Noname manuscript No.
(will be inserted by the editor)

Finding differential trails on ChaCha by means of state functions

Emanuele Bellini · Rusydi Makarim · Carlo Sanna

Received: date / Accepted: date

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 maximize differential characteristics for the full quarter-round, and use it to determine explicit differential trails for the ChaCha permutation. 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

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 <u>A</u>ddition, bitwise <u>R</u>otation, and e<u>X</u>clusive OR, hence the name ARX.

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

E. Bellini
Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
E-mail: emanuele.bellini@tii.ae

R. Makarim
Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
E-mail: rusydi.makarim@tii.ae

C. Sanna
Politecnico di Torino, Torino, Italy
member of GNSAGA of INdAM and of CrypTO, the group of Cryptography and Number Theory of Politecnico di Torino
E-mail: carlo.sanna.dev@gmail.com

1. Cryptographic permutations such as SPARKLE (SCHWAEMM and ESCH) [4] (2019), candidate to the NIST Lightweight Cryptography standardization process (NIST LWC) [27].
2. Block ciphers such as the Rivest cipher RC5 [28] (1994), the South Korean Electronic and Telecommunication Research Institute cipher LEA [18] (2013) the NIST LWC candidate Limdolen [22] (2019, using a Feistel structure and ARX operations to achieve diffusion), the American NSA cipher Speck [3] (2013) standardization in ISO/IEC 29167-22, the Tiny Encryption Algorithm TEA [31] (1994) and Threefish [16] (2010), used as Skein internal permutation.
3. Stream ciphers such as Bernstein's Salsa20 [6,9] (2005) and ChaCha20 [7] (2008). The latter one is part of the TLS 1.3 standard.
4. Hash functions such as the SHA-3 Project [26] finalists (2007 - 2012) BLAKE2 [2] and Skein [16], and other SHA-3 candidates, Blue Midnight Wish [17], CubeHash [8], Shabal [12], SIMD [19].
5. Message Authentication Codes such as Chaskey [24] (2014), standardized 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 $\mathsf{xdp}^{\boxplus}$, in a linear time with respect to the input size, were Lipmaa and Moriai, in 2001 [20]. 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 [25]). In 2004, Lipmaa, Wallén, and Dumas [21] obtained the dual result of [20], by computing the additive differential probability of the XOR operation, denoted by $\mathsf{adp}^{\oplus}$.

In 2005, in his Ph.D. thesis [14], Daum collected a set of differential properties of bit rotation, in particular he defined the additive and the XOR differential probability of bitwise rotation, $\mathsf{adp}^{\lll}$ and $\mathsf{xdp}^{\lll}$.

Taking inspiration from the cryptanalysis techniques for SHA-1 by De Cannière and Rechberger [15, 23], the results of [20] and [21] were generalized by Mouha et al. in 2010 [25]. 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, in 2011, Velichkov et al. [29] showed how to compute the additive differential probability of what they called the ARX operation, i.e. $\mathsf{ARX}(a, b, r, d) = ((a \boxplus b) \lll r) \oplus d$. 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., in 2009 [1], use the algorithms provided in [20] for computing differential properties of modular addition to find modular differentials and mount a boomerang attack on Threefish. Since the methods from [29] do not scale well with large components, In 2012, Velichkov et al. [30] introduced the concept of a UNAF difference, representing a set of specially chosen additive differences. This allows them to find a 3-round differential trail in Salsa stream cipher of probability 2^{-4}, and then to mount a key recovery attack on Salsa reduced to 5 rounds, with data complexity of 2^7 chosen plaintexts and time complexity of 2^{167} encryptions. A couple of years later, the results from [20], [21], [25], and [29] were used by Biryukov et al. to instantiate automatic search of differential trails in TEA, XTEA, RAIDEN [10], and in SPECK [11] block ciphers.

Currently, the best known attack on ChaCha20 stream cipher is [5], where 6 rounds are attacked using $2^{77.4}$ time and 2^{58} data, while 7 rounds are attacked using $2^{230.86}$ time and $2^{48.83}$ data. The attack is a combination of differential and linear cryptanalysis.

1.2 Our contribution

In this work, we slightly generalize 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 maximize differential characteristic probability for the full quarter-round, and then use this strategy to find explicit XOR and additive differential trails up to 3 rounds, with probability, respectively, 2^{-166} and 2^{-98}. 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 maximization of the XDP for ChaCha quarter round, while from subsection 3.5 to subsection 3.6 we deal with the same problem in the ADP case. In section 4, we provide explicit differential trails and simple statistics on the minimum, maximum and average quarter round differential characteristic probability. Finally, in section 5 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:

$x[i]$	ith 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
$x \mid\mid y$	concatenation of x and y

Moreover, for vectors $\mathbf{x}, \mathbf{y} \in \mathbb{W}_n^k$, 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 $\mathsf{w} = 32$ bits, i.e.

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

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

$$y_0 = x_0 \boxplus x_1$$

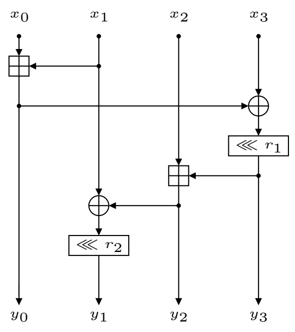

Fig. 1 The ChaCha half quarter round diagram.

