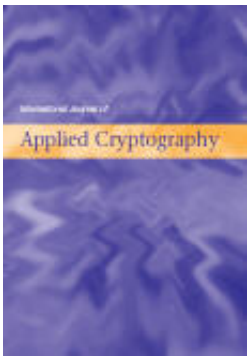


International Journal of Applied Cryptography

ISSN online: 1753-0571 - ISSN print: 1753-0563
https://www.inderscience.com/ijact

Finding differential trails on ChaCha by means of state functions

Emanuele Bellini, Juan Grados, Rusydi H. Makarim, Carlo Sanna

DOI: 10.1504/IJACT.2024.10063705

Article History:

Received:	18 October 2021
Last revised:	28 February 2023
Accepted:	08 March 2023
Published online:	03 May 2024

Finding differential trails on ChaCha by means of state functions

Emanuele Bellini*, Juan Grados and Rusydi H. Makarim

Cryptography Research Center,
Technology Innovation Institute,
Abu Dhabi, UAE
Email: emanuele.bellini@tii.ae
Email: juan.grados@tii.ae
Email: rusydi.makarim@tii.ae
*Corresponding author

Carlo Sanna

GNSAGA of INdAM and of CrypTO,
The Group of Cryptography and Number Theory of Politecnico di Torino,
Politecnico di Torino,
Torino, Italy
Email: carlo.sanna.dev@gmail.com

Abstract: We provide fast algorithms to compute the exact additive and XOR differential probabilities of ChaCha20 half quarter-round H and, under an independence assumption, an approximation of the differential probabilities of the full quarter-round. We give experimental evidence of the correctness of our approximation, and show that the independence assumption holds better for the XOR differential probability than the additive differential probability. We then propose an efficient greedy strategy to maximise differential characteristics for the full quarter-round, and use it to determine explicit differential trails for the ChaCha permutation. We also provide an MILP model to search for differential trails in ChaCha and compare its performance and effectiveness with our method. We believe these results might bring new insights in the differential cryptanalysis of ChaCha20 and of similar ARX ciphers.

Keywords: ChaCha20; differential cryptanalysis; additive differential probability; XOR differential probability; state functions.

Reference to this paper should be made as follows: Bellini, E., Grados, J., Makarim, R.H. and Sanna, C. (2023) 'Finding differential trails on ChaCha by means of state functions', *Int. J. Applied Cryptography*, Vol. 4, Nos. 3/4, pp.156–175.

Biographical notes: Emanuele Bellini is a Senior Director of the Cryptography Research Center of the Technology Innovation Institute in Abu Dhabi, UAE. He also worked as a Cryptographer in DarkMatter LLC (UAE) and Telsy S.p.A. (Italy). He received his PhD in Coding Theory and Cryptography from the University of Trento, Italy, and Master's and Bachelor's degree from the University of Turin, Italy.

Juan Grados obtained his BSc from the Universidad Nacional de Trujillo in Peru in 2009. Subsequently, he was awarded a scholarship to pursue his MSc and PhD at the National Laboratory for Scientific Computing (LNCC) in Brazil, where he earned both degrees in 2010 and 2018, respectively. He then gained experience working in cybersecurity for various companies in Brazil before joining the Cryptography Research Centre at the Technology Innovation Institute, a private research institution located in Abu Dhabi, UAE. His primary research focuses on symmetric cryptography, with a particular interest in cryptanalysis of stream and block ciphers.

Rusydi H. Makarim is a Cryptographer. His research work focuses on the design and the cryptanalysis of symmetric-key primitives. Previously he worked as a Lead Cryptography Analyst at the Cryptography Research Center, Technology Innovation Institute (TII). Prior to TII, he was a PhD student at the Mathematical Institute, University Leiden and CWI Cryptology Group, working on the development of Groebner bases algorithm tailored for the cryptanalysis of multivariate public-key cryptosystem. His main research interests are techniques to solve multivariate quadratic polynomial equations over a finite field and cryptographic aspect of (vectorial) Boolean functions. He is also a contributor to SageMath.

Carlo Sanna is a researcher (RTDb) working in the Group of Cryptography and Number Theory of the Department of Mathematical Sciences of the Polytechnic of Turin, Italy. He received his PhD in Pure and Applied Mathematics from the University of Turin (joint with the Polytechnic of Turin), and Master's and Bachelor's degree from the University of Turin.

1 Introduction

Due to their efficiency in software, to their simple description, and to their resistance against timing attacks, ARX ciphers have become among the most popular symmetric constructions. These ciphers are based on only three basic bitwise operations: modular *addition*, bitwise *rotation*, and *eXclusive* OR, hence the name ARX.

A non-exhaustive list of the most popular ARX symmetric ciphers includes:

1 Cryptographic permutations such as SPARKLE (SCHWAEMM and ESCH) (Beierle et al., 2019), candidate to the NIST Lightweight Cryptography standardisation process (NIST LWC) (NIST, 2019).

2 Block ciphers such as the Rivest cipher RC5 (Rivest, 1994), the South Korean Electronic and Telecommunication Research Institute cipher LEA (Hong et al., 2013) the NIST LWC candidate Limdolen (Mehner, 2019, using a Feistel structure and ARX operations to achieve diffusion), the American NSA cipher Speck (Beaulieu et al., 2015) standardisation in ISO/IEC 29167-22, the tiny encryption algorithm (TEA) (Wheeler and Needham, 1994) and Threefish (2010), used as Skein internal permutation.

3 Stream ciphers such as Bernstein's Salsa20 (Bernstein, 2005, 2008b) and ChaCha20 (Bernstein, 2008a). The latter one is part of the TLS 1.3 standard.

4 Hash functions such as the SHA-3 Project (NIST, 2007) finalists (2007–2012) BLAKE2 (Aumasson et al., 2013) and Skein (Ferguson et al., 2010), and other SHA-3 candidates, Blue Midnight Wish (Gligoroski et al., 2009), CubeHash (Bernstein, 2008c), Shabal (Bresson et al., 2008), SIMD (Leurent et al., 2009).

5 Message authentication codes such as Chaskey (Mouha et al., 2014), standardised in ISO/IEC 29192-6.

A common technique to evaluate the security of a symmetric cipher is differential cryptanalysis. In order for this technique to be successful, the attacker needs to find input/output pairs of a cipher such that they have a fixed difference, called *differential characteristic*, with respect to a certain operation. These characteristics must occur more or less often than how they would occur in a random function. In order to compute the probability for such a characteristic to occur, one has to break the cipher in smaller components and study how the probability propagates through these components. Despite several works investigated the problem just described in the case of ARX constructions, its accurate calculation still remains an open problem for those ARX ciphers with large components and/or a large state, as it is the case, for example, for Salsa20, ChaCha20, or BLAKE2.

1.1 Related works

As mentioned above, one of the first steps to assess the security against differential cryptanalysis is to efficiently and accurately evaluate the probability with which differences with respect to a certain operation propagate through the basic components of a cipher and through their composition. In the case of ARX ciphers, one might consider differences with respect to the three ARX operations. In this work, we will only focus on exclusive or and modular addition differences.

The first to determine an exact formula to compute the XOR differential probability of modular addition, denoted as $\mathrm{xdp}^{\boxplus}$, in a linear time with respect to the input size, were Lipmaa and Moriai (2001). Note that, in general, if n is the size of the input, it is not possible to perform such operation faster than $O(n)$, as one must read the entire input at least [although faster than $O(n)$ is possible if differences are sparse, see Mouha et al., 2010]. In 2004, Lipmaa et al. obtained the dual result of Lipmaa and Moriai (2001), by computing the additive differential probability of the XOR operation, denoted by $\mathrm{adp}^{\oplus}$.

In 2005, in his PhD thesis, Daum (2005) collected a set of differential properties of bit rotation; in particular he defined the additive and the XOR differential probability of bitwise rotation, $\mathrm{adp}^{\lll}$ and $\mathrm{xdp}^{\lll}$.

Taking inspiration from the cryptanalysis techniques for SHA-1 by De Canniere and Rechberger (2006) and Mouha et al. (2009), the results of Lipmaa and Moriai (2001) and Lipmaa et al. (2004) were generalised by Mouha et al. (2010). In this work, the authors introduced the elegant theory of *state functions* (S-functions in brief). These provided a unified framework to compute the XOR differential probability of modular addition, even when this has more then two inputs, and, consequently, of multiplication by a constant, and the additive differential probability of the XOR operation. S-functions allow to derive differential properties by means of simple matrix multiplications.

Even knowing how the probability with which additive or XOR differences propagates through basic operations, such as modular addition, XOR or rotation, it is not straightforward to compute how this probability propagates through compositions of these operations. In particular, Velichkov et al. (2011) showed how to compute the additive differential probability of what they called the ARX operation, i.e., $\mathrm{ARX}(a, b, r, d) = ((a \boxplus b) \lll r) \oplus d$. They

also showed that, due to the input/output dependency of the three operations, this differential probability differs significantly from the simple multiplication of the differential probability of each operation. Indeed, the accurate calculation of the probability of a differential characteristic still remains an open problem for many ARX constructions.

The just mentioned results have been used to mount cryptographic attacks to several ciphers. Aumasson et al. (2009), use the algorithms provided in Lipmaa and Moriai (2001) for computing differential properties of modular addition to find modular differentials and mount a boomerang attack on Threefish. Since the methods from Velichkov et al. (2011) do not scale well with large components. In 2012, Velichkov (2012) introduced the concept of a UNAF difference, representing a set of specially chosen additive differences. This allows them to find a three-round differential trail in Salsa stream cipher of probability 2^{-4}, and then to mount a key recovery attack on Salsa reduced to five rounds, with data complexity of 27 chosen plaintexts and time complexity of 2,167 encryptions. A couple of years later, the results from Lipmaa and Moriai (2001), Lipmaa et al. (2004), Mouha et al. (2010) and Velichkov et al. (2011) were used by Biryukov et al. to instantiate automatic search of differential trails in TEA, XTEA, RAIDEN (Biryukov and Velichkov, 2014), and in SPECK (Biryukov et al., 2016) block ciphers.

Table 1 Weight of the best differential trails found using both the S-function (for XOR and ModAdd differential trails) and the MILP (only for XOR differential trails) techniques

	Input difference		
	nonce, counter	*nonce, counter*	*full state*
	Technique		
	S-function	*MILP*	*S-function*
Round	*XOR-diff.*	*XOR-diff.*	*ModAdd-diff*
1	3	3	208
2	37	37	286
3	157	147	304
4	349	316	306

Currently, the best known attacks on ChaCha20 stream cipher are derivations of the work of Aumasson et al. (2008), which is a differential-linear key recovery attack. Most recent variants of this work include (Shi et al., 2012; Dey and Sarkar, 2017; Beierlee et al., 2020; Coutinho and Neto, 2021; Dey et al., 2022). In these attacks, one round XOR-differential trails with average probability $2^{-4.5}$ and input difference injected in the nonce/counter is used. Truncated XOR-differential trails (1-bit input difference and 1-bit output difference) for three rounds and probability $2^{-5.26}$ are used, e.g., in Aumasson et al. (2008). A XOR-differential trail for two rounds and probability 2^{-24} was found by Aaraj et al. (2017) by means of MILP. In this work the input difference was injected in the full state.

1.2 Our contribution

In this work, we slightly generalise the theory of S-functions, to be able to compute the exact additive and XOR differential probability of ChaCha20 half quarter-round H. Supposing independence among two consecutive applications of H, we are able to compute also the differential probability of the full quarter-round. We also provide experimental evidence of the correctness of our approximation, and show that the independence assumption seems to hold better for the xdp rather than the adp. We also propose a greedy strategy to maximise differential characteristic probability for the full quarter-round, and then use this strategy to find explicit XOR and additive differential trails up to four rounds. We also implement an MILP model to find XOR-differential trails and compare the best trails found with the S-function technique. The results are summarised in Table 1.[1] The code to reproduce our results can be found at https://github.com/Crypto-TII/chacha differential trails with s-functions.

We believe these results might bring new insights in the differential cryptanalysis of ChaCha20 and of similar constructions.

1.3 Outline

In Section 2, we introduce the necessary notions to describe our result. We devote from Subsection 3.1 to Subsection 3.3 to the maximisation of the XDP for ChaCha quarter round, while from Subsection 4.2 to Subsection 4.3 we deal with the same problem in the ADP case. In Section 5, we provide explicit differential trails and simple statistics on the minimum, maximum and average quarter round differential characteristic probability. In Section 6, we describe an MILP model to find differential trails for ChaCha internal permutation. Finally in Section 7, we draw the conclusions and point to possible future developments of this research.

2 Preliminaries

In this section, we first define the notation we adhere to, we recall ChaCha20 specifications, formally define the concept of XDP and ADP, and the theory of S-functions.

2.1 Notation

For every positive integer n, let $\mathbb{W}_n$ denote the set of n-bits words. For all $x, y \in \mathbb{W}_n$, we use the following notation:

$x[i]$	i^{th} bit of x
$x \oplus y$	bitwise XOR of x and y
$x \boxplus y$	addition modulo 2^n of x and y
$x \boxminus y$	subtraction modulo 2^n of x and y
$x \lll r$	left rotation of x by r bits
$x \ggg r$	right rotation of x by r bits

$x \| y$ concatenation of x and y

Moreover, for vectors $\mathrm{x}, \mathrm{y}, \in \mathbb{W}_n^k$, all the previous operations are extended component wise. Also, we write $\mathbb{F}_2$ for the field of two elements, and $\lfloor t \rfloor$ for the greatest integer not exceeding t.

