \gamma)$ for the XOR can be easily modeled with

$$\left\{ \mathbf{\Delta}^i = \mathbf{XOR}(\mathbf{a}^i, \mathbf{b}^i) \quad 0 \le i < n \right.$$

Proposition 17. *Let (δ, γ) be an RX-differential over n bits, and for $0 \le \eta < n$, denote $\lll_\eta (x) = x \lll \eta$. Then we have the following propagation, with probability 1*

$$(\delta, \gamma) \xrightarrow{\lll_\eta} (\delta \lll \eta, \gamma)$$

Proof. Let (x, x') be a pair with RX-difference (δ, γ), i.e. $(x, x') = (x, \overleftarrow{x} \oplus \delta)$, and $(y, y') = (x \lll \eta, x' \lll \eta)$. Then we obviously have $y = x \lll \eta$ along with

$$\begin{aligned} y' &= x' \lll \eta \\ &= (\overleftarrow{x} \oplus \delta) \lll \eta \\ &= (\overleftarrow{x} \lll \eta) \oplus (\delta \lll \eta) \\ &= \overleftarrow{x \lll \eta} \oplus (\delta \lll \eta) \\ &= \overleftarrow{y} \oplus (\delta \lll \eta) \end{aligned}$$

Thus $(y, y') = (y, \overleftarrow{y} \oplus (\delta \lll \eta))$ is a pair satisfying the RX-differential $(\delta \lll \eta, \gamma)$. □

Hence the propagation $(a, \gamma) \xrightarrow{\lll_\eta} (b, \gamma)$ for the cyclic rotation by η is easily modeled with

$$\left\{ \mathbf{b}^i = \mathbf{a}^{(i-\eta) \bmod n} \quad 0 \le i < n \right.$$