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

Definition 2 (ChaCha quarter round) Let $x_i, y_i, i = 0, 1, 2, 3$ be w-bit words and $r_1, r_2, r_3, r_4 \in \{1, \ldots, \mathsf{w}-1\}$. Then we define ChaCha quarter round $(y_0, y_1, y_2, y_3) = \mathsf{QR}(x_0, x_1, x_2, x_3)$ as follows:

$$\begin{aligned} (y'_0, y'_1, y'_2, y'_3) &= \mathsf{HQR}_{r_1,r_2}(x_0, x_1, x_2, x_3) \\ (y_0, y_1, y_2, y_3) &= \mathsf{HQR}_{r_3,r_4}(y'_0, y'_1, y'_2, y'_3) \end{aligned}$$

We show in Fig. 1 a schematic drawing of Chacha half quarter round. The permutation used in ChaCha20 stream cipher performs 20 rounds or, equivalently, 10 *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:

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 $\mathsf{n} \times \mathsf{n}$ matrices with entries in $\mathbb{F}_2^{\mathsf{w}}$.

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

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

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

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

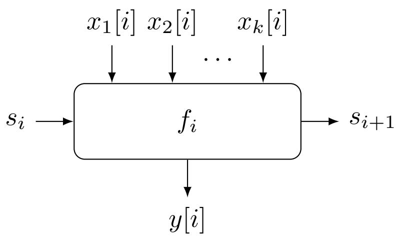

Fig. 2 Representation of the ith block of an S-machine.

2.3 Definition of differential probability

Definition 4 Let f be a function from $\mathbb{F}_2^{n_1}$ to $\mathbb{F}_2^{n_2}$. The *XOR differential probability* (XDP) of f with respect to the input/output pair $(\Delta x, \Delta y) \in \mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2}$ is defined as

$$\mathsf{xdp}^f(\Delta x, \Delta y) = \mathsf{xdp}^f(\Delta x \xrightarrow{f} \Delta y) = \frac{\#\{x \in \mathbb{F}_2^{n_1} : f(x \oplus \Delta x) = f(x) \oplus \Delta y\}}{\#\{x \in \mathbb{F}_2^{n_1}\}}$$

The *additive differential probability* (ADP) of f with respect to the input/output pair $(\Delta x, \Delta y) \in \mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2}$ is defined as

$$\mathsf{adp}^f(\Delta x, \Delta y) = \mathsf{adp}^f(\Delta x \xrightarrow{f} \Delta y) = \frac{\#\{x \in \mathbb{F}_2^{n_1} : f(x \boxplus \Delta x) = f(x) \boxplus \Delta y\}}{\#\{x \in \mathbb{F}_2^{n_1}\}}$$

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 Section 3.4. Actually, we shall develop a bit more theory than the one strictly necessary for Section 3.4.

S-functions were introduced in [25] and were already applied to the differential cryptanalysis of some ARX primitives [25,29]. 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 5 An *S-machine* is a $(n+2)$-tuple $(\mathcal{S}, s_{\text{in}}, f_0, \ldots, f_{n-1})$ consisting of:

- A finite set of *states* $\mathcal{S}$;
- An *initial state* $s_{\text{in}} \in \mathcal{S}$;
- 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 labeled by $i = 0, \ldots, n-1$ (see Figure 2). Starting from $i = 0$, the ith 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 $(i+1)$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.

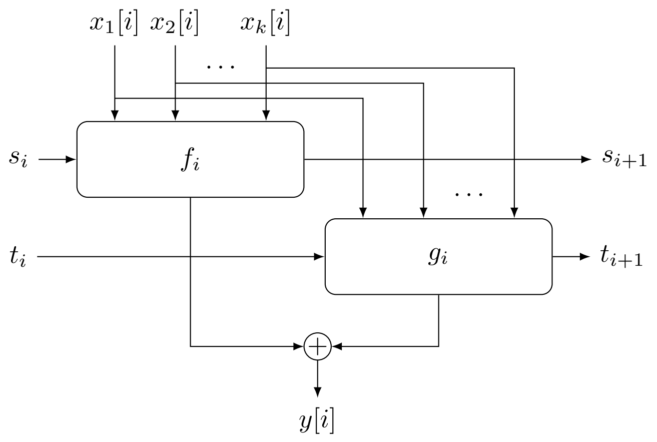

Fig. 3 The ith block of the XOR of two S-functions.

Definition 6 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 $(\mathcal{S}, s_{\text{in}}, f_0, \ldots, f_{n-1})$ with the following property: For every $(x_1, \ldots, x_k) \in \operatorname{dom}(F)$ there exist some states $s_0 = s_{\text{in}}, s_1, \ldots, s_n \in \mathcal{S}$ such that

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

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 [25] in two ways. First, in [25] the transition functions f_i for $i = 0, \ldots, n-1$ are all equal to a single function f, although a generalization with different transition functions is already suggested. Second, and more important, our definition lets the transition functions be *partial* functions, while in [25] only total functions are considered.

It is easy to see that, among the functions $(\mathbb{F}_2^n)^k \to \mathbb{F}_2$, all projections $(x_1, \ldots, x_k) \mapsto x_j$, with $j \in \{1, \ldots, n\}$, and all constant functions $(x_1, \ldots, x_k) \to c$, 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 1 *If F and G are S-functions, then $F \oplus G$ and $F \boxplus G$ are S-functions.*

Proof Let $(\mathcal{S}, s_{\text{in}}, f_0, \ldots, f_{n-1})$ and $(\mathcal{T}, t_{\text{in}}, g_0, \ldots, g_{n-1})$ 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 $(s_{\text{in}}, t_{\text{in}})$, and ith 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 $(s_{\text{in}}, t_{\text{in}}, 0)$, and ith 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 $x_1[0], \ldots, x_k[0]$.

