Improving the Security of ChaCha against Differential-Linear Cryptanalysis

Murilo Coutinho [1,2], **Iago Passos** [2], **Rafael T. de Sousa Jr.** [2], **Fábio Borges** [3]

[1]Centro de Pesquisa e Desenvolvimento para a Segurança da Comunicações (CEPESC)
Agência Brasileira de Inteligência
Brasília, Brasil.

[2]Departamento de Engenharia Elétrica
Universidade de Brasília (UnB)
Brasília, Brasil.

[3]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis, Brasil

Abstract. *The stream cipher ChaCha has received a lot of attention and recently is being used as a new cipher suite in TLS 1.3, as a random number generator for operating systems (Linux, FreeBSD, OpenBSD, NetBSD, and DragonFly BSD), a proposed standardization in RFC 7634 for use IKE and IPsec, and by the WireGuard VPN protocol. Because of that, it is very important to understand and study the security of this algorithm. Previous works showed that it is possible to break up to 7 of the 20 rounds of ChaCha. In this paper, we show that a simple modification in the algorithm, namely changing the rotation distances in the Quarter Round Function, makes ChaCha more secure against all the most effective known attacks without any loss in performance. In fact, we show that with these changes, it is only possible to break up to 6 rounds of ChaCha. Therefore, it would be no longer possible to break 7 rounds of ChaCha with the best-known attacks.*

1. Introduction

In 2008, Bernstein proposed the stream cipher Salsa20 [Bernstein 2008b] as a contender to the eStream competition. Later, Bernstein proposed some modifications to Salsa20 to improve diffusion and security, creating a new stream cipher, which he called ChaCha20 [Bernstein 2008a]. Although Salsa20 was one of the winners of the eStream competition, ChaCha20 has received much more attention through the years. Nowadays, we see the usage of this cipher in several projects and applications.

ChaCha, along Poly1305 [Bernstein 2005], is in one of the cipher suits of the new TLS 1.3 [Langley et al. 2016], which is actually used by Google on both Chrome and Android. ChaCha is used not only in TLS but in many other protocols such as SSH, Noise, and S/MIME 4.0. In addition, the RFC 7634 proposes the use of ChaCha in IKE and IPsec. ChaCha is used not only for encryption but also as a random number generator, for example, in any operating system running Linux kernel 4.8 or newer [Torvalds 2016]. Additionally, ChaCha is used in several applications, for example, WireGuard (VPN), Keepass (password manager), and Veracrypt (disk encryption). See [IANIX 2020] for a huge list of applications, protocols, and libraries using ChaCha20.

Since ChaCha is so heavily used, it is very important to fully understand its security. Indeed, the cryptanalysis of ChaCha is well understood and several authors studied its security [Aumasson et al. 2008, Hernandez-Castro et al. 2008, Crowley 2006, Fischer et al. 2006, Ishiguro et al. 2011, Maitra 2016, Maitra et al. 2015, Mouha and Preneel 2013, Choudhuri and Maitra 2016, Shi et al. 2012, Tsunoo et al. 2007, Dey and Sarkar 2017, Dey et al. 2019, Ding 2019, Coutinho and Neto 2020].

In this work, we study the most important attacks against ChaCha and show that it is possible to improve its security by changing the rotation distances in the Quarter Round Function (QRF). In fact, to this day, the best attack against ChaCha works on only 7 rounds of the 20 provided by the algorithm. However, using the proposed modification, we show that the security is enhanced, limiting the best attack to succeed on only 6 rounds.

This work is organized as follows: in Section 2, we define the notation used in the paper and define the ChaCha algorithm. In Section 3, we review the best attacks available against ChaCha. In Section 4, we provide an intensive analysis of the security of the algorithm for all combinations of rotation distances showing that it is possible to improve the security of ChaCha. In Section 5, we provide a security comparison of the original ChaCha, and its new proposed version. Finally, in Section 6, we present the conclusions.

2. Specifications and Preliminaries

In this section, we define the notation that we will use throughout the paper in Table 1. Afterwards, we define the algorithm ChaCha.