2.2 ChaCha stream cipher

ChaCha20 stream cipher has a state of 512 bits, which can be seen as a 4×4 matrix whose elements are binary vectors of w = 32 bits, i.e.

$$X = \{x_{i,j}\}_{\substack{i=0,\ldots,3 \\ j=0,\ldots,3}} = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix} \in \mathcal{M}_{n\times n}\left(\mathbb{F}_2^{\mathrm{w}}\right).$$

Definition 1 (ChaCha half quarter round): Let x_i, y_i, i = 0, 1, 2, 3 be w-bit words and r_1, $r_2 \in \{1, \ldots, \mathrm{w} - 1\}$. Then we define ChaCha half quarter round $(y_0, y_1, y_2, y_3) = \mathrm{HQR}_{n r 2}(x_0, x_1, x_2, x_3)$ as follows:

$$\begin{aligned} y_0 &= x_0 \boxplus x_1 \\ y_3 &= (y_0 \oplus x_3) \lll r_1 \\ y_2 &= y_3 \boxplus x_2 \\ y_1 &= (y_2 \oplus x_1) \lll r_2. \end{aligned}$$

Definition 2 (ChaCha quarter round): Let x_i, y_i, i = 0, 1, 2, 3 be w-bit words and r_1, r_2, r_3, $r_4 \in \{1, \ldots, \mathrm{w} - 1\}$. Then we define ChaCha quarter round

$$(y_0, y_1, y_2, y_3) = \mathrm{QR}(x_0, x_1, x_2, x_3)$$

as follows:

$$\begin{aligned} (y_0', y_1', y_2', y_3') &= \mathrm{HQR}_{n r 2}(x_0, x_1, x_2, x_3) \\ (y_0, y_1, y_2, y_3) &= \mathrm{HQR}_{r_3 r_4}(y_0', y_1', y_2', y_3') \end{aligned}$$

We show in Figure 1 a schematic drawing of Chacha half quarter round. The permutation used in ChaCha20 stream cipher performs 20 rounds or, equivalently, ten *double rounds*. Two consecutive rounds (or a *double round*) of ChaCha permutation consist in applying the quarter round four times in parallel to the columns of the state (first round), and then four times in parallel to the diagonals of the state (second round). More formally:

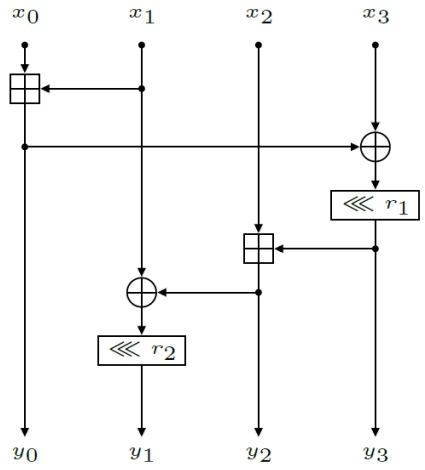
$\boxplus$

Definition 3 (ChaCha column/diagonal round): We let $X = \{x_{i,j}\}_{\substack{i=0,\ldots,3 \\ j=0,\ldots,3}}$ and $Y = \{y_{i,j}\}_{\substack{i=0,\ldots,3 \\ j=0,\ldots,3}}$ be two $n \times n$ matrices with entries in $\mathbb{F}_2^{\mathrm{w}}$.

A *column round* $Y = \mathcal{R}^{\mathrm{C}}(X)$ is defined as follows, with i = 0, 1, 2, 3:

$$(y_{0,i}, y_{1,i}, y_{2,i}, y_{3,i}) = \mathrm{QR}(x_{0,i}, x_{1,i}, x_{2,i}, x_{3,i}).$$

Figure 1 The ChaCha half quarter round diagram



A *diagonal round* $Y = \mathcal{R}^{\mathrm{D}}(X)$ is defined as follows, for i = 0, 1, 2, 3 and where each subscript is computed modulo n = 4:

$$(y_{0,i}, y_{1,i+1}, y_{2,i+2}, y_{3,i+3}) = \mathrm{QR}(x_{0,i}, x_{1,i+1}, x_{2,i+2}, x_{3,i+3}).$$

2.3 Differential probabilities

Definition 4: Let F be a function from $\mathbb{W}_n^k \to \mathbb{W}_n^h$ and let $(\Delta\mathbf{x}, \Delta\mathbf{y}) \in \mathbb{W}_n^k \times \mathbb{W}_n^h$. The *XOR differential probability* (XDP) of F with respect to the input/output pair $(\Delta\mathbf{x}, \Delta\mathbf{y})$ is defined as

$$\mathrm{xdp}^F(\Delta\mathbf{x} \to \Delta\mathbf{y}) = \Pr_{\mathbf{x}\in\mathbb{W}_n^k}[F(\mathbf{x} \oplus \Delta\mathbf{x}) = F(\mathbf{x}) \oplus \Delta\mathbf{y}],$$

where the probability is meant for an uniformly distributed random variable $\mathbf{x} \in \mathbb{W}_n^k$. Similarly, the *additive differential probability* (ADP) of F with respect to the input/output pair $(\Delta\mathbf{x}, \Delta\mathbf{y})$ is defined as

$$\mathrm{adp}^F(\Delta\mathbf{x} \to \Delta\mathbf{y}) = \Pr_{\mathbf{x}\in\mathbb{W}_n^k}[F(\mathbf{x} \boxplus \Delta\mathbf{x}) = F(\mathbf{x}) \boxplus \Delta\mathbf{y}].$$

In general, there is no simple way to express the differential probability of the composition of two functions in terms of the differential probabilities of the single functions. However, we have the following result:

Lemma 1: Let $F : \mathbb{W}_n^k \to \mathbb{W}_n^h$ and $G : \mathbb{W}_n^h \to \mathbb{W}_n^\ell$ be two functions, and let $(\Delta\mathbf{x}, \Delta\mathbf{z}) \in \mathbb{W}_n^k \times \mathbb{W}_n^\ell$. Assume that:

1 For an uniform random variable $\mathbf{x} \in \mathbb{W}_n^k$, the events $F(\mathbf{x} \boxplus \Delta\mathbf{x}) = F(\mathbf{x}) \boxplus \Delta\mathbf{y}$ and $G(F(\mathbf{x}) \boxplus \Delta\mathbf{y}) = G(F(\mathbf{x})) \boxplus \Delta\mathbf{z}$ are independent, for every $\Delta\mathbf{y} \in \mathbb{W}^k$.

2 $$\Pr_{\mathbf{x}\in\mathbb{W}_n^k}[G(F(\mathbf{x}) \boxplus \Delta\mathbf{y}) = G(F(\mathbf{x})) \boxplus \Delta\mathbf{z}] = \Pr_{\mathbf{w}\in\mathbb{W}_n^h}[G(\mathbf{w} \boxplus \Delta\mathbf{y}) = G(\mathbf{w}) \boxplus \Delta\mathbf{z}].$$

Then we have

$$\mathrm{xdp}^{G\circ F}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = \sum_{\Delta\mathbf{y}\in\mathbb{W}_n^k} \mathrm{xdp}^F(\Delta\mathbf{x} \to \Delta\mathbf{y})\mathrm{xdp}^G(\Delta\mathbf{y} \to \Delta\mathbf{z}), \quad (1)$$

and

$$\mathrm{adp}^{G\circ F}(\Delta\mathbf{x}\to\Delta\mathbf{z}) = \sum_{\Delta\mathbf{y}\in\mathbb{W}_n^k} \mathrm{adp}^{F}(\Delta\mathbf{x}\to\Delta\mathbf{y})\mathrm{adp}^{G}(\Delta\mathbf{y}\to\Delta\mathbf{z}). \quad (2)$$

Proof: Using the definition of XDP and the assumptions, we get

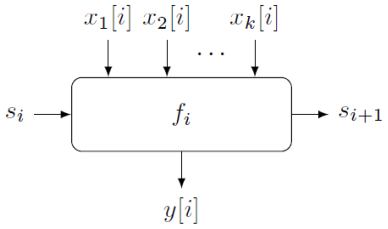
$$\begin{aligned}
\mathrm{xdp}^{G\circ F}(\Delta\mathbf{x}\to\Delta\mathbf{z}) &= \Pr[G(F(\mathbf{x}\boxplus\Delta\mathbf{x})) = G(F(\mathbf{x}))\boxplus\Delta\mathbf{z}] \\
&= \sum_{\Delta\mathbf{y}\in\mathbb{W}_n^k} \Pr[F(\mathbf{x}\boxplus\Delta\mathbf{x}) = F(\mathbf{x})\boxplus\Delta\mathbf{y} \wedge G(F(\mathbf{x})\boxplus\Delta\mathbf{y}) \\
&\quad = G(F(\mathbf{x}))\boxplus\Delta\mathbf{z}] \\
&= \sum_{\Delta\mathbf{y}\in\mathbb{W}_n^k} \Pr[F(\mathbf{x}\boxplus\Delta\mathbf{x})) = F(\mathbf{x})\boxplus\Delta\mathbf{y}]\Pr[G(F(\mathbf{x})\boxplus\Delta\mathbf{y}) \\
&\quad = G(F(\mathbf{x}))\boxplus\Delta\mathbf{z}] \\
&= \sum_{\Delta\mathbf{y}\in\mathbb{W}_n^k} \Pr[F(\mathbf{x}\boxplus\Delta\mathbf{x})) = F(\mathbf{x})\boxplus\Delta\mathbf{y}]\Pr[G(\mathbf{w}\boxplus\Delta\mathbf{y}) \\
&\quad = G(\mathbf{w})\boxplus\Delta\mathbf{z}] \\
&= \sum_{\Delta\mathbf{y}\in\mathbb{W}_n^k} \mathrm{xdp}^{F}(\Delta\mathbf{x}\to\Delta\mathbf{y})\mathrm{xdp}^{G}(\Delta\mathbf{y}\to\Delta\mathbf{z}),
\end{aligned}$$

as claimed. A similar reasoning gives the result on the ADP.□

Even if F and G do not satisfy the two assumptions of Lemma 1, it can be that (1) and (2) are good approximations for the XDP and ADP of the composition $F \circ G$. In general, (1) and (2) can be used as heuristic formulas for the differential probabilities of $F \circ G$.

Definition 5: Given two functions $F:\mathbb{W}_n^k \to \mathbb{W}_n^h$ and $G:\mathbb{W}_n^h \to \mathbb{W}_n^\ell$, we say that F and G are 'independent' if (1) and (2) are good approximations for the differential probabilities of $F \circ G$.

Figure 2 Representation of the i^{th} block of an S-machine



2.4 S-functions

This section contains the preliminaries on S-functions (short for 'state functions') needed for the computation of the ADP of half quarter round performed in Subsection 4.1. Actually, we shall develop a bit more theory than the one strictly necessary for Subsection 4.1.

S-functions were introduced in Mouha et al. (2010) and were already applied to the differential cryptanalysis of some ARX primitives (Mouha et al., 2010; Velichkov et al., 2011). Here we redefine S-functions in a slightly more general way, which is better suited for our purposes. Throughout this section, let n and k be fixed positive integers.

Definition 6: An *S-machine* is a $(n + 2)$-tuple $(\mathcal{S}, s_{in}, f_0, \ldots, f_{n-1})$ consisting of:

- a finite set of *states* $\mathcal{S}$
- an initial state $s_{in} \in \mathcal{S}$
- n partial functions $f_i : \mathcal{S}\times\mathbb{F}_2^k \to \mathcal{S}\times\mathbb{F}_2$ called *transitions functions*.

An S-machine can be represented as a device built of n blocks labelled by $i = 0, \ldots, n - 1$ (see Figure 2). Starting from $i = 0$, the i^{th} block takes as input the current state s_i and the bits $x_1[i], \ldots, x_k[i]$. If $(s_i, x_1[i], \ldots, x_k[i]) \in \mathrm{dom}(f_i)$ then the block returns as output $y[i]$ and the next state s_{i+1}, which is fed to the $(i+1)^{\text{th}}$ block, if any. If $(s_i, x_1[i], \ldots, x_k[i]) \notin \mathrm{dom}(f_i)$ then the computation stops. Considering when the computation is performed through all the n blocks leads to the definition of S-functions.

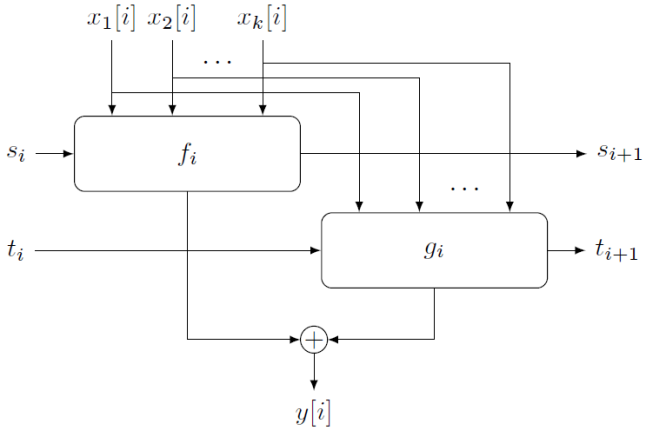
Definition 7: An S-function F is a partial function $(\mathbb{F}_2^n)^k \to \mathbb{F}_2^n$ such that there exists an S-machine $(\mathcal{S}, s_{in}, f_0, \ldots, f_{n-1})$ with the following property: For every $(x_1, \ldots, x_k) \in \mathrm{dom}(F)$ there exist some states $s_0 = s_{in}, s_1, \ldots, s_n \in \mathcal{S}$ such that

$$\begin{aligned}
&(s_i, x_1[i], \ldots, x_k[i]) \in dom(f_i), \\
&\qquad (s_{i+1}, y[i]) = f_i(s_i, x_1[i], \ldots, x_k[i]) \quad \text{for } i = 0, 1, \ldots, n-1,
\end{aligned}$$

where $y = F(x_1, \ldots, x_k)$. In other words, an S-function is a partial function that is computed by an S-machine.

Remark 1: Our definition of S-function differs from the one given in Mouha et al. (2010) in two ways. First, in Mouha et al. (2010), the transition functions f_i for $i = 0, \ldots, n - 1$ are all equal to a single function f, although a generalisation with different transition functions is already suggested. Second, and more important, our definition lets the transition functions be *partial* functions, while in Mouha et al. (2010) only total functions are considered.