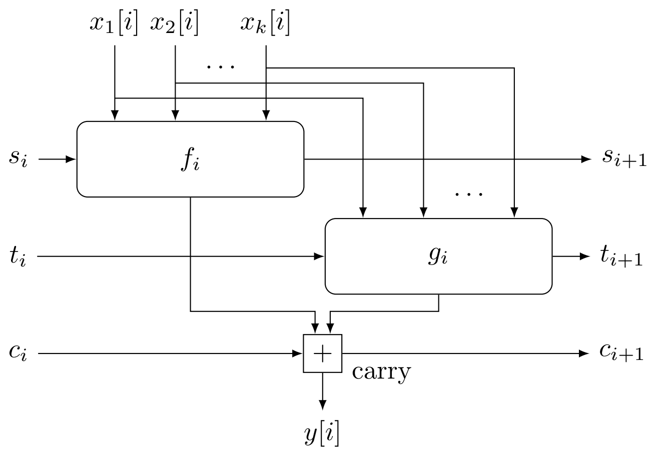

Fig. 4 The ith block of the addition of two S-functions.

Definition 7 Let F be an S-function with S-machine $(\mathcal{S}, s_{\text{in}}, f_0, \ldots, f_{n-1})$, and let $i \in \{0, \ldots, n-1\}$ and $\gamma \in \mathbb{F}_2$. Also, let $s_1 := s_{\text{in}}, s_2, \ldots, s_h$ be all the elements of $\mathcal{S}$. The ith *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 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 $(\mathcal{S}, s_{\text{in}}, f_0, \ldots, f_{n-1})$ be the S-machine of F and let $s_1 := s_{\text{in}}, s_2, \ldots, s_h$ 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 [13], 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 $(x_1, \dots, x_k) \in \mathrm{dom}(F)$ 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 $\overleftarrow{\boxplus}^r$ by

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

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

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

where $c := \lfloor (x_L + y_L)/2^r \rfloor$. Hence, the computation $x \overleftarrow{\boxplus}^r y$ 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 $(n-r)$th digit; and there the carry c of the nth digit is added to the least significant digit. In particular, note that $x \overleftarrow{\boxplus}^r y$ 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 $(n-r)$th position. However, assuming that we know the value of c in advance, we can compute $x \overleftarrow{\boxplus}^r y$ by an S-machine an check at the end that the nth carry is actually equal to c. This would be our strategy to prove Lemma 5 later.

2.6 XDPs and ADPs of composite functions

In general, given two functions $F : \mathbb{W}_n^k \to \mathbb{W}_n^h$ and $G : \mathbb{W}_n^h \to \mathbb{W}_n^\ell$, there is no simple way to express the XDP of the composite function $F \circ G$ in terms of the XDPs of F and G. However, assuming that F and G are sufficiently "independent", a heuristic formula for the XDP of $G \circ F$ can be given. Precisely, for every $\Delta\mathbf{x} \in \mathbb{W}_n^k$ and $\Delta\mathbf{z} \in \mathbb{W}_n^\ell$, we have

$$\begin{aligned} &\mathsf{xdp}^{G \circ F}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = \Pr\big[G\big(F(\mathbf{x} \boxplus \Delta\mathbf{x})\big) = G\big(F(\mathbf{x})\big) \boxplus \Delta\mathbf{z}\big] \\ &= \sum_{\Delta\mathbf{y} \in \mathbb{W}_n^k} \Pr\big[F(\mathbf{x} \boxplus \Delta\mathbf{x})\big) = F(\mathbf{x}) \boxplus \Delta\mathbf{y} \wedge G\big(F(\mathbf{x}) \boxplus \Delta\mathbf{y}\big) = G\big(F(\mathbf{x})\big) \boxplus \Delta\mathbf{z}\big] \\ &= \sum_{\Delta\mathbf{y} \in \mathbb{W}_n^k} \Pr\big[F(\mathbf{x} \boxplus \Delta\mathbf{x})\big) = F(\mathbf{x}) \boxplus \Delta\mathbf{y}\big] \Pr\big[G\big(F(\mathbf{x}) \boxplus \Delta\mathbf{y}\big) = G\big(F(\mathbf{x})\big) \boxplus \Delta\mathbf{z}\big] \qquad (1) \\ &= \sum_{\Delta\mathbf{y} \in \mathbb{W}_n^k} \Pr\big[F(\mathbf{x} \boxplus \Delta\mathbf{x})\big) = F(\mathbf{x}) \boxplus \Delta\mathbf{y}\big] \Pr\big[G\big(\mathbf{w} \boxplus \Delta\mathbf{y}\big) = G\big(\mathbf{w}\big) \boxplus \Delta\mathbf{z}\big] \qquad (2) \end{aligned}$$

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

where in (1) we assumed that the two events in the probability of the previous line are independent, while in (2) we assumed that the change of variable $F(\mathbf{x}) = \mathbf{w}$ does not affect the probability.

Reasoning in exactly the same way, we get that the following heuristic formula for the ADP of $G \circ F$

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

assuming that F and G are sufficiently “independent”.

3 XOR and additive differential probability of ChaCha round

In this section, we first give an exact formula for the XOR (respectively, additive) differential probability of the half quarter round of ChaCha. Then we provide a heuristic formula for the XDP (respectively, ADP) 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 (respectively, ADP) of ChaCha full round. For the ease of exposition, we keep the exposition of XDP and ADP independent.

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 [25] 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 $L = \begin{pmatrix} 1 & 0 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$.

Lemma 2 *Let A_w, L, and C be the matrices defined above. Then, for all $\Delta x_0, \Delta x_1, \Delta y \in \mathbb{W}_n$, we have*

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

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

Proof See [21, Theorem 4]. Note that, our A_w, L, C are the transposes of the A_w, L, C in [21], 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 3 *For all* $\Delta\mathbf{x}, \Delta\mathbf{y} \in \mathbb{W}_n^4$, *we have*

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

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

$$\mathsf{xdp}^{\mathsf{HQR}_{r_1,r_2}}(\Delta\mathbf{x} \to \Delta\mathbf{y}) = \mathsf{xdp}^{\boxplus}(\Delta x_0, \Delta x_1 \to \Delta y_0) \cdot \mathsf{xdp}^{\boxplus}(\Delta y_3, \Delta x_2 \to \Delta y_2).$$

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

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

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 $\mathbf{x}$ and $\mathbf{y}$. Therefore, they are necessary conditions for the XDP to be nonzero.