Bernstein proposed the stream cipher Salsa [Bernstein 2008b] to the *eStream* competition and later Bernstein proposed ChaCha [Bernstein 2008a] as an improvement of Salsa. ChaCha consists of a series of ARX (addition, rotation, and XOR) operations on 32-bit words, being highly efficient in software and hardware. Salsa operates on a state of 64 bytes, organized as a 4×4 matrix with 32-bit integers, initialized with a 256-bit key $k_0, k_1, ..., k_7$, a 64-bit nonce v_0, v_1 and a 64-bit counter t_0, t_1 (we may also refer to the nonce and counter words as the initialization vector – IV), and 4 constants $c_0 = 0\text{x}61707865$, $c_1 = 0\text{x}3320646e$, $c_2 = 0\text{x}79622d32$ and $c_3 = 0\text{x}6b206574$. For ChaCha, we have the following initial state matrix:

$$X^{(0)} = \begin{pmatrix} x_0^{(0)} & x_1^{(0)} & x_2^{(0)} & x_3^{(0)} \\ x_4^{(0)} & x_5^{(0)} & x_6^{(0)} & x_7^{(0)} \\ x_8^{(0)} & x_9^{(0)} & x_{10}^{(0)} & x_{11}^{(0)} \\ x_{12}^{(0)} & x_{13}^{(0)} & x_{14}^{(0)} & x_{15}^{(0)} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & t_1 & v_0 & v_1 \end{pmatrix}. \tag{1}$$

The state matrix is modified in each round by a *Quarter Round Function* (QRF), named $QR_{r_1,r_2,r_3,r_4}(a, b, c, d)$, which receives and updates 4 integers in the following way:

$$\begin{array}{lllllllll} a & \leftarrow & a+b; & d & \leftarrow & d \oplus a; & d & \leftarrow & d \lll r_1; \\ c & \leftarrow & c+d; & b & \leftarrow & b \oplus c; & b & \leftarrow & b \lll r_2; \\ a & \leftarrow & a+b; & d & \leftarrow & d \oplus a; & d & \leftarrow & d \lll r_3; \\ c & \leftarrow & c+d; & b & \leftarrow & b \oplus c; & b & \leftarrow & b \lll r_4; \end{array} \tag{2}$$

Notation	Description
X	a 4×4 state matrix of the cipher of 16 words
$X^{(0)}$	initial state matrix
$X^{(R)}$	state matrix after application of R round functions
Z	output of an algorithm, $Z = X + X^{(R)}$
$x_i^{(R)}$	i^{th} word of the state matrix $X^{(R)}$ (words arranged in row major)
$x_{i,j}^{(R)}$	j^{th} bit of i^{th} word of the state matrix $X^{(R)}$
$x + y$	addition of x and y modulo 2^{32}
$x - y$	subtraction of x and y modulo 2^{32}
$x \oplus y$	bitwise XOR of x and y
$x \lll n$	rotation of x by n bits to the left
$x \ggg n$	rotation of x by n bits to the right
Δx	XOR difference of x and x'. $\Delta x = x \oplus x'$
$\Delta_i^{(R)}$	differential $\Delta_i^{(R)} = x_i^{(R)} \oplus {x'}_i^{(R)}$
$\Delta_{i,j}^{(R)}$	differential $\Delta_{i,j}^{(R)} = x_{i,j}^{(R)} \oplus {x'}_{i,j}^{(R)}$
$\mathbb{P}(E)$	probability of occurrence of an event E
$\mathbb{B}(E)$	bias of an event E, thus $\mathbb{B}(E) = 2\mathbb{P}(E) - 1$
$\varepsilon_{(x_1 \oplus ... \oplus x_m)}$	bias of event $E = \{\Delta x_1 \oplus ... \oplus \Delta x_m = 0\}$
$\mathcal{I}D$	input differential
$\mathcal{O}D$	output differential

Table 1. Notation.

One round of ChaCha is defined as 4 applications of $QR_{16,12,8,7}$. There is a difference, however, between odd and even rounds. For odd rounds, when $r \in \{1, 3, 5, 7, ...\}$, $X^{(r)}$ is defined from $X^{(r-1)}$, as

$$\begin{array}{rcl}
\left(x_0^{(r)}, x_4^{(r)}, x_8^{(r)}, x_{12}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_0^{(r-1)}, x_4^{(r-1)}, x_8^{(r-1)}, x_{12}^{(r-1)}\right) \\
\left(x_1^{(r)}, x_5^{(r)}, x_9^{(r)}, x_{13}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_1^{(r-1)}, x_5^{(r-1)}, x_9^{(r-1)}, x_{13}^{(r-1)}\right) \\
\left(x_2^{(r)}, x_6^{(r)}, x_{10}^{(r)}, x_{14}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_2^{(r-1)}, x_6^{(r-1)}, x_{10}^{(r-1)}, x_{14}^{(r-1)}\right) \\
\left(x_3^{(r)}, x_7^{(r)}, x_{11}^{(r)}, x_{15}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_3^{(r-1)}, x_7^{(r-1)}, x_{11}^{(r-1)}, x_{15}^{(r-1)}\right)
\end{array},$$