Figure 3 The i^{th} block of the XOR of two S-functions

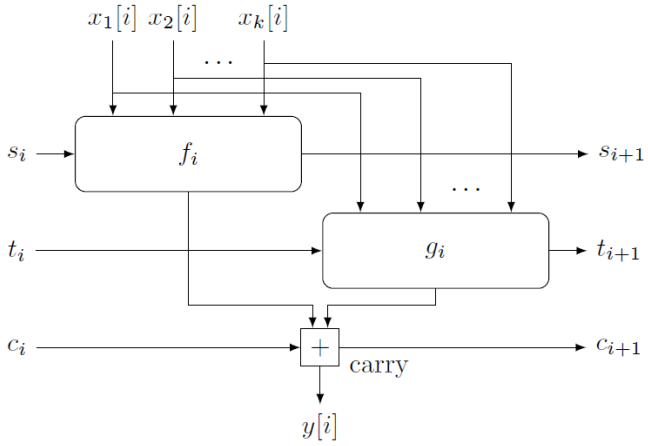


It is easy to see that, among the functions $(\mathbb{F}_2^n)^k \to \mathbb{F}_2$ all coordinate projections $(x_1, \ldots, x_k) \mapsto x_j$, with $j \in \{1, \ldots,$

$n\}$, and all constant functions $(x_1, \ldots, x_k) \mapsto c$, with $c \in \mathbb{F}_2^n$, are S-functions. The next lemma shows that the set of S-functions is closed by addition and XOR.

Lemma 2: If F and G are S-functions, then $F \oplus G$ and $F \boxplus G$ are S-functions.

Figure 4 The i^{th} block of the addition of two S-functions



Proof: Let $(\mathcal{S}, s_{in}, f_0, \ldots, f_{n-1})$ and $(\mathcal{T}, t_{in}, g_0, \ldots, g_{n-1})$ be the S-machines of F and G, respectively. The S-machine computing $F \oplus G$ has set of states $\mathcal{S} \times \mathcal{T}$, initial state (s_{in}, t_{in}), and i^{th} block built from f_i and g_i as shown in Figure 3. The S-machine computing $F \boxplus G$ is only slightly more complex, because it has to take care of the propagation of carries. It has set of states $\mathcal{S} \times \mathcal{T} \times \mathbb{F}_2$, initial state $(s_{in}, t_{in}, 0)$, and i^{th} block built from f_i and g_i as shown in Figure 4.□

Remark 2: In general, rotations cannot be computed by S-functions. Indeed, already the simple rotation $x_1 \lll 1$ cannot be computed by an S-function, since the least significant bit of $x_1 \lll 1$ is $x_1[n-1]$, which is not a function of $x_1[0], \ldots, x_k[0]$.

Definition 8: Let F be an S-function with S-machine $(\mathcal{S}, s_{in}, f_0, \ldots, f_{n-1})$, and let $i \in \{0, \ldots, n-1\}$ and $\gamma \in \mathbb{F}_2$. Also, let $s_1 := s_{in}, s_2, \ldots, s_h$ be all the elements of $\mathcal{S}$. The i^{th} transition matrix of F is the $h \times h$ matrix $A_{i,\gamma} = (a_{j,\ell})_{1 \le j,\ell \le h}$ where $a_{j,\ell}$ is equal to the number of $\chi_1, \ldots, \chi_k \in \mathbb{F}_2$ such that $(s_j, \chi_1, \ldots, \chi_k) \in \text{dom}(f_j)$ and $(s_\ell, \gamma) = f_j(s_j, \chi_1, \ldots, \chi_k)$.

The next theorem is the key result about counting solutions of equations involving S-functions.

Theorem 1: Let F be an S-function and let $y \in \mathbb{F}_2^n$. Then, we have that the number of $(x_1, \ldots, x_k) \in \text{dom}(F)$ such that $F(x_1, \ldots, x_k) = y$ is equal to

$$LA_{0,y[0]}A_{1,y[1]} \cdots A_{n-1,y[n-1]}C$$

where $L := (1, 0, \ldots, 0)$ is a row vector of length h, $C = (1, 1, \ldots, 1)^\top$ is a column vector of length h, and $A_{i,\gamma}$ are the transition matrices of F.

Proof: Let $(\mathcal{S}, s_{in}, f_0, \ldots, f_{n-1})$ be the S-machine of F and let $s_1 := s_{in}, s_2, \ldots, s_h$ be all the states in $\mathcal{S}$. We build a directed graph G in the following way. The vertices of G are the pairs (i, s_j), where $i = 0, \ldots, n-1$ and $j = 1, \ldots, h$. For all $i = 0, \ldots, n-2$, $j, \ell = 1, \ldots, h$ and $\chi_1, \ldots, \chi_k \in \mathbb{F}_2$, if $(s_\ell, y[i]) = f_i(s_j, \chi_1, \ldots, \chi_k)$ then we draw an edge from (i, s_j) to $(i+1, s_\ell)$. (Note that we can draw multiple edges between two vertices). Hence, by the definition of S-function, the $(x_1, \ldots, x_k) \in \text{dom}(F)$ such that $F(x_1, \ldots, x_k) = y$ are in bijection with the direct paths from $(0, s_1)$ to one of $(n, s_1), \ldots, (n, s_h)$. Moreover, $A_{i,y[i]}$ is the adjacency matrix of the subgraph consisting of vertices (i, s_j), $(i+1, s_\ell)$. By a well-known result of graph theory (Chittenden, 1947), the (j, ℓ) entry of the matrix $B := A_{0,y[0]}A_{1,y}[1] \cdots A_{n-1,y[n-1]}$ is equal to the number of direct paths from $(0, s_j)$ to (n, s_ℓ). Then the claim follows since LBC is equal to the sum of the elements in the first row of B.□

Remark 3: More generally, the (i, j) entry of the matrix

$$A_{0,y[0]}A_{1,y[1]} \cdots A_{n-1,y[n-1]}$$

is equal to the number of $(x_1, \ldots, x_k) \in \text{dom}(F)$ that leads the S-machine associated to F from state i to state j.

2.5 *Rotate, add, and rotate back*

For every integer $r \in [0, n)$, let us define the operator $\overleftrightarrow{\boxplus}^r$ by

$$x \overleftrightarrow{\boxplus}^r y := \overleftarrow{\overrightarrow{x} \boxplus \overrightarrow{y}}$$

for all $x, y \in \mathbb{W}_n$, where the arrows denote left/right rotations by r bits. Letting $x = x_L \| x_R$ and $y = y_L \| y_R$, where $x_L, y_L \in \mathbb{W}_r$ and $x_R, y_R \in \mathbb{W}_{n-r}$, it follows that

$$\begin{aligned} x \overleftrightarrow{\boxplus}^r y &= \overleftarrow{\overrightarrow{(x_L \| x_R)} \boxplus \overrightarrow{(y_L \| y_R)}} \\ &= \overrightarrow{(x_R \| x_R) \boxplus (y_R \| y_L)} \\ &= \overrightarrow{(x_R \boxplus y_R \boxplus c) \| (x_L \boxplus y_L)} \\ &= (x_L \boxplus y_L) \| (x_R \boxplus y_R \boxplus c), \end{aligned}$$

where $c := \lfloor (x_L + y_L) = 2^r \rfloor$. Hence, the computation $x \overleftrightarrow{\boxplus}^r y$ proceeds almost as the addition modulo 2^n addition of x and y, with the only differences that: there is no carry propagation from the $(n-r)^{\text{th}}$ digit; and the carry c of the nth digit is added to the least significant digit. In particular, note that $x \overleftrightarrow{\boxplus}^r y$ cannot be computed by an S-function, since its least significant bit depends on c, which in turn depends on the bits of x and y after the $(n-r)^{\text{th}}$ position. However, assuming that we know the value of c in advance, we can compute $x \overleftrightarrow{\boxplus}^r y$ by an S-machine and check at the end that the n^{th} carry is actually equal to c. This would be our strategy to prove Lemma 6 later.

3 XOR differential probability of ChaCha round

In this section, we first give an exact formula for the XOR differential probability of the half quarter round of ChaCha. Then we provide a heuristic formula for the XDP of ChaCha quarter round, under the assumption that the two half quarter rounds are 'independent'. Finally, we illustrate a greedy strategy to find the best XDP of ChaCha full round.

3.1 XDP of ChaCha half quarter round

Here, we give a formula for the XOR differential probability of the half quarter round of ChaCha. First, we need a formula for the XDP of modular addition. This was computed in Mouha et al. (2010) using S-functions.

First let us define the matrices that are going to be used in the next lemma:

$$A_{000}=\begin{pmatrix}2&0\\0&0\end{pmatrix}\quad A_{001}=\begin{pmatrix}0&0\\1&1\end{pmatrix}\quad A_{011}=\begin{pmatrix}1&1\\0&0\end{pmatrix}\quad A_{111}=\begin{pmatrix}0&0\\0&2\end{pmatrix}$$

with the remaining matrices given by $A_{010} = A_{100} = A_{001}$ and $A_{101} = A_{110} = A_{011}$, and $L = (1\ 0)$, $C = (1\ 1)^T$.

Lemma 3: Let A_w, L, and C be the matrices defined above. Then, we have that for all $\Delta x_0, \Delta x_1, \Delta y \in \mathbb{W}_n$:

$$\mathrm{xdp}^{\boxplus}(\Delta x_0, \Delta x_1 \to \Delta y) = 2^{-n} L A_{w[0]} A_{w[1]} \cdots A_{w[n-1]} C,$$

where $w[i] := \Delta x_0[i] \,||\, \Delta x_1[i] \,||\, \Delta y[i]$ for $i = 0, 1, \ldots, n-1$.

Proof: See [27, Theorem 4]. Note that, our A_w, L, C are the transposes of the A_w, L, C in Lipmaa et al. (2004), hence the order of the product is reversed.□

Now, we express the XDP of the half quarter round in terms of the XDPs of the modular additions.

Lemma 4: For all $\Delta x, \Delta y \in \mathbb{W}_n^4$, we have

$$\mathrm{xdp}^{\mathrm{HQR}_{r1,r2}}(\Delta x \to \Delta y) \neq 0$$

only if

$$\Delta y_0 \oplus \Delta x_3 = \Delta y_3 \ggg r_1 \tag{3}$$

$$\Delta y_2 \oplus \Delta x_1 = \Delta y_1 \ggg r_2. \tag{4}$$

In such a case

$$\begin{aligned}\mathrm{xdp}^{\mathrm{HQR}_{r1,r2}}(\Delta x \to \Delta y) = {} & \mathrm{xdp}^{\boxplus}(\Delta x_0, \Delta x_1 \to \Delta y_0) \\ & \cdot \mathrm{xdp}^{\boxplus}(\Delta y_3, \Delta x_2 \to \Delta y_2).\end{aligned}$$

Proof: By the definition of HQR_{r_1,r_2}, we have that

$$\mathrm{HQR}_{r_1,r_2}(\mathbf{x} \oplus \Delta\mathbf{x}) = \mathrm{HQR}_{r_1,r_2}(\mathbf{y}) \oplus \Delta\mathbf{y}$$

is equivalent to

$$(x_0 \oplus \Delta x_0) \boxplus (x_1 \oplus \Delta x_1) = (x_0 \boxplus x_1) \oplus \Delta y_0 \tag{5}$$

$$\big((y_0 \oplus \Delta y_0) \oplus (x_3 \oplus \Delta x_3)\big) \lll r_1 = \big((y_0 \oplus x_3) \lll r_1\big) \oplus \Delta y_3 \tag{6}$$

$$(y_3 \oplus \Delta y_3) \boxplus (x_2 \oplus \Delta x_2) = (y_3 \boxplus x_2) \oplus \Delta y_2 \tag{7}$$

$$\big((y_2 \oplus \Delta y_2) \oplus (x_1 \oplus \Delta x_1)\big) \lll r_2 = \big((y_2 \oplus x_1) \lll r_2\big) \oplus \Delta y_1. \tag{8}$$

Equations (6) and (8) simplify at once to (3) and (4), respectively, which do not depend on x and y. Therefore, they are necessary conditions for the XDP to be non-zero.

Since the map $\mathbb{W}_n^4 \to \mathbb{W}_n^4 : \mathbf{x} \mapsto \mathbf{z}$ given by

$$\begin{aligned} z_0 &= x_0 \\ z_1 &= x_1 \\ z_2 &= x_2 \\ z_3 &= \big((x_0 \boxplus x_1) \oplus x_3\big) \lll r_1 \end{aligned}$$

is a bijection, we can make the change of variable $\mathbf{x} \mapsto \mathbf{z}$ without changing the XDP, and equations (5) and (7) turn into

$$(z_0 \oplus \Delta x_0) \boxplus (z_1 \oplus \Delta x_1) = (z_0 \boxplus z_1) \oplus \Delta y_0 \tag{9}$$

$$(z_2 \oplus \Delta y_3) \boxplus (z_3 \oplus \Delta x_2) = (z_2 \boxplus z_3) \oplus \Delta y_2. \tag{10}$$

Note that (9) and (10) are independent, since the first is an equation in z_0, z_1 while the second is an equation in z_2, z_3. The claim follows.□

At this point, using Lemma 3 and Lemma 4, the XDP of ChaCha half quarter round can be computed in time $O(n)$.

3.2 XDP of ChaCha quarter round

The next lemma provides a heuristic formula for the XDP of ChaCha quarter round, under the assumption that the two half quarter rounds are 'independent'.