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

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

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 Lemmas 2 and 3, 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 4 *Assuming that* HQR_{r_1,r_2} *and* HQR_{r_3,r_4} *are "independent", for every* $\Delta\mathbf{x}, \Delta\mathbf{z} \in \mathbb{W}_n^4$ *we have*

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

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

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

Proof Since $\mathsf{QR} = \mathsf{HQR}_{r_3,r_4} \circ \mathsf{HQR}_{r_1,r_2}$, by the reasoning of Section 2.6 we have

$$\mathsf{xdp}^{\mathsf{QR}}(\Delta\mathbf{x} \to \Delta\mathbf{z}) = \sum_{\Delta\mathbf{y} \in \mathbb{W}_n^k} \mathsf{xdp}^{\mathsf{HQR}_{r_1,r_2}}(\Delta\mathbf{x} \to \Delta\mathbf{y})\, \mathsf{xdp}^{\mathsf{HQR}_{r_3,r_4}}(\Delta\mathbf{y} \to \Delta\mathbf{z}). \qquad (13)$$

Furthermore, from the first part of Lemma 3, we get that the addend of (13) is non-zero only if it holds the following system of equations

$$\begin{aligned}\Delta y_0 \oplus \Delta x_3 &= \Delta y_3 \ggg r_1 \\ \Delta y_2 \oplus \Delta x_1 &= \Delta y_1 \ggg r_2 \\ \Delta z_0 \oplus \Delta y_3 &= \Delta z_3 \ggg r_1 \\ \Delta z_2 \oplus \Delta y_1 &= \Delta z_1 \ggg r_2,\end{aligned}$$

which solved gives a unique value of $\Delta\mathbf{y}$ by (12). Then the claim follows from the second part of Lemma 3.

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 4 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 $\Delta\mathbf{y} = \mathsf{QR}(\mathbf{x} \oplus \Delta\mathbf{x}) \oplus \mathsf{QR}(\mathbf{x})$, which guarantees that the XDP is nonzero. We collect the results in Table 1 and Figure 5, which shows the distribution of $L = \log(\text{exact value of } \mathsf{xdp}^{\mathsf{HQR}}/\text{heuristic value of } \mathsf{xdp}^{\mathsf{HQR}})$, given by Lemma 4 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 1 E is the average factor which the heuristic formula of Lemma 4 is off from the exact XDP, and σ is the standard deviation (sample size $N = 16000$)

n	5	6	7	8
E	0.67	0.63	0.60	0.57
σ	0.52	0.56	0.61	0.65

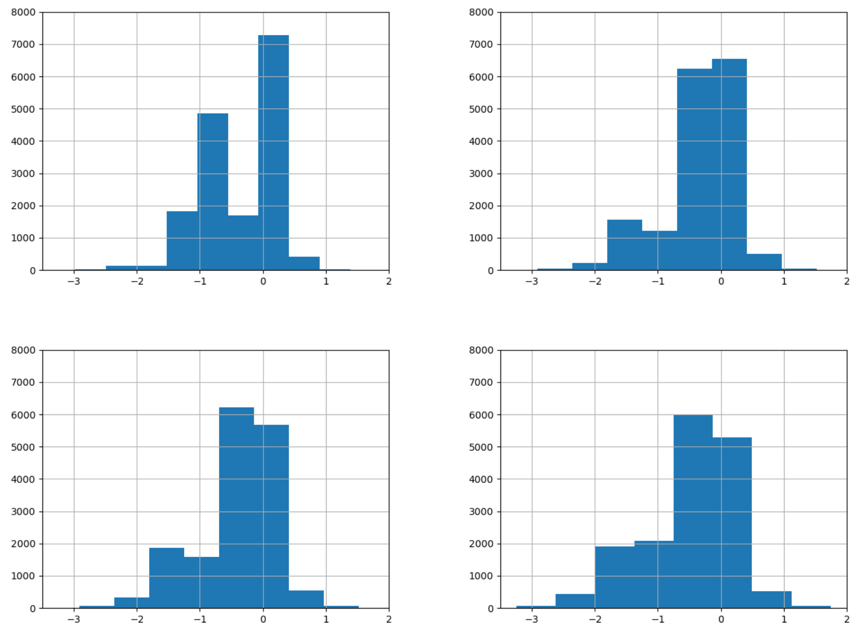

Fig. 5 Distribution of the logarithm of the ratio between the XDP given by Lemma 4 and the correct value of the XDP, for $n = 5$ (top-left), $n = 6$ (top-right), $n = 7$ (bottom-left), and $n = 8$ (bottom-right).

3.3 Maximizing 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 $\mathsf{xdp}^{\mathsf{HQR}_{r_1,r_2}}(\Delta\mathbf{x} \to \Delta\mathbf{y})$ 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 2, 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 ouput $\Delta y \in \mathbb{W}_n$ and p such that $p = \mathsf{xdp}^{\boxplus}(\Delta x_0, \Delta x_1 \to \Delta y)$ 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.