and, for even rounds $r \in \{2, 4, 6, 8, ...\}$, as

$$\begin{array}{rcl}
\left(x_0^{(r)}, x_5^{(r)}, x_{10}^{(r)}, x_{15}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_0^{(r-1)}, x_5^{(r-1)}, x_{10}^{(r-1)}, x_{15}^{(r-1)}\right) \\
\left(x_1^{(r)}, x_6^{(r)}, x_{11}^{(r)}, x_{12}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_1^{(r-1)}, x_6^{(r-1)}, x_{11}^{(r-1)}, x_{12}^{(r-1)}\right) \\
\left(x_2^{(r)}, x_7^{(r)}, x_8^{(r)}, x_{13}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_2^{(r-1)}, x_7^{(r-1)}, x_8^{(r-1)}, x_{13}^{(r-1)}\right) \\
\left(x_3^{(r)}, x_4^{(r)}, x_9^{(r)}, x_{14}^{(r)}\right) & \leftarrow & QR_{16,12,8,7}\left(x_3^{(r-1)}, x_4^{(r-1)}, x_9^{(r-1)}, x_{14}^{(r-1)}\right)
\end{array}.$$

The algorithm ChaCha20/R is then defined as the sum of the initial state with the state obtained after R rounds of operations $Z = X + X^{(R)}$. One should note that it is possible to parallelize each application of the QRF on each round and that each round is reversible. Hence, we can compute $X^{(r-1)}$ from $X^{(r)}$. For more information on ChaCha, we refer to [Bernstein 2008a].

3. Cryptanalysis of ChaCha

Several authors studied the security and diffusion of both Salsa and ChaCha [Aumasson et al. 2008, Hernandez-Castro et al. 2008, Crowley 2006, Coutinho et al. 2020, Fischer et al. 2006, Ishiguro et al. 2011, Maitra 2016, Maitra et al. 2015, Mouha and Preneel 2013, Choudhuri and Maitra 2016, Shi et al. 2012, Tsunoo et al. 2007, Dey and Sarkar 2017, Dey et al. 2019, Ding 2019, Coutinho and Neto 2020] which show weaknesses in the reduced rounds of the ciphers. The attacks in most cases, apply some input differences to the initial state to observe output differences after certain rounds. Once one of them can proceed a few rounds forward as above, it may be possible to invert a few rounds from a final state to obtain further non-randomness. Crowley introduced the cryptanalysis of Salsa [Crowley 2006] in 2006, but the most important cryptanalysis in this regard was proposed by Aumasson et al. at FSE 2008 [Aumasson et al. 2008] with the introduction of Probabilistic Neutral Bits (PNBs). After that, several authors proposed small enhancements on the attack of Aumasson et al. The work by Shi et al [Shi et al. 2012] introduced the concept of Column Chaining Distinguisher (CCD) to achieve some incremental advancements over [Aumasson et al. 2008] for both Salsa and ChaCha. Maitra, Paul, and Meier [Maitra et al. 2015] studied an interesting observation about round reversal of Salsa, but no significant cryptanalytic improvement could be obtained using this method. Maitra [Maitra 2016] used a technique of Chosen IVs to obtain certain improvements over existing results. Dey and Sarkar [Dey and Sarkar 2017] showed how to choose values for the PNB to improve the attack. The best improvement for the technique was given by Choudhuri and Maitra [Choudhuri and Maitra 2016] using the technique of differential-linear cryptanalysis and exploring the mathematical structure of both Salsa and ChaCha to find differential characteristics with much higher biases. Later Coutinho and Neto improved Choudhuri and Maitra's attack by showing better linear approximations [Coutinho and Neto 2020].

Here, we analyze and improve the security of ChaCha by first replicating and checking the results of the attack of Aumasson [Aumasson et al. 2008], Choudhuri and Maitra [Choudhuri and Maitra 2016], and Coutinho and Neto [Coutinho and Neto 2020] and then applying the technique against different rotation combinations for the QRF. We chose these attacks since they are the most important works on the cryptanalysis of Salsa and ChaCha to this day.

3.1. Probabilistic Neutral Bits

This section reviews the attack of Aumasson [Aumasson et al. 2008]. The attack first identifies good choices of truncated differentials, then it uses probabilistic backwards computation with the notion of Probabilistic Neutral Bits (PNB), and, finally, it estimates the complexity of the attack.