Lemma 5: Assuming that HQR_{r_1,r_2}, and HQR_{r_3,r_4} are 'independent' (see Definition 5), for every $\Delta\mathbf{x}, \Delta\mathbf{z} \in \mathbb{W}_n^4$ we have

$$\begin{aligned}\mathrm{xdp}^{\mathrm{HQR}}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = {} & \mathrm{xdp}^{\boxplus}(\Delta x_0, \Delta x_1 \to \Delta y_0) \\ & \cdot \mathrm{xdp}^{\boxplus}(\Delta y_3, \Delta x_2 \to \Delta y_2) \\ & \cdot \mathrm{xdp}^{\boxplus}(\Delta y_0, \Delta y_1 \to \Delta z_0) \\ & \cdot \mathrm{xdp}^{\boxplus}(\Delta z_3, \Delta y_2 \to \Delta z_2),\end{aligned} \tag{11}$$

where $\Delta\mathbf{y} \in \mathbb{W}_n^4$ is given by

$$\begin{aligned} \Delta y_0 &= \Delta x_3 \oplus (\Delta y_3 \ggg r_1) \\ \Delta y_1 &= \Delta z_2 \oplus (\Delta z_1 \ggg r_4) \\ \Delta y_2 &= \Delta x_1 \oplus (\Delta y_1 \ggg r_2) \\ \Delta y_3 &= \Delta z_0 \oplus (\Delta z_3 \ggg r_3). \end{aligned} \tag{12}$$

Proof: Since $\mathrm{QR} = \mathrm{HQR}_{r_3,r_4} \circ \mathrm{HQR}_{r_1,r_2}$, by the assumption that HQR_{r_3,r_4} and HQR_{r_1,r_2} are 'independent' we have

$$\begin{aligned}\mathrm{xdp}^{\mathrm{QR}}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = {} & \sum_{\Delta\mathbf{y} \in \mathbb{W}_n^k} \mathrm{xdp}^{\mathrm{HQR}_{r1,r2}}(\Delta\mathbf{x} \to \Delta\mathbf{y}) \\ & \mathrm{xdp}^{\mathrm{HQR}_{r3,r4}}(\Delta\mathbf{y} \to \Delta\mathbf{z}).\end{aligned} \tag{13}$$

Furthermore, from the first part of Lemma 4, we get that the addend of (13) is non-zero only if it holds the following system of equations

$$\Delta y_0 \oplus \Delta x_3 = \Delta y_3 \ggg r_1$$
$$\Delta y_2 \oplus \Delta x_1 = \Delta y_1 \ggg r_2$$
$$\Delta z_0 \oplus \Delta y_3 = \Delta z_3 \ggg r_1$$
$$\Delta z_2 \oplus \Delta y_1 = \Delta z_1 \ggg r_2,$$

which solved gives a unique value of Δy by (12). Then the claim follows from the second part of Lemma 4.□

For small word sizes n = 5, 6, 7, 8, and for a random sample of $\Delta\mathbf{x}$'s and $\Delta\mathbf{y}$'s, we compared the values of the XDP of the quarter round (with $r_1 = 4$, $r_2 = 3$, $r_3 = 2$, $r_4 = 1$) given by the heuristic formula of Lemma 5 with the exact values computed by brute force. Actually, since the XDP is zero for most of the choices of $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$, we generated $\Delta\mathbf{x}$ randomly, then we generated a random $\mathbf{x} \in \mathbb{W}_n^4$ and we picked $\Delta\mathbf{y} = \mathrm{QR}(\mathbf{x} \oplus a\Delta\mathbf{x}) \oplus \mathrm{QR}(\mathbf{x})$, which guarantees that the XDP is non-zero. We collect the results in Table 2 and Figure 5, which shows the distribution of L = log(exact value of $\mathrm{xdp}^{\mathrm{HQR}}$/ heuristic value of $\mathrm{xdp}^{\mathrm{HQR}}$), given by Lemma 5 as the input/output differences ranges over our 16,000 samples. For example, the top left graph shows that slightly less than 5,000 input/output differences have L in [–1, –0.5]. Notice that the reason for the deviation is due to the lack of independence of HQR_{r_1,r_2} and HQR_{r_3,r_4}.

Table 2 E is the average factor which the heuristic formula of Lemma 5 is off from the exact XDP, and σ is the standard deviation (sample size N = 16,000)

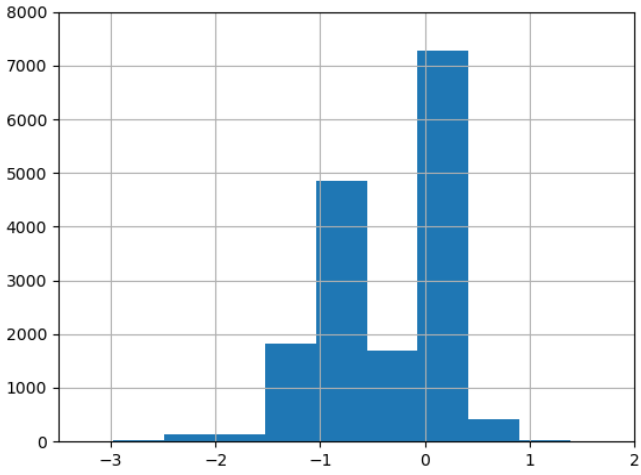
n	5	6	7	8
E	0.67	0.63	0.60	0.57
σ	0.52	0.56	0.61	0.6

3.3 *Maximising the XDP of the half quarter round*

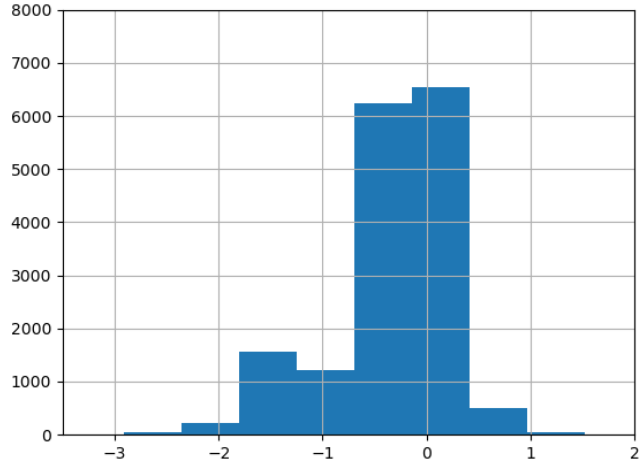
We now illustrate an algorithm, based on the previous results, that takes as input $\Delta\mathbf{x} \in \mathbb{W}_n^4$ and returns as output $\Delta\mathbf{y} \in \mathbb{W}_n^4$ such that $\mathrm{xdp}^{\mathrm{HQR}_{r1,r2}}(\Delta\mathbf{x} \to \Delta\mathbf{y})$ is high. Assuming the independence of the half quarter rounds, this algorithm can then be applied multiple times to obtain high XDPs for the quarter round, or even iterated of the quarter round up to a full round.

First, in light of Lemma 3, we have Algorithm 1, which is a greedy algorithm that takes as input $\Delta x_0, \Delta x_1 \in \mathbb{W}_n$ and $c \in \mathbb{W}_n$, and returns as output $\Delta y \in \mathbb{W}_n$ and p such that $p = \mathrm{xdp}^{\boxplus}(\Delta x_0, \Delta x_1 \to \Delta y)$ is high. If the parameter c is changed, then a different Δy is returned ($c = 0$ means that Δy is obtained in a completely greedy way). This is to avoid being trapped in a local maximum.

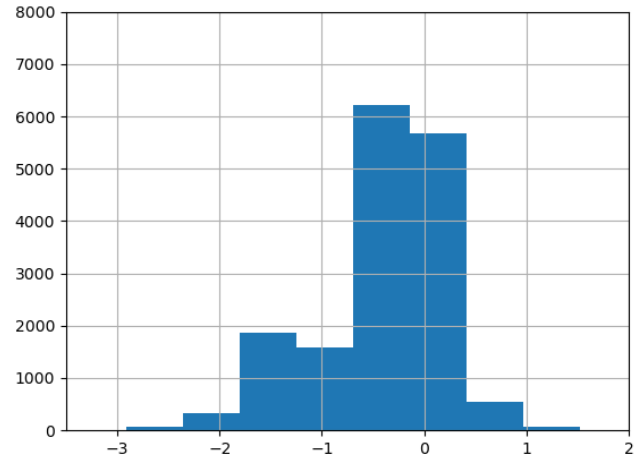
Figure 5 Distribution of the logarithm of the ratio between the XDP given by Lemma 5 and the correct value of the XDP, for (a) n = 5, (b) n = 6, (c) n = 7, (d) n = 8 (see online version for colours)



(a)

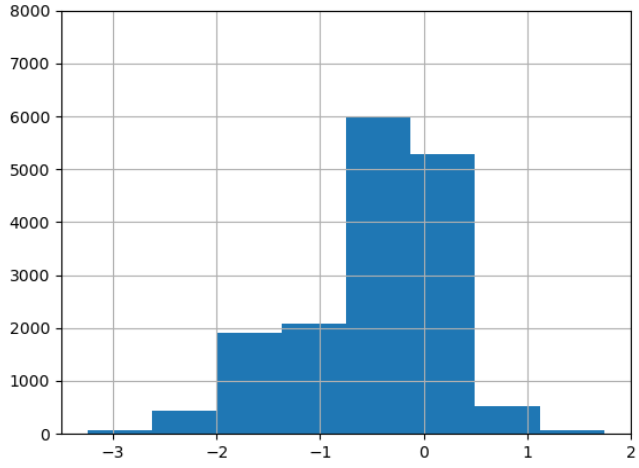


(b)



(c)

Figure 5 Distribution of the logarithm of the ratio between the XDP given by Lemma 5 and the correct value of the XDP, for (a) $n = 5$, (b) $n = 6$, (c) $n = 7$, (d) $n = 8$ (continued) (see online version for colours)



(d)

Then, according to Lemma 4, to obtain a high XDP for the half quarter round two calls to Algorithm 1 are sufficient, as done in Algorithm 2. Note that Algorithm 2 has two parameters c_0, c_1 that when changed give different values of $\Delta\mathbf{y}$.

Algorithm 1:

Function Greedy_XDP_Add(Δx_0, Δx_1, c):
- $\Delta y \leftarrow 0$ (n bits word)
- $B \leftarrow 4 \times 4$ identity matrix
- $p \leftarrow 1$
- **for** $i = 0, 1, \ldots, n-1$ **do**
  - $B_0 \leftarrow BA_{\Delta x_0[i]\|\Delta x_1[i]\|0}$
  - $B_1 \leftarrow BA_{\Delta x_0[i]\|\Delta x_1[i]\|1}$
  - $p_0 = 4^{-(i+1)}LB_0C$
  - $p_1 = 4^{-(i+1)}LB_1C$
  - **if** ($p_0 \geq p_1$ and $c[i] = 0$) or ($p_0 < p_1$ and $c[i] = 1$) **then**
    - $\Delta y[i] \leftarrow 0$
    - $B \leftarrow B_0$
    - $p \leftarrow p_0$
  - **else**
    - $\Delta y[i] \leftarrow 1$
    - $B \leftarrow B_1$
    - $p \leftarrow p_1$
  - **return** Δy, p

Algorithm 2:

Function Greedy_XDP_HQR(r_1, r_2, Δx_0, Δx_1, Δx_2, Δx_3, c_0, c_1):
- Δy_0, $p_0 \leftarrow$ Greedy_XDP_Add (Δx_0, Δx_1, c_0)
- $\Delta y_3 \leftarrow (\Delta x_3 \oplus \Delta y_0) \lll r_1$
- Δy_2, $p_1 \leftarrow$ Greedy_XDP_Add (Δy_3, Δx_2, c_1)
- $\Delta y_1 \leftarrow (\Delta x_1 \oplus \Delta y_2) \lll r_2$
- $p \leftarrow p_0 p_1$
- **return** Δy_0, Δy_1, Δy_2, Δy_3, p

4 Additive differential probability of ChaCha round

In this section we illustrate the results on the ADP analog to the results on the XDP of the previous section.

4.1 ADP of ChaCha half quarter round

Here, we give a formula for the additive differential probability of the half quarter round of ChaCha.

First, let us define the matrices that are going to be used in the next lemma. These are

$$A_{000} = \begin{pmatrix} 4&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&1&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&1&0&0&1&1&0&0\\ 1&0&1&0&1&0&1&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix} \quad A_{001} = \begin{pmatrix} 0&0&0&0&0&0&0&0\\ 0&4&0&0&0&0&0&0\\ 1&1&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&1&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&1&0&1&0&1&0&1 \end{pmatrix}$$

$$A_{010} = \begin{pmatrix} 0&0&0&0&0&0&0&0\\ 1&1&1&1&0&0&0&0\\ 0&0&4&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&0&1&0&1&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&1&1&0&0&1&1 \end{pmatrix} \quad A_{011} = \begin{pmatrix} 1&1&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&4&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&1&0&1&0&1&0&1\\ 0&0&1&1&0&0&1&1\\ 0&0&0&0&0&0&0&0 \end{pmatrix}$$

$$A_{100} = \begin{pmatrix} 0&0&0&0&0&0&0&0\\ 1&1&0&0&1&1&0&0\\ 1&0&1&0&1&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&4&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&1&1 \end{pmatrix} \quad A_{101} = \begin{pmatrix} 1&1&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&1&0&1&0&1&0&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&4&0&0\\ 0&0&0&0&1&1&1&1\\ 0&0&0&0&0&0&0&0 \end{pmatrix}$$

$$A_{110}=\begin{pmatrix}1&0&1&0&1&0&1&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&1&1&0&0&1&1\\0&0&0&0&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&0&0&4&0\\0&0&0&0&0&0&0&0\end{pmatrix}\quad A_{111}=\begin{pmatrix}0&0&0&0&0&0&0&0\\0&1&0&1&0&1&0&1\\0&0&1&1&0&0&1&1\\0&0&0&0&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&4\end{pmatrix}$$

$$R=\begin{pmatrix}1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&1&0\end{pmatrix}$$

$$L_0=(1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)\quad L_1=(0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)$$

$$C_0=(1\ 0\ 1\ 0\ 1\ 0\ 1\ 0)^T\quad C_1=(0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)^T$$

Lemma 6: Let A_w, L_i, C_i, and R be the matrices defined as above. For all constants $a_0, a_1, a_{01} \in \mathbb{W}_n$ and every integer $r \in [0, n)$, the number of solutions $(x_0, x_1) \in \mathbb{W}_n^2$ of the equation

$$(x_0 \boxplus a_0) \oplus (x_1 \boxplus a_1) = (x_0 \boxplus x_1) \overleftarrow{\boxplus}^r a_{01} \tag{14}$$

is equal to

$$\sum_{i\in\{0,1\}} L_i A_{w[0]} A_{w[1]} \cdots A_{w[n-r-1]} R A_{w[n-r]} \cdots A_{w[n-1]} C_i,$$

where $w[i] := a_0[i] \,\|\, a_1[i] \,\|\, a_{01}[i]$ for $i = 0, 1, \ldots, n-1$.