Then, accordingly to Lemma 3, 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($\Delta x_0, \Delta x_1, c$):
    $\Delta y \leftarrow 0$ ($n$ 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 $\Delta y$, $p$
```

Algorithm 2:

```
Function
  Greedy_XDP_HQR($r_1, r_2, \Delta x_0, \Delta x_1, \Delta x_2, \Delta x_3, c_0, c_1$):
    $\Delta y_0, p_0 \leftarrow$ Greedy_XDP_Add
      ($\Delta x_0, \Delta x_1, c_0$)
    $\Delta y_3 \leftarrow (\Delta x_3 \oplus \Delta y_0) \lll r_1$
    $\Delta y_2, p_1 \leftarrow$ Greedy_XDP_Add
      ($\Delta y_3, \Delta x_2, c_1$)
    $\Delta y_1 \leftarrow (\Delta x_1 \oplus \Delta y_2) \lll r_2$
    $p \leftarrow p_0 p_1$
    return $\Delta y_0, \Delta y_1, \Delta y_2, \Delta y_3, p$
```

3.4 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

$$A_{000} = \begin{pmatrix} 4&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0 \\ 1&0&1&0&1&0&1&0 \\ 0&0&0&0&0&0&0&0 \end{pmatrix} \quad A_{001} = \begin{pmatrix} 0&0&0&0&0&0&0&0 \\ 0&4&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 \end{pmatrix} \quad A_{010} = \begin{pmatrix} 0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0 \\ 0&0&4&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&0&1&0&1&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&1&1 \end{pmatrix}$$

$$A_{011} = \begin{pmatrix} 1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&4&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&0&0&0&0&0&0&0 \end{pmatrix} \quad A_{100} = \begin{pmatrix} 0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0 \\ 1&0&1&0&1&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&4&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1 \end{pmatrix} \quad A_{101} = \begin{pmatrix} 1&1&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&4&0&0 \\ 0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0 \end{pmatrix}$$

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

$$L_0 = \begin{pmatrix} 1&0&0&0&0&0&0&0 \end{pmatrix} \qquad L_1 = \begin{pmatrix} 0&1&0&0&0&0&0&0 \end{pmatrix}$$

$$C_0 = \begin{pmatrix} 1&0&1&0&1&0&1&0 \end{pmatrix}^T \qquad C_1 = \begin{pmatrix} 0&1&0&1&0&1&0&1 \end{pmatrix}^T$$

Lemma 5 *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*

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

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

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

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

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 $c_0 \mid\mid c_1 \mid\mid c_{01}$, where c_0, c_1, c_{01} are the carries in the first, second, and third addition of (15), respectively. We identify each state $c_0 \mid\mid c_1 \mid\mid c_{01}$ with the corrisponding 3-bit integer $4c_0 + 2c_1 + c_{01}$. The S-function for (15) is defined by the recurrences

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

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

$$\begin{aligned}
\alpha_0 \oplus \alpha_1 \oplus \alpha_{01} \oplus c_0 \oplus c_1 \oplus c_{01} = 0, c_0' = \lfloor (\chi_0 + \alpha_0 + c_0)/2 \rfloor,\\
c_1' = \lfloor (\chi_1 + \alpha_1 + c_1)/2 \rfloor, c_{01}' = \big\lfloor \big((\chi_0 \oplus \chi_1) + \alpha_{01} + c_{01}\big)/2 \big\rfloor,
\end{aligned}$$

with $i = 4c_0 + 2c_1 + c_{01}$ and $i = 4c_0' + 2c_1' + c_{01}'$. Finally, in light of Remark 3 and Section 2.5, we introduce the matrices R, L_i, and C_i to handle $\overrightarrow{\boxplus}^r$: the projection matrix R has the purpose to not propagate the carry c_{01} of the $(n-r)$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 6 *Let* A_w, L_i, C_i, *and* R *be the matrices of Lemma 5. For all* $\Delta \mathbf{x} \in \mathbb{W}_n^2$, $\Delta y \in \mathbb{W}_n$, *and every integer* $r \in [0, n)$, *we have*

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

where $w[i] := \Delta x_0[i] \mid\mid \Delta x_1[i] \mid\mid (\Delta y \ggg r)[i]$ *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 5.

Lemma 7 *For all* $\Delta\mathbf{x}, \Delta\mathbf{y} \in \mathbb{W}_n^4$, *we have*

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

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

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

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

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

is equivalent to

$$(x_0 \boxplus \Delta x_0) \boxplus (x_1 \boxplus \Delta x_1) = (x_0 \boxplus x_1) \boxplus \Delta y_0 \tag{19}$$
$$\big((y_0 \boxplus \Delta y_0) \oplus (x_3 \boxplus \Delta x_3)\big) \lll r_1 = \big((y_0 \oplus x_3) \lll r_1\big) \boxplus \Delta y_3 \tag{20}$$
$$(y_3 \boxplus \Delta y_3) \boxplus (x_2 \boxplus \Delta x_2) = (y_3 \boxplus x_2) \boxplus \Delta y_2 \tag{21}$$
$$\big((y_2 \boxplus \Delta y_2) \oplus (x_1 \boxplus \Delta x_1)\big) \lll r_2 = \big((y_2 \oplus x_1) \lll r_2\big) \boxplus \Delta y_1. \tag{22}$$

Equations (19) and (21) simplify at once to (16) and (17), respectively, which do not depend on $\mathbf{x}$ and $\mathbf{y}$. Therefore, they are necessary conditions for the adp to be nonzero.

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

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

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

$$\big((z_0 \boxplus \Delta y_0) \oplus (z_3 \boxplus \Delta x_3)\big) \lll r_1 = \big((z_0 \oplus z_3) \lll r_1\big) \boxplus \Delta y_3 \tag{23}$$
$$\big((z_1 \boxplus \Delta x_1) \oplus (z_2 \boxplus \Delta y_2)\big) \lll r_2 = \big((z_1 \oplus z_2) \lll r_2\big) \boxplus \Delta y_1. \tag{24}$$

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 Lemmas 6 and 7, the ADP of ChaCha half quarter round can be computed in time $O(n)$.

3.5 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 8 *Assuming that* HQR_{r_1,r_2} *and* HQR_{r_2,r_3} *are "independent", for every* $\Delta\mathbf{x}, \Delta\mathbf{z} \in \mathbb{W}_n^4$ *we have*

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

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

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

Proof Since $\mathsf{QR} = \mathsf{HQR}_{r_3,r_4} \circ \mathsf{HQR}_{r_1,r_2}$, by the reasoning of Section 2.6 we have

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

Furthermore, from the first part of Lemma 7, we get that the addend of (27) is non-zero only if it holds the following system of equations

$$\Delta x_0 \boxplus \Delta x_1 = \Delta y_0, \Delta y_3 \boxplus \Delta x_2 = \Delta y_2, \Delta y_0 \boxplus \Delta y_1 = \Delta z_0, \Delta z_3 \boxplus \Delta y_2 = \Delta z_2,$$

which solved gives a unique value of $\Delta\mathbf{y}$ by (26). Then the claim follows from the second part of Lemma 7.

Table 2 E is the average factor which the heuristic formula of Lemma 8 is off from the exact ADP, and σ is the standard deviation (sample size $N = 15000$)

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 8 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 $\mathbf{x} \in \mathbb{W}_n^4$ and we picked $\Delta\mathbf{y} = \mathsf{HQR}(\mathbf{x} \boxplus \Delta\mathbf{x}) \boxplus \mathsf{HQR}(\mathbf{x})$, which guarantees that the ADP is nonzero. We collect the results in Table 2 and Figure 6.

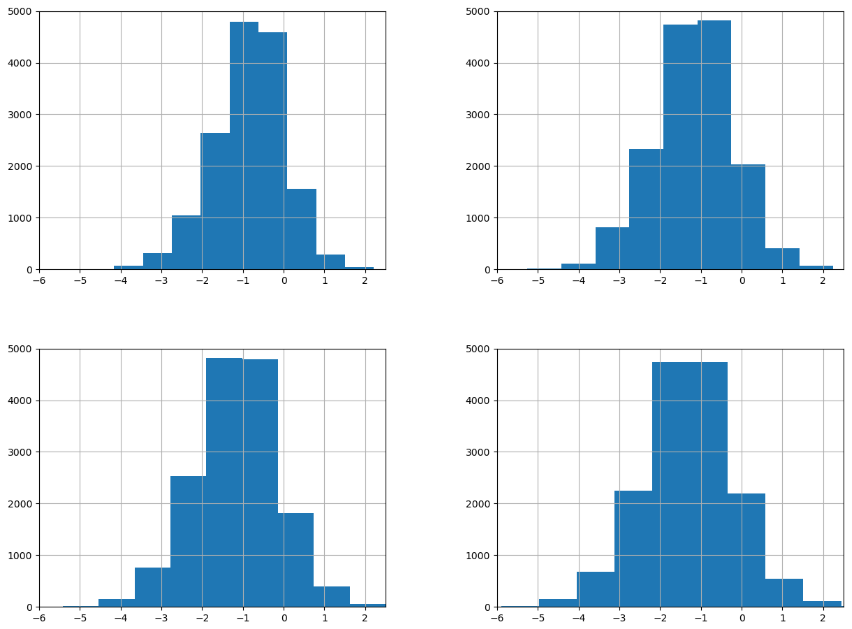

Fig. 6 Distribution of the logarithm of the ratio between the ADP given by Lemma 8 and the correct value of the ADP, for $n = 5$ (top-left), $n = 6$ (top-right), $n = 7$ (bottom-left), and $n = 8$ (bottom-right).

3.6 Maximizing 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 $\mathsf{adp}^{\mathsf{HQR}_{r_1,r_2}}(\Delta\mathbf{x} \to \Delta\mathbf{y})$ 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 Section 3.3 to maximize 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 6, 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 $p = \mathsf{adp}^{J_r}(\Delta x_0, \Delta x_1 \to \Delta y)$ is high.

Then, accordingly to Lemma 7, 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 maximized using Algorithm 3. On the contrary, if $\Delta\mathbf{x}$ is fixed, the two factors are not independent functions of $\Delta y_0, \Delta y_3$ and $\Delta y_1, \Delta y_2$, because equations (16) and (17) have to be taken into account, and consequently they cannot be independently maximized.

Algorithm 3:

```
Function Greedy_ADP_J($r, \Delta x_1, \Delta y, c$):
    $\Delta x_0 \leftarrow 0$ ($n$ bits word)
    $B \leftarrow 8 \times 8$ identity matrix
    $p \leftarrow 1$
    for $i = 0, 1, \ldots, n-1$ do
        $B_0 \leftarrow BA_{0\,||\,\Delta x_1[i]\,||\,(\Delta y \ggg r)[i]}$
        $B_1 \leftarrow BA_{1\,||\,\Delta x_1[i]\,||\,(\Delta y \ggg r)[i]}$
        if $i = n - r - 1$ then
            $B_0 \leftarrow B_0 R$
            $B_1 \leftarrow B_1 R$
        end
        $p_0 =$
          $4^{-(i+1)}(L_0 B_0 C_0 + L_1 B_0 C_1)$
        $p_1 =$
          $4^{-(i+1)}(L_0 B_1 C_0 + L_1 B_1 C_1)$
        if $(p_0 \geq p_1$ and $c[i] = 0)$ or
          $(p_0 < p_1$ and $c[i] = 1)$ then
            $\Delta x_0[i] \leftarrow 0$
            $B \leftarrow B_0$
            $p \leftarrow p_0$
        else
            $\Delta x_0[i] \leftarrow 1$
            $B \leftarrow B_1$
            $p \leftarrow p_1$
        end
    end
    return $\Delta x_0$, $p$
```

Algorithm 4:

```
Function
  Greedy_ADP_HQR($r_1, r_2, \Delta y_0, \Delta y_1, \Delta y_2, \Delta y_3, c_0, c_1$):
    $\Delta x_3, p_0 \leftarrow$ Greedy_ADP_J
      $(r_1, \Delta y_0, \Delta y_3, c_0)$
    $\Delta x_1, p_1 \leftarrow$ Greedy_ADP_J
      $(r_2, \Delta y_2, \Delta y_1, c_1)$
    $\Delta x_0 \leftarrow \Delta y_0 \boxminus \Delta x_1$
    $\Delta x_2 \leftarrow \Delta y_2 \boxminus \Delta y_3$
    $p \leftarrow p_0 p_1$
    return $\Delta x_0, \Delta x_1, \Delta x_2, \Delta x_3, p$
```

4 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 $2^{4\mathsf{w}}$ 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 $2^{8\mathsf{w}}$ 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 3, while for the adp we used Lemma 7. In Table 3, 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$\cdot 100/2^{4\mathsf{w}}$. 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.

Table 3 Precision of the greedy strategy applied in algorithm 2 and algorithm 4 (see Section 4 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 4 Maximum, minimum and average characteristic probability for half and full quarter round, found using algorithm 2 and algorithm 4.

	(c_0, c_1)	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)	$2^{-31.11}$	$2^{-51.43}$	$2^{-41.50}$	$2^{-67.10}$	$2^{-101.04}$	$2^{-80.51}$

We recall that ChaCha state is a 16 32-bit word vector $(s_0, \ldots, s_{15})$, where $s_0, \ldots, s_3$ are initialized with constant values, $s_4, \ldots, s_{11}$ store a 256 bit key, and $s_{12}, \ldots, s_{15}$ are used for the counter and the nonce. Thus, in a single key differential attack, only this 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 4 words. Given these constrains, the differential characteristic with highest probability returned by algorithm 2 is `0x00000000 00000000 00000000 00008000 -> 0x00000800 04044040 80080080 00080080`, holding with probability 2^{-3}.

In Table 4, 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 5 and Table 6, we report, respectively, a XOR and an additive differential trail covering 4 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 [5]. 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 [5] 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 [1], 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 $s_0, \ldots, s_{11}$. Thus, algorithm 4 can only be of use for an attacker if combined with other techniques. Notice also that in Table 6, the 3 rounds differential trail from round 2 to round 4 has probability $2^{-98.4}$, which is much lower than any other 3 rounds differential trail we were able to find for XOR differences with algorithm 2.

[1] 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.

Table 5 4 rounds XOR differential trail (of ChaCha stream cipher), with relative and cumulative probability of the differential characteristics found iterating algorithm 2.

Round	Δx	Δy	Probability	
			rel.	cum.
1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00008000	00000000 00000000 00000000 00080000 00000000 00000000 00000000 01011010 00000000 00000000 00000000 80800800 00000000 00000000 00000000 00800800	$2^{-3.00}$	$2^{-3.00}$
2	00000000 00000000 00000000 00080000 00000000 00000000 00000000 01011010 00000000 00000000 00000000 80800800 00000000 00000000 00000000 00800800	08008000 80080800 10100101 00880000 10110110 01000100 10001010 20020220 10100101 08008808 88000000 80000008 00800808 00000000 08008800 80000080	$2^{-41.0}$	$2^{-44.0}$
3	08008000 80080800 10100101 00880000 10110110 01000100 10001010 20020220 10100101 08008808 88000000 80000008 00800808 00000000 08008800 80000080	80081080 01080900 80110000 0AA0280A 32223132 11121001 33221232 40510514 89010889 09080088 11080808 02A8A00A 10091018 08080888 10190008 80080088	$2^{-122.}$	$2^{-166.}$
4	80081080 01080900 80110000 0AA0280A 32223132 11121001 33221232 40510514 89010889 09080088 11080808 02A8A00A 10091018 08080888 10190008 80080088	008A0090 10888109 15044050 889128A9 65037757 70236112 46063735 1BA3B00A 84844849 12888911 81008809 22A822A8 2A098281 18189808 32819130 92008311	$2^{-210.}$	$2^{-376.}$

Table 6 4 rounds additive differential trail (of ChaCha internal permutation), with relative and cumulative probability of the differential characteristics found iterating algorithm 4.

Round	Δx	Δy	Probability	
			rel.	cum.
1	11BC469C 222C642C 3306926E DDF975C0 2DC13904 02464248 08A714E2 21458940 A27CE21C C90A2EF7 FF3E72F8 BE7AB700 0287A010 28E22301 04222F50 81010100	BFBE7FE0 34F3AE78 FFBEF378 7F7F8000 80008000 80410020 C2844104 80000C40 77BEF7C0 FF7F0000 7FFF0000 7EFB7700 85001084 C1414040 80000080 C0008000	$2^{-208.}$	$2^{-208.}$
2	BFBE7FE0 34F3AE78 FFBEF378 7F7F8000 80008000 80410020 C2844104 80000C40 77BEF7C0 FF7F0000 7FFF0000 7EFB7700 85001084 C1414040 80000080 C0008000	80000000 FFFBF800 FFBEFFC0 FF800000 00000000 00040000 80408040 00800000 80000000 00000000 7FFF8000 FF7FFF80 80000080 00000800 80008000 00008000	$2^{-77.4}$	$2^{-286.}$
3	80000000 FFFBF800 FFBEFFC0 FF800000 00000000 00040000 80408040 00800000 80000000 00000000 7FFF8000 FF7FFF80 80000080 00000800 80008000 00008000	80000000 00000000 00000000 00000000 00000000 80000000 00000000 00000000 00000000 00000000 7FFF8000 00000000 00000000 00000000 00000000 00800000	$2^{-19.0}$	$2^{-304.}$
4	80000000 00000000 00000000 00000000 00000000 80000000 00000000 00000000 00000000 00000000 7FFF8000 00000000 00000000 00000000 00000000 00800000	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 80000000	$2^{-2.00}$	$2^{-306.}$