Let Δ_i^R be the differential for the i_{th} word of state matrix $X^{(R)}$, thus $\Delta_i^R = x_i^R \oplus {x'}_i^R$; and let $\Delta_{i,j}^R$ be the differential for the j_{th} bit of the i_{th} word, thus $\Delta_{i,j}^R = x_{i,j}^R \oplus {x'}_{i,j}^R$. In [Aumasson et al. 2008] the input differential $\mathcal{I}D$ is defined for a single-bit difference $\Delta_{i,j}^0 = 1$ and consider a single-bit output difference $\mathcal{O}D$ after r rounds $\Delta_{p,q}^r$, such differential is denoted $(\Delta_{p,q}^r | \Delta_{i,j}^0)$. For a fixed key, the bias ε_d of the $\mathcal{O}D$ is defined by $\mathbb{P}_{v,t}(\Delta_{p,q}^r = 1 | \Delta_{i,j}^0) = \frac{1}{2}(1 + \varepsilon_d)$, where the probability holds over all nonces v and coun-

ters t. Furthermore, considering the key as a random variable, we denote the median value of ε_d by $\varepsilon_d^\star$. Hence, for half of the keys, this differential have a bias of at least $\varepsilon_d^\star$.

Now, assume that the differential $(\Delta_{p,q}^r|\Delta_{i,j}^0)$ of bias ε_d is fixed, and we observe outputs Z and Z' of $R = l + r$ rounds for nonce v, counter t and unknown key k. If we guess the key k we can invert l rounds of the algorithm to get $X^{(r)}$ and $X'^{(r)}$ and compute $\Delta_{p,q}^r$, let f be the function which executes this procedure. Hence $f(k, v, t, Z, Z') = \Delta_{p,q}^r$ and we expect that

$$\mathbb{P}(f(\hat{k}, v, t, Z, Z') = 1) = \begin{cases} \frac{1}{2}(1 + \varepsilon_d), & \text{if } \hat{k} = k \\ 0.5, & \text{if } \hat{k} \neq k \end{cases},$$

thus, if we have several pairs of Z and Z', it is possible to test our guesses for k.

Thus, we can search only over a subkey of $m = 256 - n$ bits, provided we can find a function g that approximates f but only uses m key bits as input. Then, let $\bar{k}$ correspond to the subkey of m bits of key k and let f to be correlated to g with bias ε_a i.e., $\mathbb{P}(f(k, v, t, Z, Z') = g(\bar{k}, v, t, Z, Z')) = \frac{1}{2}(1 + \varepsilon_a)$.

If we denote the bias of g by ε, i.e. $P(g(\bar{k}, v, t, Z, Z') = 1) = \frac{1}{2}(1 + \varepsilon)$, and $\varepsilon^\star$ the median bias of g over all keys, we can approximate ε by $\varepsilon_d\varepsilon_a$. The problem that remains is how to efficiently find such a function g. In [Aumasson et al. 2008], this is done by first identifying key bits that have little influence on the result of $f(k, v, t, Z, Z')$, these are called *probabilistic neutral bits* (PNBs). This is done by defining the *neutrality measure* $\gamma_{i,j}$ of a key bit $k_{i,j}$.

After computing $\gamma_{i,j}$ (see [Aumasson et al. 2008] for a method of estimation), for all $i = (0, 1, ..., 7)$ and $j = (0, 1, ..., 31)$, we can define the set of significant key bits as $\Psi = \{(i, j) : \gamma_{i,j} \leq \gamma\}$ where γ is a threshold value, and then define our approximation g as $g(k_\Psi, v, t, Z, Z') = f(k^*, v, t, Z, Z')$ where k_Ψ is defined as the subkey with key bits in the set Ψ and k^* is computed from k_Ψ by setting $k_{i,j} = 0$ for all $(i, j) \notin \Psi$. Thus, the attack can be evaluated with the following steps:

1. Compute a good differential for r rounds $(\Delta_{p,q}^r|\Delta_{i,j}^0)$ by estimating the bias ε_d for all single-bit $\mathcal{I}D$ with several random combinations of keys, nonces, and counters.
2. Empirically estimate the neutrality measure $\gamma_{r,s}$ for each key bit $k_{r,s}$.
3. Construct the function g by setting all key bit such that $\gamma_{r,s} > \gamma$ to zero and estimate the median bias $\varepsilon^\star$ by empirically measuring bias of g using many randomly chosen keys, nonces, and counters.
4. Estimate the data and time complexity of the attack.

We refer to [Aumasson et al. 2008] for further information about the estimation of the data and time complexity of the attack and for further details on the described technique.