Proof: The result follows from the theory developed in Subsection 2.4. First, we consider

$$y = (x_0 \boxplus a_0) \oplus (x_1 \boxplus a_1) \oplus \big((x_0 \boxplus x_1) \boxplus a_{01}\big), \tag{15}$$

Noticing that $y = 0$ gives (14) with $\overleftarrow{\boxplus}^r$ replaced by $\boxplus$. We represent the states of the S-function associated to (15) by the 3-bits words $c_0 \,\|c_1\|\, c_{01}$, where c_0, c_1, c_{01} are the carries in the first, second, and third addition of (15), respectively. We identify each state $c_0 \,\|c_1\|\, c_{01}$ with the corresponding 3-bit integer $4c_0 + 2c_1 + c_{01}$. The S-function for (15) is defined by the recurrences

$$\begin{aligned} y[i] &\leftarrow (x_0[i] \oplus a_0[i] \oplus c_0) \oplus (x_i[i] \oplus a_i[i] \oplus c_1) \\ &\quad \oplus (x_0[i] \oplus x_1[i] \oplus a_{01}[i] \oplus c_{01}), \\ c_0 &\leftarrow \lfloor x_0[i] + a_0[i] + c_0 / 2 \rfloor, \\ c_1 &\leftarrow \lfloor x_1[i] + a_1[i] + c_1 / 2 \rfloor, \\ c_{01} &\leftarrow \lfloor \big((x_0[i] \oplus x_1[i]) + a_{01}[i] + c_{01}\big)/2 \rfloor, \end{aligned}$$

The first of which can be simplified to $y[i] \leftarrow a_0[i] \oplus a_1[i] \oplus a_{01}[i] \oplus c_0 \oplus c_1 \oplus c_{01}$. By Theorem 1, the number of $(x_0, x_1) \in \mathbb{W}_n^2$ such that $y = 0$ is equal to

$$L A_{w[0]} A_{w[1]} \cdots A_{w[n-1]} C$$

where L:= (1, 0, …, 0) and $C = (1, 1, ..., 1)^\top$ are vectors of length 8 (the number of states), and the (i, j) entry of $A_{\alpha_0 \| \alpha_1 \| \alpha_{01}}$ is equal to the number of $(\chi_0, \chi_1) \in \mathbb{W}_2^2$ such that

$$\alpha_0 \oplus \alpha_1 \oplus \alpha_{01} \oplus c_0 \oplus c_1 \oplus c_{01} = 0, c_0' = \lfloor (\chi_0 + \alpha_0 + c_0)/2 \rfloor,$$

$$c_1' = \lfloor (\chi_1 + \alpha_1 + c_1)/2 \rfloor, c_{01}' = \lfloor ((\chi_0 \oplus \chi_1) + \alpha_{01} + c_{01})/2 \rfloor,$$

with $i = 4c_0 + 2c_1 + c_{01}$ and $i = 4c_0' + 2c_1' + c_{01}'$. Finally, in light of Remark 3 and Subsection 2.5, we introduce the matrices R, L_i, and C_i to handle $\overleftarrow{\boxplus}^r$: the projection matrix R has the purpose to not propagate the carry c_{01} of the $(n - r)^{\text{th}}$ digit; and L_i, C_i have the purpose of counting only the $(x_0, x_1) \in \mathbb{W}_n$ such that the initial and final state have the same c_{01}.□

Now define $J_r(x_0, x_1) := (x_0 \oplus x_1) \lll r$ for every integer $r \in [0, n)$ and every $x_0, x_1 \in \mathbb{W}_n$.

Lemma 7: Let A_w, L_i, C_i, and R be the matrices of Lemma 6. For all $\Delta\mathbf{x} \in \mathbb{W}_n^2$, $\Delta\mathbf{y} \in \mathbb{W}_n$, and every integer $r \in [0, n)$, we have

$$\begin{aligned} \mathrm{adp}^{J_r}(\Delta x \to \Delta y) = 4^{-n} \sum_{i\in\{0,1\}} & L_i A_{w[0]} A_{w[1]} \cdots A_{w[n-r-1]} R A_{w[n-r]} \\ & \cdots A_{w[n-1]} C_i, \end{aligned}$$

where $w[i] := \Delta x_0[i] \| \Delta x_1[i] \| (\Delta y \ggg r)[i]$ for $i = 0, 1, \ldots, n-1$.

Proof: Noting that $J_r(x_0 \boxplus \Delta x_0, x_1 \boxplus \Delta x_1) = J_r(x_0, x_1) \boxplus \Delta y$ is equivalent to $(x_0 \boxplus \Delta x_0) \oplus (x_1 \boxplus \Delta x_1) = (x_0 \oplus x_1) \overleftarrow{\boxplus}^r (\Delta y \ggg r)$, the claim follows immediately from Lemma 6. □

Lemma 8: For all $\Delta\mathbf{x}, \Delta\mathbf{y} \in \mathbb{W}_n^4$, we have

$$\mathrm{adp}^{\mathrm{HQR}_{r_1,r_2}}(\Delta x \to \Delta y) \neq 0$$

Only if

$$\Delta x_0 \boxplus \Delta x_1 = \Delta y_0 \tag{16}$$

$$\Delta y_3 \boxplus \Delta x_2 = \Delta y_2. \tag{17}$$

In such a case

$$\begin{aligned} \mathrm{adp}^{\mathrm{HQR}_{r_1,r_2}}(\Delta\mathbf{x} \to \Delta\mathbf{y}) = \mathrm{adp}^{J_{r_1}}(\Delta y_0, \Delta x_3 \to \Delta y_3) \\ \cdot \mathrm{adp}^{J_{r_2}}(\Delta x_1, \Delta y_2 \to \Delta y_1). \end{aligned} \tag{18}$$

Proof: By the definition of HQR_{r_1,r_2}, we have that

$$\mathrm{HQR}_{r_1,r_2}(\mathbf{x} \boxplus \Delta\mathbf{x}) = \mathrm{HQR}_{r_1,r_2}(\mathbf{y}) \boxplus \Delta\mathbf{y}$$

is equivalent to

$$(x_0 \boxplus \Delta x_0) \boxplus (x_1 \boxplus \Delta x_1) = (x_0 \boxplus x_1) \boxplus \Delta y_0 \quad (19)$$

$$\big((y_0 \boxplus \Delta y_0) \oplus (x_3 \boxplus \Delta x_3)\big) \lll r_1 = \big((y_0 \oplus x_3) \lll r_1\big) \boxplus \Delta y_3 \quad (20)$$

$$(y_3 \boxplus \Delta y_3) \boxplus (x_2 \boxplus \Delta x_2) = (y_3 \boxplus x_2) \boxplus \Delta y_2 \quad (21)$$

$$\big((y_2 \boxplus \Delta y_2) \oplus (x_1 \boxplus \Delta x_1)\big) \lll r_2 = \big((y_2 \oplus x_1) \lll r_2\big) \boxplus \Delta y_1. \quad (22)$$

Equations (19) and (21) simplify at once to (16) and (17), respectively, which do not depend on x and y. Therefore, they are necessary conditions for the adp to be non-zero.

Since the map $\mathbb{W}_n^4 \to \mathbb{W}_n^4 : \mathbf{x} \mapsto \mathbf{z}$ given by

$$z_0 = x_0 \boxplus x_1,\ z_1 = x_1,\ z_2 = \big(\big((x_0 \boxplus x_1) \oplus x_3\big) \ggg r_1\big) \boxplus x_2,\ z_3 = x_3$$

is a bijection, we can make the change of variable $\mathbf{x} \mapsto \mathbf{z}$ without changing the adp, and equations (20) and (22) turn into

$$\big((z_0 \boxplus \Delta y_0) \oplus (z_3 \boxplus \Delta x_3)\big) \lll r_1 = \big((z_0 \oplus z_3) \lll r_1\big) \boxplus \Delta y_3 \quad (23)$$

$$\big((z_1 \boxplus \Delta x_1) \oplus (z_2 \boxplus \Delta y_2)\big) \lll r_2 = \big((z_1 \oplus z_2) \lll r_2\big) \boxplus \Delta y_1. \quad (24)$$

Note that (23) and (24) are independent, since the first is an equation in z_0, z_3 while the second is an equation in z_1, z_2. Moreover, they can be rewritten as:

$$J_{r_1}\big((z_0, z_3) \boxplus (\Delta y_0, \Delta x_3)\big) = J_{r_1}(z_0, z_3) \boxplus \Delta y_3$$

$$J_{r_2}\big((z_1, z_2) \boxplus (\Delta x_1, \Delta y_2)\big) = J_{r_2}(z_1, z_2) \boxplus \Delta y_1,$$

which are the equations in the ADPs of J_{r_1} and J_{r_2} and the claim follows.□

At this point, using Lemma 7 and Lemma 8, the ADP of ChaCha half quarter round can be computed in time $O(n)$.

4.2 ADP of ChaCha quarter round

The next lemma provides a heuristic formula for the ADP of ChaCha quarter round, under the assumption that the two half quarter rounds are 'independent'.

Lemma 9: Assuming that HQR_{r_1,r_2} and HQR_{r_2,r_3} are 'independent' (see Definition 5), for every $\Delta\mathbf{x}, \Delta\mathbf{z} \in \mathbb{W}_n^4$ we have

$$\begin{aligned} \mathrm{adp}^{\mathrm{QR}}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = {} & \mathrm{adp}^{J_{r_1}}(\Delta y_0, \Delta x_3 \to \Delta y_3) \\ & \cdot \mathrm{adp}^{J_{r_2}}(\Delta x_1, \Delta y_2 \to \Delta y_1) \\ & \cdot \mathrm{adp}^{J_{r_3}}(\Delta z_0, \Delta y_3 \to \Delta z_3) \\ & \cdot \mathrm{adp}^{J_{r_4}}(\Delta y_1, \Delta z_2 \to \Delta z_1) \end{aligned} \quad (25)$$

where $\Delta\mathbf{y} \in \mathbb{W}_n^4$ is given by

$$\begin{aligned} \Delta y_0 &= \Delta x_0 \boxplus \Delta x_1 \\ \Delta y_1 &= \Delta z_0 \boxminus \Delta x_0 \boxminus \Delta x_1 \\ \Delta y_2 &= \Delta z_2 \boxminus \Delta z_3 \\ \Delta y_3 &= \Delta z_2 \boxminus \Delta z_3 \boxminus \Delta x_2. \end{aligned} \quad (26)$$

Proof: Since $\mathrm{QR} = \mathrm{HQR}_{r_3,r_4} \circ \mathrm{HQR}_{r_1,r_2}$, by the assumption that HQR_{r_3,r_4} and HQR_{r_1,r_2} are 'independent', we have

$$\mathrm{adp}^{\mathrm{QR}}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = \sum_{\Delta\mathbf{y} \in \mathbb{W}_n^4} \mathrm{adp}^{\mathrm{HQR}_{r_1,r_2}}(\Delta\mathbf{x} \to \Delta\mathbf{y}) \mathrm{adp}^{\mathrm{HQR}_{r_3,r_4}}(\Delta\mathbf{y} \to \Delta\mathbf{z}). \quad (27)$$

Furthermore, from the first part of Lemma 8, we get that the addend of (27) is non-zero only if it holds the following system of equations

$$\begin{aligned} &\Delta x_0 \boxplus \Delta x_1 = \Delta y_0,\ \Delta y_3 \boxplus \Delta x_2 = \Delta y_2, \\ &\quad \Delta y_0 \boxplus \Delta y_1 = \Delta z_0,\ \Delta z_3 \boxplus \Delta y_2 = \Delta z_2, \end{aligned}$$

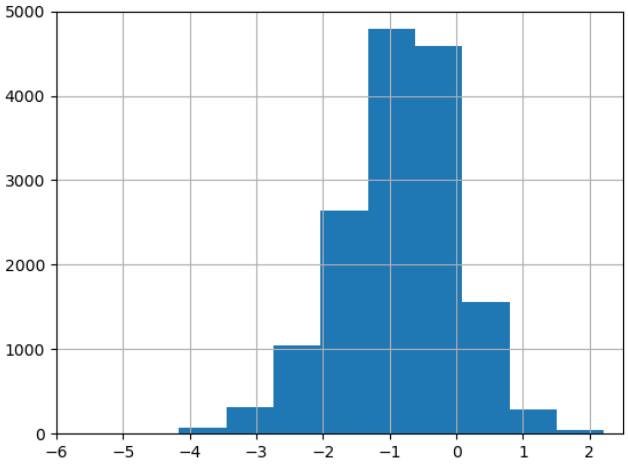
which solved gives a unique value of Δy by (26). Then the claim follows from the second part of Lemma 8.□

Table 3 E is the average factor which the heuristic formula of Lemma 9 is off from the exact ADP, and σ is the standard deviation (sample size N = 15,000)

n	5	6	7	8
E	0.43	0.31	0.31	0.27
σ	0.85	0.97	1.02	1.11

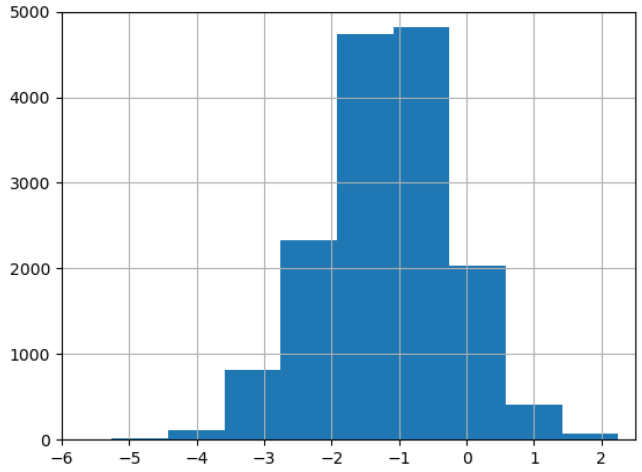
For small word sizes n = 5, 6, 7, 8, and for a random sample of $\Delta\mathbf{x}$'s and $\Delta\mathbf{y}$'s, we compared the values of the ADP of the quarter round (with $r_1 = 4$, $r_2 = 3$, $r_3 = 2$, $r_4 = 1$) given by the heuristic formula of Lemma 9 with the exact values computed by brute force. Actually, since the ADP is zero for most of the choices of $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$, we generated $\Delta\mathbf{x}$ randomly, then we generated a random $\Delta\mathbf{x} \in \mathbb{W}_n^4$ and we picked $\Delta\mathbf{y} = \mathrm{HQR}(\mathbf{x} \boxplus \Delta\mathbf{x}) \boxplus \mathrm{HQR}(\mathbf{x})$, which guarantees that the ADP is non-zero. We collect the results in Table 3 and Figure 6.