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 6 and use the output difference as the input difference to algorithm 2, we obtain a 4 round trail of probability $2^{-(19+2+5+43)} = 2^{-69}$, and if we started from round 2 of Table 6, the corresponding 5 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.

5 Conclusions and future work

We proved exact formulas for the XDP and the ADP of the half quarter round of ChaCha (Lemma 3 and Lemma 7, 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 4 and Lemma 8, 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 1 and Table 2). 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.

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, 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.

References

1. Aumasson, J.P., Çalık, Ç., Meier, W., Özen, O., Phan, R.C.W., Varıcı, K.: Improved cryptanalysis of Skein. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 542–559. Springer (2009)
2. Aumasson, J.P., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: BLAKE2: simpler, smaller, fast as MD5. In: International Conference on Applied Cryptography and Network Security, pp. 119–135. Springer (2013)
3. Beaulieu, R., Treatman-Clark, S., Shors, D., Weeks, B., Smith, J., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE (2015)
4. Beierle, C., Biryukov, A., dos Santos, L.C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q., Biryukov, A.: Schwaemm and esch: Lightweight authenticated encryption and hashing using the sparkle permutation family. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/SPARKLE-spec.pdf
5. Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to arx ciphers. In: Annual International Cryptology Conference, pp. 329–358. Springer (2020)
6. Bernstein, D.J.: Salsa20 specification. eSTREAM Project algorithm description, http://www.ecrypt.eu.org/stream/salsa20pf.html (2005)
7. Bernstein, D.J.: ChaCha, a variant of Salsa20. In: Workshop Record of SASC, vol. 8, pp. 3–5 (2008)
8. Bernstein, D.J.: Cubehash specification (2. b. 1). Submission to NIST (2008)
9. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: New stream cipher designs, pp. 84–97. Springer (2008)
10. Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Cryptographers' Track at the RSA Conference, pp. 227–250. Springer (2014)

11. Biryukov, A., Velichkov, V., Le Corre, Y.: Automatic search for the best trails in ARX: application to block cipher SPECK. In: International Conference on Fast Software Encryption, pp. 289–310. Springer (2016)
12. 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.: Shabal, a submission to nist's cryptographic hash algorithm competition. Submission to NIST (2008)
13. Chittenden, E.W.: On the number of paths in a finite partially ordered set. Amer. Math. Monthly **54**, 404–405 (1947)
14. Daum, M.: Cryptanalysis of hash functions of the md4-family. Ph.D. thesis, Ruhr University Bochum (2005)
15. De Canniere, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 1–20. Springer (2006)
16. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family. Submission to NIST (round 3) **7**(7.5), 3 (2010)
17. Gligoroski, D., Klima, V., Knapskog, S.J., El-Hadedy, M., Amundsen, J.: Cryptographic hash function blue midnight wish. In: 2009 Proceedings of the 1st International Workshop on Security and Communication Networks, pp. 1–8. IEEE (2009)
18. Hong, D., Lee, J.K., Kim, D.C., Kwon, D., Ryu, K.H., Lee, D.G.: Lea: A 128-bit block cipher for fast encryption on common processors. In: International Workshop on Information Security Applications, pp. 3–27. Springer (2013)
19. Leurent, G., Bouillaguet, C., Fouque, P.A.: Simd is a message digest. Submission to the NIST SHA-3 Competition (Round 2) (2009)
20. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: FSE 2001, Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer (2001)
21. Lipmaa, H., Wallén, J., Dumas, P.: On the Additive Differential Probability of Exclusive–Or. In: FSE 2004, Lecture Notes in Computer Science, vol. 3017, pp. 317–331. Springer (2004)
22. Mehner, C.E.: Limdolen (2019). `https://github.com/cem-/limdolen/blob/master/Documents/Limdolen%20Specification.pdf`
23. Mouha, N., De Canniere, C., Indesteege, S., Preneel, B.: Finding collisions for a 45-step simplified HAS-V. In: International Workshop on Information Security Applications, pp. 206–225. Springer (2009)
24. Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: an efficient MAC algorithm for 32-bit microcontrollers. In: International Conference on Selected Areas in Cryptography, pp. 306–323. Springer (2014)
25. Mouha, N., Velichkov, V., De Canniere, C., Preneel, B.: The differential analysis of S-functions. In: International Workshop on Selected Areas in Cryptography, pp. 36–56. Springer (2010)
26. NIST: Hash Functions - SHA-3 Project. `https://csrc.nist.gov/projects/hash-functions/sha-3-project`
27. NIST: Lightweight Cryptography Standardization Process. `https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates`, Accessed 07/28/2020
28. Rivest, R.L.: The rc5 encryption algorithm. In: International Workshop on Fast Software Encryption, pp. 86–96. Springer (1994)
29. Velichkov, V., Mouha, N., De Canniere, C., Preneel, B.: The additive differential probability of ARX. In: International Workshop on Fast Software Encryption, pp. 342–358. Springer (2011)
30. Velichov, V.e.a.: UNAF: A Special Set of Additive Differences with Application to the Differential Analysis of ARX. In: FSE 2012, Lecture Notes in Computer Science, vol. 7549, pp. 287–305. Springer (2012)
31. Wheeler, D.J., Needham, R.M.: TEA, a tiny encryption algorithm. In: International Workshop on Fast Software Encryption, pp. 363–366. Springer (1994)