3.2. Multi-bit Differentials

This section reviews the attack of Choudhuri and Maitra [Choudhuri and Maitra 2016] and later improved by Coutinho and Neto [Coutinho and Neto 2020]. The attack first identifies linear relationships between the bits of two successive rounds of ChaCha. From these relationships, it is possible to compute single bit differentials for r rounds, obtaining

a distinguisher for $r+1$ rounds, which reduces the complexities of the attacks described in Section 3.1. The first step is to write ChaCha's QRF (Eq. (2)) only using XOR bit by bit operations.

Then, is possible to compute $x_{a,i}^{(m-1)}, x_{b,i}^{(m-1)}, x_{c,i}^{(m-1)}$, and $x_{d,i}^{(m-1)}$ in terms of $x_{a,i}^{(m)}, x_{b,i}^{(m)}, x_{c,i}^{(m)}, x_{d,i}^{(m)}, C_i^1, C_i^2, C_i^3$, and C_i^4, we get:

$$\begin{array}{rcl} x_{a,i}^{(m-1)} & = & x_{a,i}^{(m)} \oplus x_{b,i+r_4}^{(m)} \oplus x_{b,i+r_2+r_4}^{(m)} \oplus x_{c,i+r_2}^{(m)} \oplus x_{d,i}^{(m)} \oplus C_i^4 \oplus C_i^3 \oplus C_i^1 \\ x_{b,i}^{(m-1)} & = & x_{b,i+r_2+r_4}^{(m)} \oplus x_{c,i+r_2}^{(m)} \oplus x_{d,i}^{(m)} \oplus x_{c,i}^{(m)} \oplus C_i^4 \\ x_{c,i}^{(m-1)} & = & x_{d,i}^{(m)} \oplus x_{c,i}^{(m)} \oplus x_{d,i+r_3}^{(m)} \oplus x_{a,i}^{(m)} \oplus C_i^2 \oplus C_i^4 \\ x_{d,i}^{(m-1)} & = & x_{d,i+r_1+r_3}^{(m)} \oplus x_{a,i+r_1}^{(m)} \oplus x_{a,i}^{(m)} \oplus x_{c,i}^{(m)} \oplus x_{b,i+r_4}^{(m)} \oplus C_i^3 \end{array} \quad (3)$$

where C_i^1, C_i^2, C_i^3, and C_i^4 denote the i-th carry bit of the first, second, third, and fourth additions contained in the QRF, respectively. Since we have that $C_0^1 = C_0^2 = C_0^3 = C_0^4 = 0$ by definition, the following lemma has been proved in [Choudhuri and Maitra 2016]:

Lemma 1. *Let*

$$\begin{array}{rcl} \Delta A^{(m)} & = & \Delta x_{\alpha,0}^{(m)} \oplus \Delta x_{\beta,r_4}^{(m)} \oplus \Delta x_{\beta,r_2+r_4}^{(m)} \oplus \Delta x_{\gamma,r_2}^{(m)} \oplus \Delta x_{\delta,0}^{(m)} \\ \Delta B^{(m)} & = & \Delta x_{\beta,r_2+r_4}^{(m)} \oplus \Delta x_{\gamma,0}^{(m)} \oplus \Delta x_{\gamma,r_2}^{(m)} \oplus \Delta x_{\delta,0}^{(m)} \\ \Delta C^{(m)} & = & \Delta x_{\delta,0}^{(m)} \oplus \Delta x_{\gamma,0}^{(m)} \oplus \Delta x_{\delta,r_3}^{(m)} \oplus \Delta x_{\alpha,0}^{(m)} \\ \Delta D^{(m)} & = & \Delta x_{\delta,r_1+r_3}^{(m)} \oplus \Delta x_{\alpha,r_1}^{(m)} \oplus \Delta x_{\alpha,0}^{(m)} \oplus \Delta x_{\gamma,0}^{(m)} \oplus \Delta x_{\beta,r_4}^{(m)} \end{array}$$

After m rounds of ChaCha, the following holds:

$$\left|\varepsilon_{(A^{(m)})}\right| = \left|\varepsilon_{x_{\alpha,0}^{(m-1)}}\right|, \quad \left|\varepsilon_{(B^{(m)})}\right| = \left|\varepsilon_{x_{\beta,0}^{(m-1)}}\right|,$$

$$\left|\varepsilon_{(C^{(m)})}\right| = \left|\varepsilon_{x_{\gamma,0}^{(m-1)}}\right|, \quad \left|\varepsilon_{(D^{(m)})}\right| = \left|\varepsilon_{x_{\delta,0}^{(m-1)}}\right|,$$

The tuples $(\alpha, \beta, \gamma, \delta)$ vary depending on whether m is odd or even

1. *m is odd:* $(\alpha, \beta, \gamma, \delta) \in \{(0,4,8,12), (1,5,9,13), (2,6,10,14), (3,7,11,15)\}$
2. *m is even:* $(\alpha, \beta, \gamma, \delta) \in \{(0,5,10,15), (1,6,11,12), (2,7,8,13), (3,4,9,14)\}$

Using Lemma 1, it is possible to find differentials for $m-1$ rounds and then use the bias to make a distinguisher for a linear combination in m rounds, improving the attack. It is possible to go one round further by using linear equations that hold with high probability. The following lemma defines useful linear relationships, see [Choudhuri and Maitra 2016] for a proof of this lemma.