Figure 6 Distribution of the logarithm of the ratio between the ADP given by Lemma 9 and the correct value of the ADP, for (a) n = 5, (b) n = 6, (c) n = 7, (d) n = 8 (see online version for colours)

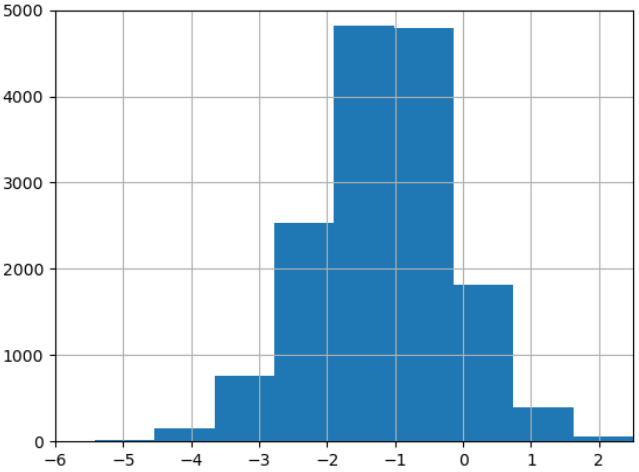


(a)

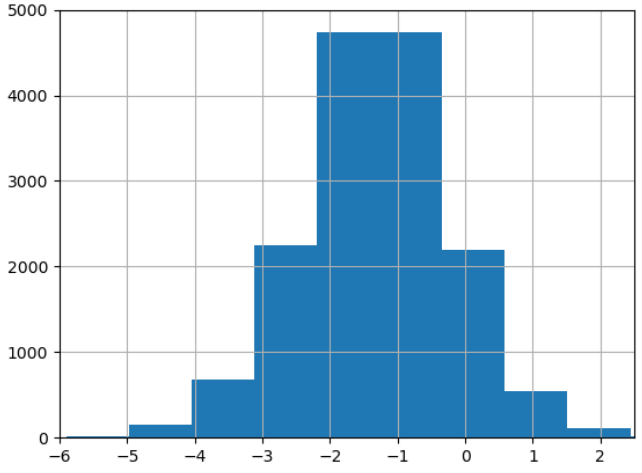
Figure 6 Distribution of the logarithm of the ratio between the ADP given by Lemma 9 and the correct value of the ADP, for (a) $n = 5$, (b) $n = 6$, (c) $n = 7$, (d) $n = 8$ (continued) (see online version for colours)



(b)



(c)



(d)

4.3 Maximising the ADP of the half quarter round

Now, we illustrate an algorithm, based on the previous results, that given as input $\Delta\mathbf{y} \in \mathbb{W}_n^4$ returns as output $\Delta\mathbf{x} \in \mathbb{W}_n^4$ such that $\mathrm{adp}^{\mathrm{HQR}_{n,r2}}(\Delta\mathbf{x} \to \Delta\mathbf{y})$ is high. Note that this works backward (it takes as input $\Delta\mathbf{y}$ and returns $\Delta\mathbf{x}$) respect to the algorithm given in Subsection 3.3 to maximise the XDP. Assuming the independence of the half quarter rounds, this algorithm can then be applied multiple times to obtain high ADPs for the quarter round, or even iterated of the quarter round up to a full round.

First, in light of Lemma 7, we have Algorithm 3, which is a greedy algorithm that takes as input $\Delta x_1, \Delta y \in \mathbb{W}_n$ and $c \in \mathbb{W}_n$ and returns as output $\Delta x_0 \in \mathbb{W}_n$ and p such that $p = \mathrm{adp}^{J_r}(\Delta x_0, \Delta x_1 \to \Delta y)$ is high.

Then, according to Lemma 8, to obtain a high ADP for the half quarter round two calls to Algorithm 3 are sufficient, as done in Algorithm 4. Note that Algorithm 4 has two parameters c_0, c_1 that when changed give different values of $\Delta\mathbf{x}$.

Remark 4: Taking as input $\Delta\mathbf{y}$ and returning as output $\Delta\mathbf{x}$, instead of the contrary, might seem unnatural. We do so because, once $\Delta\mathbf{y}$ is fixed, the two factors of the product for the ADP of half quarter round (18) are independent functions of Δx_1 and Δx_3 and consequently they can be independently maximised using Algorithm 3. On the contrary, if $\Delta\mathbf{x}$ is fixed, the two factors are not independent functions of Δy_0, Δy_3 and Δy_1, Δy_2, because equations (16) and (17) have to be taken into account, and consequently they cannot be independently maximised.

Algorithm 3:

```
1  Function Greedy_ADP_J(r, Δx1, c):
2      Δx0 ← 0 (n bits word)
3      B ← 8 × 8 identity matrix
4      p ← 1
5      for i = 0, 1, …, n − 1 do
6          B0 ← BA0‖Δx1[i]‖(Δy⋙r)[i]
7          B1 ← BA1‖Δx1[i]‖(Δy⋙r)[i]
8          if i = n − r − 1 then
9              B0 ← B0R
10             B1 ← B1R
11         end
12         p0 = 4^−(i+1)(L0B0C0 + L1B0C1)
13         p1 = 4^−(i+1)(L0B1C0 + L1B1C1)
14         if (p0 ≥ p1 and c[i] = 0) or (p0 < p1 and c[i] = 1) then
15             Δx0[i] ← 0
16             B ← B0
17             p ← p0
18         else
19             Δx0[i] ← 1
20             B ← B1
21             p ← p1
22         end
23     end
24     return Δx0, p
```

Algorithm 4:

Function Greedy_ADP_HQR(r_1, r_2, Δy_0, Δy_1, Δy_2, Δy_3, c_0, c_1):
  $\Delta x_3, p_0 \leftarrow$ Greedy_ADP_J (r_1, Δy_0, Δy_3, c_0)
  $\Delta x_1, p_1 \leftarrow$ Greedy_ADP_J (r_2, Δy_2, Δy_1, c_1)
  $\Delta x_0 \leftarrow \Delta y_0 \boxminus \Delta x_1$
  $\Delta x_2 \leftarrow \Delta y_2 \boxminus \Delta y_3$
  $p \leftarrow p_0 p_1$
  return Δx_0, Δx_1, Δy_2, Δx_2, p

5 Experimental results

In this section, we provide some experimental results. In particular, we show the best differential characteristics for ChaCha half quarter round that can be found with our method. We also provide explicit differential trails and their corresponding probability for ChaCha permutation.

We first show how close Algorithm 2 is to find a characteristic with maximum probability. For each 2^{4w} input difference, we computed the maximum xdp by checking through all possible output differences and compared it to the value returned by Algorithm 2. The complexity of this approach for the xdp is 2^{8w} and hence we were able to run this experiment only for 4-bit words. We repeated the same experiment for the adp. To compute the xdp we used the result from Lemma 4, while for the adp we used Lemma 8. In Table 4, we report the results of the experiment. The row '#matches' reports the number of times that the greedy strategy returns the actual best probability, while the '%matches' is #matches·100/2^{4w}. In the xdp case, the greedy algorithm returns the best probability about 16% of the time, and a probability within 50%–40% of the best probability about 35% of the times; so that it returns a good probability (either the maximum or half of it) half of the times. This shows that there might be room for improvement for the greedy strategy we adopted. In the adp case, the best probability is returned more than half of the times. So the greedy strategy seems to be more effective in the adp case.

We recall that ChaCha state is a 16 32-bit word vector (s_0, …, s_{15}), where s_0, …, s_3 are initialised with constant values, s_4, …, s_{11} store a 256 bit key, and s_{12}, …, s_{15} are used for the counter and the nonce. Thus, in a single key differential attack, only these last four words can be used to inject a difference. Furthermore, the first application of the four parallel quarter rounds receives as input the words s_{0+i}, s_{4+i}, s_{8+i}, s_{12+i}, with i = 0, 1, 2, 3. Thus, the attacker can only inject the difference in the last of these four words. Given these constrains, the differential characteristic with highest probability returned by Algorithm 2 is 0 × 00000000 00000000 00000000 00008000 → 0 × 00000800 04044040 80080080 00080080, holding with probability 2^{-3}.

In Table 5, we report the maximum, minimum and average characteristic probability for half and full quarter round, found using Algorithm 2 and Algorithm 4.

The probability is computed over a sample of 2^{16} 128-bit random input differences.

In Table 6 and Table 7, we report, respectively, a XOR and an additive differential trail covering four rounds. The trails were obtained in few milliseconds of computing time. These trails could be used for example in differential-linear attacks such as the one in Beierle et al. (2020). In this work, the authors use differentials with input difference (0, 0, 0, x) and probability 2^{-5} on average. Our method provides an elegant and automatic way of finding such differentials. Furthermore, the attack in Beierle et al. (2020) exploits a differential characteristic for one round only. Using our method, one extra round could be added to the attack, at the expense of decreasing the trail probability.

Notice that, contrary to Algorithm 2, Algorithm 4 cannot be used as is to mount a single key differential attack[2], since the additive differential trail is found by searching for the input differences given an output difference. This makes it difficult to find an input difference which is 0 in the input words s_0, …, s_{11}. Thus, Algorithm 4 can only be of use for an attacker if combined with other techniques. Notice also that in Table 7, the three rounds differential trail from round 2 to round 4 has probability $2^{-98.4}$, which is much lower than any other three rounds differential trail we were able to find for XOR differences with Algorithm 2.

Last, we also notice that it is easy to build mixed (additive-XOR) trails, since one could easily match the output of the additive trail with the input of the XOR one. For example, if we take round 3 and 4 of Table 7 and use the output difference as the input difference to Algorithm 2, we obtain a four round trail of probability $2^{-(19+2+5+43)} = 2^{-69}$, and if we started from round 2 of Table 7, the corresponding five round trail would have had probability $2^{-(77.4+19+2+5+43)} = 2^{-146.4}$. The only problem with this approach is that the input of the trail cannot be used in the context of ChaCha20 (stream cipher), not even in the related-key scenario, as the initial state is constrained by the constant value of the first row of the state, whose additive difference is always 0.

6 Finding XOR-differential trails with MILP solvers

In this section, we will describe a MILP model to find differential trails for ChaCha internal permutation. We want to compare the performance of MILP against the method presented in Section 3. The techniques used to build our MILP model are presented in Fu et al. (2016), where the model aims at differential trails for Speck.

6.1 MILP-based automatic search

We built our MILP model for the ChaCha permutation, modelling their components using inequalities. In the following paragraphs, we will describe the inequalities used to model each component of the ChaCha stream.

Table 4 Precision of the greedy strategy applied in Algorithm 2 and Algorithm 4 (see Section 5 for explanation)

range xdp	*100%*	*(100%–50%]*	*(50%–40%]*	*(40%–30%]*	*(30%–20%]*	*(20%–10%]*	*(10%–0%]*
#matches	11,040	0	23,472	0	22,656	7,072	1,296
%matches	16.85	0.00	35.82	0.00	34.57	10.79	1.97
range adp	*100%*	*(100%–50%]*	*(50%–40%]*	*(40%–30%]*	*(30%–20%]*	*(20%–10%]*	*(10%–0%]*
#matches	34,344	15,312	3,100	5,372	5,112	2,000	296
%matches	52.41	23.36	4.73	8.19	7.80	3.05	0.45

Table 5 Maximum, minimum and average characteristic probability for half and full quarter round, found using Algorithm 2 and Algorithm 4

	(c0; c1)	*HQR*			*QR*		
		max	*min*	*average*	*max*	*min*	*average*
xdp	(0, 0)	$2^{-29:00}$	$2^{-59:00}$	$2^{-42:09}$	$2^{-58:00}$	$2^{-103:00}$	$2^{-71:46}$
adp	(0, 0)	$2^{-31:11}$	$2^{-51:43}$	$2^{-41:50}$	$2^{-67:10}$	$2^{-101:04}$	$2^{-80:51}$

Table 6 Four rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found iterating Algorithm 2

Round	Δ*x*				Δ*y*				*Probability*	
									Rel.	*Cum.*
1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	08000080	$2^{-7.00}$	$2^{-7.00}$
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00100010		
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000880		
	00000000	00000000	00000000	00800008	00000000	00000000	00000000	00080800		
2	00000000	00000000	00000000	08000080	00808000	88000000	00100010	88000880	$2^{-35.0}$	$2^{-42.0}$
	00000000	00000000	00000000	00100010	00001100	10011001	01111011	20002000		
	00000000	00000000	00000000	00000880	00100010	80088800	00088088	00880880		
	00000000	00000000	00000000	00080800	00880000	00000000	80888000	08088080		
3	00808000	88000000	00100010	88000880	10088001	88091090	00080090	2088A008	$2^{-118.}$	$2^{-160.}$
	00001100	10011001	01111011	20002000	13303033	30011010	23130232	00411150		
	00100010	80088800	00088088	00880880	88910888	00800889	88880880	A8002088		
	00880000	00000000	80888000	08088080	09810880	91980898	19900981	00888800		
4	10088001	88091090	00080090	2088A008	81008890	8802008A	90411049	08A00088	$2^{-189.}$	$2^{-349.}$
	13303033	30011010	23130232	00411150	16074136	12221600	73475523	A333B001		
	88910888	00800889	88880880	A8002088	08400048	08892888	90000888	809080A1		
	09810880	91980898	19900981	00888800	039200A0	98889981	90912809	10991188		

Table 7 Four rounds additive differential trail (of ChaCha internal permutation), with relative and cumulative probability of the differential characteristics found iterating Algorithm 4

Round	Δ*x*				Δ*y*				*Probability*	
									Rel.	*Cum.*
1	11BC469C	222C642C	3306926E	DDF975C0	BFBE7FE0	34F3AE78	FFBEF378	7F7F8000	$2^{-208.}$	$2^{-208.}$
	2DC13904	02464248	08A714E2	21458940	80008000	80410020	C2844104	80000C40		
	A27CE21C	C90A2EF7	FF3E72F8	BE7AB700	77BEF7C0	FF7F0000	7FFF0000	7EFB7700		
	0287A010	28E22301	04222F50	81010100	85001084	C1414040	80000080	C0008000		
2	BFBE7FE0	34F3AE78	FFBEF378	7F7F8000	80000000	FFFBF800	FFBEFFC0	FF800000	$2^{-77.4}$	$2^{-286.}$
	80008000	80410020	C2844104	80000C40	00000000	00040000	80408040	00800000		
	77BEF7C0	FF7F0000	7FFF0000	7EFB7700	80000000	00000000	7FFF8000	FF7FFF80		
	85001084	C1414040	80000080	C0008000	80000080	00000800	80008000	00008000		

Table 7 Four rounds additive differential trail (of ChaCha internal permutation), with relative and cumulative probability of the differential characteristics found iterating Algorithm 4 (continued)

Round	Δx				Δy				*Probability*	
									Rel.	*Cum.*
3	80000000	FFFBF800	FFBEFFC0	FF800000	80000000	00000000	00000000	00000000	$2^{-19.0}$	$2^{-304.}$
	00000000	00040000	80408040	00800000	00000000	80000000	00000000	00000000		
	80000000	00000000	7FFF8000	FF7FFF80	00000000	00000000	7FFF8000	00000000		
	80000080	00000800	80008000	00008000	00000000	00000000	00000000	00800000		
4	80000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$2^{-2.00}$	$2^{-306.}$
	00000000	80000000	00000000	00000000	00000000	00000000	00000000	00000000		
	00000000	00000000	7FFF8000	00000000	00000000	00000000	00000000	00000000		
	00000000	00000000	00000000	00800000	00000000	00000000	00000000	80000000		

Table 8 One rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found using the MILP model presented in Subsection 6.1

Round	Δx				Δy				*Probability*	
									Rel.	*Cum.*
1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000800	$2^{-3.00}$	$2^{-3.00}$
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	04044040		
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	80080080		
	00000000	00000000	00000000	00008000	00000000	00000000	00000000	00080080		

Table 9 Two rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found using the MILP model presented in Subsection 6.1

Round	Δx				Δy				*Probability*	
									Rel.	*Cum.*
1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000800	2^{-4}	$2^{-4.}$
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	04044040		
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	80080080		
	00000000	00000000	00000000	00008000	00000000	00000000	00000000	00080080		
2	00000000	00000000	00000000	80000800	00808000	00000880	00040004	88000880	2^{-33}	$2^{-37.}$
	00000000	00000000	00000000	00040004	44000000	04400440	04444044	02000200		
	00000000	00000000	00000000	88000000	00040004	08880080	80880008	88088000		
	00000000	00000000	00000000	08080000	00088000	00000000	00888080	80880800		

Table 10 Three rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found using the MILP model presented in Subsection 6.1

Round	Δx				Δy				*Probability*	
									Rel.	*Cum.*
1	00000000	00000000	00000000	00000000	80000800	00000000	00000000	00000000	2^{-4}	$2^{-4.}$
	00000000	00000000	00000000	00000000	00040004	00000000	00000000	00000000		
	00000000	00000000	00000000	00000000	88000000	00000000	00000000	00000000		
	00008008	00000000	00000000	00000000	08080000	00000000	00000000	00000000		
2	80000800	00000000	00000000	00000000	88000880	00808000	00000880	00040004	2^{-36}	$2^{-40.}$
	00040004	00000000	00000000	00000000	02000200	43800000	04400440	04444044		
	88000000	00000000	00000000	00000000	88088000	00040004	08870080	80880008		
	08080000	00000000	00000000	00000000	80880800	00088000	00000000	00888080		

Table 10 Three rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found using the MILP model presented in Subsection 6.1 (continued)

Round	Δx				Δy				*Probability*	
									Rel.	*Cum.*
3	88000880	00808000	00000880	00040004	02888208	04004408	14400440	c0000400	2^{-107}	$2^{-147.}$
	02000200	43800000	04400440	04444044	44410005	00021c22	2082280a	04000062		
	88088000	00040004	08870080	80880008	82000a88	00004000	04410450	00484440		
	80880800	00088000	00000000	00888080	08888000	00040004	00000010	c000c800		

Table 11 Four rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found using the MILP model presented in Subsection 6.1

Round	Δx				Δy				*Probability*	
									Rel.	*Cum.*
1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000800	2^{-3}	$2^{-3.}$
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	04044040		
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	80080080		
	00000000	00000000	00000000	00008000	00000000	00000000	00000000	00080080		
2	00000000	00000000	00000000	00000800	00008008	80080800	40400404	00000880	2^{-41}	$2^{-44.}$
	00000000	00000000	00000000	04044040	04404404	00400040	04040004	02202042		
	00000000	00000000	00000000	80080080	c0400404	08088008	80000008	88000000		
	00000000	00000000	00000000	00080080	08080080	00000000	00088008	80800000		
3	00008008	80080800	40400404	00000880	40844404	00480840	80080c00	28020224	2^{-123}	$2^{-167.}$
	04404404	00400040	04040004	02202042	20222020	20042444	02002060	30100305		
	c0400404	08088008	80000008	88000000	04044440	08400848	04400040	20420620		
	08080080	00000000	00088008	80800000	00484c04	08880808	00484c04	40808400		
4	40844404	00480840	80080c00	28020224	e0402040	24042002	400c5814	2a202222	2^{-147}	$2^{-314.}$
	20222020	20042444	02002060	30100305	15250110	20302230	22052143	3e618a08		
	04044440	08400848	04400040	20420620	00009005	022a4a24	20002044	a4000220		
	00484c04	08880808	00484c04	40808400	20220240	01009445	202a4a04	00004004		

Table 12 Best probabilities found after 24 hours of computation with Algorithm 2 and MILP using different number of cores

Strategy \ *Rounds*	*1*		*2*		*3*		*4*	
	p	*t*	*p*	*t*	*p*	*t*	*p*	*t*
Algorithm 2 [single core]	2^{-3}	0.7 s	2^{-37}	1.2 s	2^{-157}	1.9 s	2^{-349}	6.7 s
MILP [single core]	2^{-3}	0.3 s	2^{-37}	3,950 s	2^{-162}	1.2 s	2^{-426}	22,786 s
MILP [56 cores]	2^{-3}	0.3 s	2^{-37}	9 s	2^{-147}	14s	2^{-314}	88 s

Note: The time *t* reported indicates after how long the trail was found within the 24 hours.

6.1.1 Constraints of XOR operation

For every XOR operation with input differences $\mathbf{a} \in \mathbb{F}_2^n$ and $\mathbf{b} \in \mathbb{F}_2^n$ and output difference $\mathbf{c} \in \mathbb{F}_2^n$, the constraints at bit level for j in $\{0 \cdots n-1\}$ are

$$\begin{aligned} &d_{\oplus}[j] \geq a[j] \\ &d_{\oplus}[j] \geq b[j] \\ &d_{\oplus}[j] \geq c[j] \\ &a[j]+b[j]+c[j] \geq +2d_{\oplus}[j] \\ &a[j]+b[j]+c[j] \leq +2 \end{aligned} \tag{28}$$

where $d_{\oplus}[j]$ is a dummy variable used to verify there are at least two active terms in $a[j] \oplus b[j] = c[[j]$ every time $a[j] \neq 0$, $b[j] \neq 0$, or $c[j] \neq 0$.

6.1.2 Constraints of modular addition

To construct the constraints of the modular addition, we followed (Fu et al., 2016). The authors of Fu et al. (2016) used 13 * (n – 1) + 5 inequalities to model the modular addition operation modulus 2^n with input differences

$\mathbf{a} \in \mathbb{F}_2^n$ and $\mathbf{b} \in \mathbb{F}_2^n$, and output difference $\mathbf{c} \in \mathbb{F}_2^n$. For j in $\{0 \cdots n-2\}$ these inequalities are

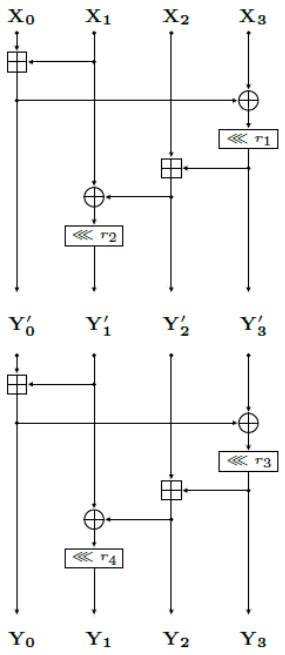
$$\begin{aligned}
&a[j+1]+b[j+1]-c[j+1]+a[j]+b[j]+c[j]+d[j] \geq -0\\
&a[j+1]-b[j+1]+c[j+1]+a[j]+b[j]+c[j]+d[j] \geq -0\\
&-a[j+1]+b[j+1]+c[j+1]+a[j]+b[j]+c[j]+d[j] \geq -0\\
&-a[j+1]-b[j+1]-c[j+1]+a[j]+b[j]+c[j]-d[j] \geq -3\\
&a[j+1]+b[j+1]+c[j+1]+a[j]+b[j]+c[j]-d[j] \geq -0\\
&a[j+1]+b[j+1]+c[j+1]-a[j]-b[j]-c[j]+d[j] \geq -6\\
&-a[j+1]-b[j+1]-c[j+1]+a[j]-b[j]-c[j]+d[j] \geq -6\\
&-a[j+1]-b[j+1]-c[j+1]-a[j]+b[j]-c[j]+d[j] \geq -6\\
&-a[j+1]-b[j+1]-c[j+1]-a[j]-b[j]+c[j]+d[j] \geq -6\\
&a[j+1]+b[j+1]+c[j+1]+a[j]+b[j]-c[j]+d[j] \geq -0\\
&a[j+1]+b[j+1]+c[j+1]+a[j]-b[j]+c[j]+d[j] \geq -0\\
&a[j+1]+b[j+1]+c[j+1]-a[j]+b[j]+c[j]+d[j] \geq -0\\
&a[j+1]+b[j+1]-c[j+1]+a[j]+b[j]+c[j]+d[j] \geq -0.
\end{aligned} \tag{29}$$

And also the inequalities

$$\begin{aligned}
&d_+ \geq a[n-1]\\
&d_+ \geq b[n-1]\\
&d_+ \geq c[n-1]\\
&a[n-1]+b[n-1]+c[n-1] \geq 2d_+\\
&a[n-1]+b[n-1]+c[n-1] \leq 2,
\end{aligned} \tag{30}$$

where d_+ is a dummy variable and $d[j]$ in equation (29) is the variable representing the function *eq* of Theorem 2 in Fu et al. (2016).

Figure 7 The ChaCha quarter round diagram with intermediate variables $\mathbf{Y}'_0$, $\mathbf{Y}'_1$, $\mathbf{Y}'_2$ and $\mathbf{Y}'_3$



6.1.3 Constraints for the quarter round

To model the quarter round it was necessary to create auxiliary variables for the output differences of the half quarter round. In Figure 7, we depict these auxiliary variables $(\mathbf{Y}'_0, \mathbf{Y}'_1, \mathbf{Y}'_2, \mathbf{Y}'_3)$ together with both the inputs variables ($\mathbf{X}_0$, $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$) and outputs variables ($\mathbf{Y}_0$, $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$) of the quarter round. From Figure 7, we can observe that the inequalities for the modular addition operations are the inequalities in equations (29) and (30) where

$$\begin{aligned}
(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \{&(\mathbf{X}_0, \mathbf{X}_1, \mathbf{Y}'_0), (\mathbf{X}_2, \mathbf{Y}'_3, \mathbf{Y}'_2),\\
&(\mathbf{Y}'_0, \mathbf{Y}'_1, \mathbf{Y}_0), (\mathbf{Y}'_2, \mathbf{Y}_3, \mathbf{Y}_2)\}.
\end{aligned} \tag{31}$$

And the inequalities for the XOR operations are the inequalities in equation (28) where

$$\begin{aligned}
(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \{&(\mathbf{X}_3, \mathbf{Y}'_0, \mathbf{Y}'_3 \ggg 16), (\mathbf{X}_1, \mathbf{Y}'_2, \mathbf{Y}'_1 \ggg 12),\\
&(\mathbf{Y}'_3, \mathbf{Y}_0, \mathbf{Y}_3 \ggg 8), (\mathbf{Y}'_1, \mathbf{Y}_2, \mathbf{Y}_1 \ggg 7)\}.
\end{aligned} \tag{32}$$

6.1.4 Constraints for R rounds

Considering that in each round the quarter round is applied four times, then we need to use the inequalities of quarter round (31) and (32) four times. Thus, for the entire R rounds we need $R \times 4$ quarter rounds. This gives us $16n(R + 1)$ variables to model the input and output differences of each quarter round. $16nR$ variables for modelling $d_\oplus$ of equation (28). $16nR$ variables for modelling the intermediate outputs of the quarter rounds. $16R$ variables for modelling d_+ of equation (29). And $16(n - 1)R$ variables for modelling the variable d of equation (29). Summing up, this gives us a total of $16(3Rn + 2n + R - 1)$ variables. The number of inequalities is distributed as follows. $(16)(13)((n - 1)R) + 20$ inequalities for the modular addition operations. And $96nR$ inequalities for the XOR operations. Summing up this gives us a total $16R(19n - 8)$ inequalities.