Lemma 2. *For ChaCha, each of the following holds with probability $\frac{1}{2}(1+\frac{1}{2})$:*

$$\begin{array}{rl} x_{8,0}^{(3)} = & x_{13,r_1+r_3}^{(5)} \oplus x_{1,r_1}^{(5)} \oplus x_{1,0}^{(5)} \oplus x_{9,0}^{(5)} \oplus x_{5,r_4}^{(5)} \oplus x_{12,0}^{(5)} \oplus x_{8,0}^{(5)} \oplus \\ & x_{12,r_3}^{(5)} \oplus x_{0,0}^{(5)} \oplus x_{2,0}^{(5)} \oplus x_{6,r_4}^{(5)} \oplus x_{6,r_2+r_4}^{(5)} \oplus x_{10,r_2}^{(5)} \oplus x_{14,0}^{(5)} \oplus \\ & x_{13,2r_3+r1}^{(5)} \oplus x_{1,r_1+r_3}^{(5)} \oplus x_{1,r_3}^{(5)} \oplus x_{9,r_3}^{(5)} \oplus x_{5,r_3+r_4}^{(5)} \oplus x_{5,r_3+r_4-1}^{(5)} \oplus x_{9,r_3-1}^{(5)} \end{array}$$

$$\begin{array}{rl} x_{9,0}^{(3)} = & x_{14,r_1+r_3}^{(5)} \oplus x_{2,r_1}^{(5)} \oplus x_{2,0}^{(5)} \oplus x_{10,0}^{(5)} \oplus x_{6,r_4}^{(5)} \oplus x_{13,0}^{(5)} \oplus x_{9,0}^{(5)} \oplus \\ & x_{13,r_3}^{(5)} \oplus x_{1,0}^{(5)} \oplus x_{3,0}^{(5)} \oplus x_{7,r_4}^{(5)} \oplus x_{7,r_2+r_4}^{(5)} \oplus x_{11,r_2}^{(5)} \oplus x_{15,0}^{(5)} \oplus \\ & x_{14,2r_3+r1}^{(5)} \oplus x_{2,r_1+r_3}^{(5)} \oplus x_{2,r_3}^{(5)} \oplus x_{10,r_3}^{(5)} \oplus x_{6,r_3+r_4}^{(5)} \oplus x_{6,r_3+r_4-1}^{(5)} \oplus x_{10,r_3-1}^{(5)} \end{array}$$

$$
\begin{aligned}
x^{(3)}_{10,0} = \; & x^{(5)}_{15,r_1+r_3} \oplus x^{(5)}_{3,r_1} \oplus x^{(5)}_{3,0} \oplus x^{(5)}_{11,0} \oplus x^{(5)}_{7,r_4} \oplus x^{(5)}_{14,0} \oplus x^{(5)}_{10,0} \oplus \\
& x^{(5)}_{14,r_3} \oplus x^{(5)}_{2,0} \oplus x^{(5)}_{0,0} \oplus x^{(5)}_{4,r_4} \oplus x^{(5)}_{4,r_2+r_4} \oplus x^{(5)}_{8,r_2} \oplus x^{(5)}_{12,0} \oplus \\
& x^{(5)}_{15,2r_3+r1} \oplus x^{(5)}_{3,r_1+r_3} \oplus x^{(5)}_{3,r_3} \oplus x^{(5)}_{11,r_3} \oplus x^{(5)}_{7,r_3+r_4} \oplus x^{(5)}_{7,r_3+r_4-1} \oplus x^{(5)}_{11,r_3-1} \\
x^{(3)}_{11,0} = \; & x^{(5)}_{12,r_1+r_3} \oplus x^{(5)}_{0,r_1} \oplus x^{(5)}_{0,0} \oplus x^{(5)}_{8,0} \oplus x^{(5)}_{4,r_4} \oplus x^{(5)}_{15,0} \oplus x^{(5)}_{11,0} \oplus \\
& x^{(5)}_{15,r_3} \oplus x^{(5)}_{3,0} \oplus x^{(5)}_{1,0} \oplus x^{(5)}_{5,r_4} \oplus x^{(5)}_{5,r_2+r_4} \oplus x^{(5)}_{9,r_2} \oplus x^{(5)}_{13,0} \oplus \\
& x^{(5)}_{12,2r_3+r1} \oplus x^{(5)}_{0,r_1+r_3} \oplus x^{(5)}_{0,r_3} \oplus x^{(5)}_{8,r_3} \oplus x^{(5)}_{4,r_3+r_4} \oplus x^{(5)}_{4,r_3+r_4-1} \oplus x^{(5)}_{8,r_3-1}
\end{aligned}
$$

Latter Coutinho and Neto showed the following Lemmas [Coutinho and Neto 2020], which they used to significantly improve the attacks against ChaCha

Lemma 3. *Let*

$$\Delta E^{(m)} = \Delta x^{(m)}_{\alpha,0} \oplus \Delta x^{(m)}_{\beta,r_4} \oplus \Delta x^{(m)}_{\gamma,0}$$

After m rounds of ChaCha, the following holds:

$$\left|\varepsilon_{(E^{(m)})}\right| = \left|\varepsilon_{(x^{(m-1)}_{\alpha,0} \oplus x^{(m-1)}_{\beta,0})}\right|$$

The tuples (α, β, γ) vary depending on whether m is odd or even.

- *Case I. m odd:* $(\alpha, \beta, \gamma) \in \{(0,4,8),(1,5,9),(2,6,10),(3,7,11)\}$
- *Case II. m even:* $(\alpha, \beta, \gamma) \in \{(0,5,10),(1,6,11),(2,7,8),(3,4,9)\}$

Lemma 4. *When m is odd, the following holds with probability $\frac{1}{2}(1+\frac{1}{2})$*

$$
\begin{aligned}
x^{(m-2)}_{3,0} \oplus x^{(m-2)}_{4,0} = \; & x^{(m)}_{1,0} \oplus x^{(m)}_{3,0} \oplus x^{(m)}_{4,2r_4+r_2} \oplus x^{(m)}_{7,r_4} \oplus x^{(m)}_{7,r_2+r_4} \oplus \\
& x^{(m)}_{8,r_4} \oplus x^{(m)}_{8,r_2+r_4} \oplus x^{(m)}_{9,0} \oplus x^{(m)}_{11,r_2} \oplus x^{(m)}_{12,r_4-1} \oplus \\
& x^{(m)}_{12,r_4} \oplus x^{(m)}_{13,0} \oplus x^{(m)}_{13,r_3} \oplus x^{(m)}_{15,0}.
\end{aligned}
$$

4. Improving ChaCha

In [Bernstein 2008a], Bernstein justify the choice of the rotation distances 16, 12, 8, 7 with the argument:

> "The above code also shows a much less important difference between ChaCha and Salsa20: I changed the rotation distances 7, 9, 13, 18 to 16, 12, 8, 7. The difference in security appears to be negligible: 7, 9, 13, 18 appears marginally better in some diffusion measures, and 16, 12, 8, 7 appears marginally better in others, but the margins are tiny, far smaller than the presumed inaccuracy of the diffusion measures as predictors of security. The change boosts speed slightly on some platforms while making no difference on other platforms".

Naturally, the attacks against ChaCha were unknown by the time of its publication. Therefore, one might expect that there could exist a distinct set of rotation distances such that ChaCha has better security against differential and linear cryptanalysis. Thus, our approach to improve the security of ChaCha consists in testing all combination of rotation distances to find if there is a set that is more secure.

4.1. Testing Differential Paths

In [Aumasson et al. 2008], the authors presented attacks for 6 and 7 rounds of ChaCha, however, both attacks use differential paths for $r = 3$ rounds. Leveraging this fact, our first test consists in computing the best differential path for 3 rounds of ChaCha considering all single-bit input differentials and all output bits. In other words, we estimated the bias ε_d for all combinations of differentials $(\Delta^r_{p,q}|\Delta^0_{i,j})$ for each combination of rotations distances. Since each rotation have 32 values and since we have 128 input differentials and 512 output bits, we conclude that we computed $128 \times 512 \times 32^4 = 2^{36}$ different biases.

More specifically, we used Algorithm 1 to compute the highest bias for all combinations of rotations distances. Unfortunately, since we are performing an empirical estimation, we need to execute the same procedure several times for each input differential. To reduce the number of necessary computations we used the same key, nonce, and counter for all output bits simultaneously. To test all combinations of rotation distances, we must execute Algorithm 1 2^{20} times. In addition, we defined the number of keys tested $N_k = 32$ and the number of tests per key $N_t = 1024$. Therefore, we have 2^{43} computation in total for 3 rounds of ChaCha. To achieve this amount of computation, we implemented Algorithm 1 in CUDA and executed it on a NVIDIA Quadro 4000 GPU, which required approximately 6 days of computation.