6.1.5 Objective function

Let R be the number of rounds that we are modelling. Also, let $d[r][j]$ be the variable representing the function *eq* of Theorem 2 in Fu et al. (2016) at round r and bit j, then according to Fu et al. (2016), we need to minimise the following expression

$$\sum_{r=1}^{R} \sum_{i=0}^{n-2} d[r][j] \tag{33}$$

6.2 Experimental result with MILP

We solved the MILP models in a machine running Ubuntu 20.10 and using 56 parallel Intel(R) Xeon(R) Platinum 8280 CPUs at 2.70 GHz. The available RAM was 1535GiB. We used MiniZinc (Nethercote et al., 2007) to implement the model and ORTools (Perron and Furnon, 2023) as the MILP solver. This solver was the winner of the MiniZinc Challenge 2020. In all our experiments, we stopped the

solver after 24 hours, so the solutions found are not always the best. We explored the full space only for 1 and 2 rounds.

In Table 8, Table 9, Table 10, and Table 11, we report the lowest weight differential trails found for the objective function (33) limiting the time of the solver to 24 hours and using the 56 physical cores. These trails are for $R \in \{1, 2, 3, 4\}$.

6.3 Benchmark comparison between MILP and S-function technique

In Table 12, we present a comparison between the MILP model and Algorithm 2 in terms of lowest weight found and time that each technique took to output the solution. The column labelled by p represents the probability of the best trail found, while the column labelled by t represents the time required to find the trail.

In the first row of Table 12 (S-function technique), we present the results corresponding to Algorithm 2. In this case, the results where found with a simple Magma script running on a MacBook Pro with macOS v11.2.3 and using a single 2.4 GHz Intel Core i9, with negligible RAM usage. To have a fair performance comparison with the S-function technique, we tried to solve the MILP model using a single core, and we reported the best trails found in less than 24 hours in the second row of the table. In the third row, we report the best probabilities found in 24 h by running the solver over 56 parallel physical cores and using some specific constraints, such as bounding the Hamming weight of the ChaCha state after the first round.

6.4 Comparison with previous work

In 2017, Aaraj et al. presented a MILP model to find differential trails in ChaCha automatically. To the best of our knowledge, that MILP model is the only one in the literature used to find differential trails in ChaCha core permutation. In this work, the authors inject the input difference in the full input state. Aaraj et al. constructed two MILP models, one at a bit level and the other one at a word level. Using those models, they found differential trails for two rounds of the ChaCha core permutation with a probability of 2^{-24}. Note that our MILP model for the ChaCha core permutation is at the bit level. By considering differentials in any word of the ChaCha state, we found differential trails for up to four rounds with a probability of 2^{-48}. In particular, we found differential trails for two rounds with a probability of 2^{-2}.

We remark that this work is not presenting a full key recovery attack of reduced round ChaCha as it is done, e.g., in Aumasson et al. (2008) and Beierle et al. (2020), but our focus is on finding ChaCha differential trails in an efficient way.

7 Conclusions and future work

We proved exact formulas for the XDP and the ADP of the half quarter round of ChaCha (Lemma 4 and Lemma 8, respectively). Both consist of matrix products that can be computed in linear time $O(n)$, and indeed they are very fast to compute in practice.

Under the hypothesis of independence of half quarter rounds, we find heuristic formulas for the XDP and the ADP of the quarter round of ChaCha (Lemma 5 and Lemma 9, respectively). For small word sizes $n = 5, 6, 7, 8$ (the real word size of ChaCha is $n = 32$), we tested these heuristic formulas by comparing their results with the exact values of XDP and ADP computed by brute force. We found that (on average) these formulas are actually lower bounds for the real XDP and ADP. Moreover, the heuristic formula for the XDP performs better than the one for the ADP, meaning both a smaller average error and a smaller standard deviation (see Table 2 and Table 3). In other words, the hypothesis of independence of half quarter rounds is more accurate for the XDP than the ADP. Finally, we proposed a greedy strategy to compute good quarter round differential characteristics, and used this strategy to provide explicit XOR and additive differential trails for up to three rounds. As a last contribution, we showed how to build an MILP model to find XOR-differential trails in ChaCha permutation. The obtained results are similar to the ones obtained using S-functions. In our implementations, the method using S-functions seems to be significantly faster on a single core, but, especially for larger rounds (3 and 4), a parallelised implementation of the MILP solver returned better probabilities (see Table 12).

We believe these techniques will help to better understand the security of ChaCha stream cipher and of other similar constructions. They could be adopted to improve current linear-differential attacks (where the differential is currently exploited over a single round only), or to build mixed XOR-additive differential attacks. Towards this direction, we believe it would be interesting to find optimal quarter round additive characteristics by starting from the input difference (rather than the output one), and optimal quarter round XOR characteristics from the output difference (rather than the input one). Also, there might exist other greedy strategies which might be more effective and produce characteristics with higher probability. We leave as future research how to use these techniques to mount an attack on ChaCha and affect the security of the cipher.

Acknowledgements

C. Sanna was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU.

The work of Rusydi H. Makarim was carried out at the Technology Innovation Institute LLC

References

Aaraj, N., Caullery, F. and Manzano, M. (2017) 'MILP-aided cryptanalysis of round reduced Chacha', *Cryptology ePrint Archive*, Report 2017/1163 [online] https://ia.cr/2017/1163 (accessed 7 December 2023).

Aumasson, J.P., Çalık, C., Meier, W., Özen, O., Phan, R.C.W. and Varıcı, K. (2009) 'Improved cryptanalysis of Skein', in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, pp.542–559.

Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W. and Rechberger, C. (2008) 'New features of Latin dances: analysis of salsa, chacha, and rumba', in *International Workshop on Fast Software Encryption*, Springer, pp.470–488.

Aumasson, J.P., Neves, S., Wilcox-O'Hearn, Z. and Winnerlein, C. (2013) 'BLAKE2: simpler, smaller, fast as MD5', in *International Conference on Applied Cryptography and Network Security*, Springer, pp.119–135.

Beaulieu, R., Treatman-Clark, S., Shors, D., Weeks, B., Smith, J. and Wingers, L. (2015) 'The SIMON and SPECK lightweight block ciphers', in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, pp.1–6.

Beierle, C., Biryukov, A., dos Santos, L.C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q. and Biryukov, A. (2019) 'Lightweight {AEAD} and hashing using the sparkle permutation family', *IACR, Trans. Symmetric Cryptol*., Vol. 2020, No. S1, pp.208–261.

Beierle, C., Leander, G. and Todo, Y. (2020) 'Improved differential-linear attacks with applications to arx ciphers', in *Annual International Cryptology Conference*, Springer, pp.329–358.

Bernstein, D.J. (2005) *Salsa20 Specification. eSTREAM Project Algorithm Description* [online] https: //www.ecrypt.eu.org/ stream/salsa20pf.html (accessed 7 December 2023).

Bernstein, D.J. (2008a) 'ChaCha, a variant of Salsa20', in *Workshop Record of SASC*, Vol. 8, pp.3–5.

Bernstein, D.J. (2008b) 'The Salsa20 family of stream ciphers', in *New Stream Cipher Designs, Lecture Notes in Computer Science*, Vol. 4986, pp.84–97, Springer.

Bernstein, D.J. (2008c) *Cubehash specification (2. b. 1)*, Submission to NIST.

Biryukov, A. and Velichkov, V. (2014) 'Automatic search for differential trails in ARX ciphers', in *Cryptographers' Track at the RSA Conference*, Springer, pp.227–250.

Biryukov, A., Velichkov, V. and Le Corre, Y. (2016) 'Automatic search for the best trails in ARX: application to block cipher SPECK', in *International Conference on Fast Software Encryption*, Springer, pp.289–310.

Bresson, E., Canteaut, A., Chevallier-Mames, B., Clavier, C., Fuhr, T., Gouget, A., Icart, T., Misarsky, J.F., Naya-Plasencia, M., Paillier, P. et al. (2008) *Shabal, A Submission to NIST'S Cryptographic Hash Algorithm Competition*, Submission to NIST.

Chittenden, E.W. (1947) 'On the number of paths in a finite partially ordered set', *Amer. Math. Monthly*, Vol. 54, No. 7, pp.404–405.

Coutinho, M. and Neto, T.C.S. (2021) 'Improved linear approximations to ARX ciphers and attacks against ChaCha', in Canteaut, A. and Standaert, F. (Eds.): *Advances in Cryptology – EUROCRYPT 2021 – 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Zagreb, Croatia, 17–21 October, Proceedings, Part I, Lecture Notes in Computer Science, Vol. 12696, pp.711–740, DOI: 10.1007/978-3-030-77870-5n25, https://doi.org/10.1007/978-3-030-77870-5_25.

Daum, M. (2005) *Cryptanalysis of Hash Functions of the Md4-Family*, PhD thesis, Ruhr University Bochum.

De Canniere, C. and Rechberger, C. (2006) 'Finding SHA-1 characteristics: General results and applications', in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, pp.1–20.

Dey, S. and Sarkar, S. (2017) 'Improved analysis for reduced round Salsa and Chacha', *Discret. Appl. Math.*, Vol. 227, pp.58–69, DOI: 10.1016/j.dam.2017.04.034, https://doi.org/ 10. 1016/j.dam.2017.04.034.

Dey, S., Garai, H.K., Sarkar, S., Sharma, N.K. (2022) 'Revamped differential-linear cryptanalysis on reduced round ChaCha', in Dunkelman, O. and Dziembowski, S. (Eds.): *Advances in Cryptology – EUROCRYPT 2022 – 41st Annual International Conference on the Theory and Applications of Cryptographic Techniq*ues, Springer, Trondheim, Norway, 30 May–3 June, Proceedings, Part III, Lecture Notes in Computer Science, Vol. 13277, pp.86–114, DOI: 10.1007/978-3-031-07082-2n 4, https: //doi.org/10.1007/978-3-031-07082-2_4.

Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J. and Walker, J. (2010) *The Skein Hash Function Family*, Submission to NIST (round 3), Vol. 7, No. 7.5, p.3.

Fu, K., Wang, M., Guo, Y., Sun, S. and Hu, L. (2016) 'Milp-based automatic search algorithms for differential and linear trails for speck', *FSE, Lecture Notes in Computer Science*, Vol. 9783, pp.268–288, Springer.

Gligoroski, D., Klima, V., Knapskog, S.J., El-Hadedy, M., Amundsen, J. (2009) 'Cryptographic hash function blue midnight wish', in *2009 Proceedings of the 1st International Workshop on Security and Communication Networks*, IEEE, pp.1–8.

Hong, D., Lee, J.K., Kim, D.C., Kwon, D., Ryu, K.H. and Lee, D.G. (2013) 'Lea: a 128-bit block cipher for fast encryption on common processors', in *International Workshop on Information Security Applications*, Springer, pp.3–27.

Leurent, G., Bouillaguet, C. and Fouque, P.A. (2009) *Simd is a Message Digest*, Submission to the NIST SHA-3 Competition (Round 2).

Lipmaa, H. and Moriai, S. (2001) 'Efficient algorithms for computing differential properties of addition', in *FSE 2001, Lecture Notes in Computer Science*, Springer, Vol. 2355, pp.336–350.

Lipmaa, H., Wallen, J. and Dumas, P. (2004) 'On the additive differential probability of exclusive Or', in *FSE 2004, Lecture Notes in Computer Science*, Springer, Vol. 3017, pp.317–331.

Mehner, C.E. (2019) *Limdolen* [online] https://github.com/cem-/limdolen/blob/master/Documents/Limdolen%20Specification .pdf (accessed 7 December 2023).

Mouha, N., De Canniere, C., Indesteege, S. and Preneel, B. (2009) 'Finding collisions for a 45-step simplified HAS-V', in *International Workshop on Information Security Applications*, Springer, pp.206–225.

Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B. and Verbauwhede, I. (2014) 'Chaskey: an efficient MAC algorithm for 32-bit microcontrollers', in *International Conference on Selected Areas in Cryptography*, Springer, pp.306–323.

Mouha, N., Velichkov, V., De Canniere, C. and Preneel, B. (2010) 'The differential analysis of Sfunctions', in *International Workshop on Selected Areas in Cryptography*, Springer, pp.36–56.

Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G. (2007) 'Minizinc: towards a standard cp modelling language', in Bessiere, C. (Ed.): *Principles and Practice of Constraint Programming – CP 2007*, pp.529–543, Springer Berlin Heidelberg, Berlin, Heidelberg.

NIST (2007) *Hash Functions – SHA-3 Project* [online] https://csrc.nist.gov/projects/hash-functions/sha-3-project (accessed 7 December 2023).

NIST (2019) *Lightweight Cryptography Standardization Process* [online] https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates (accessed 7 December 2023).

Perron, L. and Furnon, V. (2023) *Or-Tools* [online] https://developers.google.com/optimization/ (accessed 7 December 2023).

Rivest, R.L. (1994) 'The rc5 encryption algorithm', in *International Workshop on Fast Software Encryption*, Springer, pp.86–96.

Shi, Z., Zhang, B., Feng, D., Wu, W. (2012) 'Improved key recovery attacks on reduced-round Salsa20 and ChaCha', in Kwon, T., Lee, M. and Kwon, D. (Eds.): *Information Security and Cryptology – ICISC 2012 – 15th International Conference*, Springer, Seoul, Korea, 28–30 November, Revised Selected Papers, Lecture Notes in Computer Science, Vol. 7839, pp.337–351, DOI 10.1007/978-3-642-37682-5n24, https://doi.org/10.1007/978-3-642-37682-5_24.

Velichkov, V., Mouha, N., De Canniere, C., Preneel, B. (2011) 'The additive differential probability of ARX', in *International Workshop on Fast Software Encryption*, Springer, pp.342–358.

Velichov, V. (2012) 'UNAF: a special set of additive differences with application to the differential analysis of ARX', in *FSE 2012, Lecture Notes in Computer Science*, Springer, Vol. 7549, pp.287–305.

Wheeler, D.J. and Needham, R.M. (1994) 'TEA, a tiny encryption algorithm', in *International Workshop on Fast Software Encryption*, Springer, pp.363–366.

Notes

1 While for the XOR-differential trails the difference is injected in the nonce and in the counter (the fourth row of the state), in the ModAdd differential trail, we inject the difference in the full state.

2 In other words Algorithm 2 can be used to attack ChaCha stream cipher (in a single key scenario), while Algorithm 4 can only be used to attack ChaCha permutation, or in combination with other techniques.