Algorithm 1 Differential Path Computation

procedure INPUT: A SET OF ROTATION DISTANCES r_1, r_2, r_3 AND r_4, THE NUMBER OF KEYS TESTED N_k, THE NUMBER OF TESTS PER KEY N_t
  $\varepsilon_d = 0$
  for each input differential $\Delta^0_{i,j}$ **do**
    $S = 0$
    for a from 1 to N_k **do**
      Generate random key k
      for b from 1 to N_t **do**
        Generate random nonce v
        Generate random counter t
        Initialize $X^{(0)}$ from k, v, t
        Compute $X^{(3)}$ from $X^{(0)}$
        Compute $X'^{(0)}$ from $X^{(0)}$ by flipping the bit $x^{(0)}_{i,j}$
        Compute $X'^{(3)}$ from $X'^{(0)}$
        $W = X^{(3)} \oplus X'^{(3)}$
        Convert W into a array of bits B
        $S = S + B$
    $m = \max(|2 \times S/(N_t N_k) - 1|)$
    if $m > \varepsilon_d$ **then**
      $\varepsilon_d = m$
  return ε_d

The results revealed that the bias of the differential path varies significantly for each combination of rotation distances. For example, if we set $r_1 = 0$ and $r_4 = 0$ (in other words, remove these rotations), we get the biases presented in Figure 1, which are

all equal, or very close to one. In comparison, if we set $r_1 = 16$ and $r_4 = 7$, we get much better results (see Figure 2) although there are still some very high biases for certain values of r_2 and r_3. Notice in Figure 2 that the maximum bias found for ChaCha with the original rotation distances $16, 12, 8, 7$ is not the best choice, since there are several combinations with smaller biases.

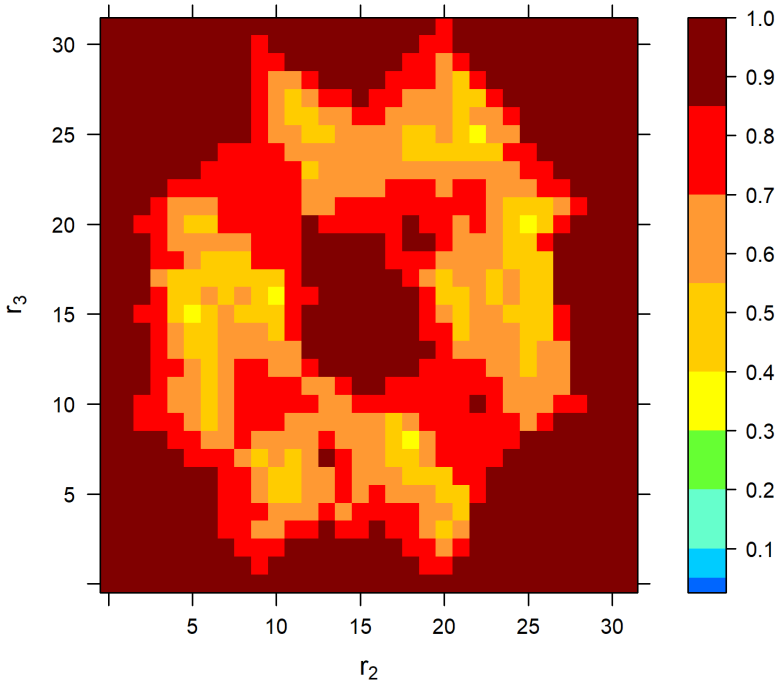

Figure 1. The biases were obtained for 3 rounds of ChaCha using rotations $r_1 = r_4 = 0$ and varying all values for r_2 and r_3. The color of the figure indicates the maximum absolute bias obtained for each combination of rotations. These are very poor results since that all biases are close to 1.

4.2. Finding Probabilistic Neutral Bits

From the results described in the previous section, we reduced the number of rotation distances under analysis by selecting the minimum bias available in the data and all the remaining biases that were statistically close to this minimum value. In total, remained 3162 combinations of rotation distances and the original set of rotation distances of ChaCha was not among these selected values. With the reduced set, we repeated the test of differential paths of the previous section but now with an increased value of $N_k = 256$, to achieve better precision.

The complexity of the attack depends not only on the bias of the differential path but also on the number of PNB. Thus, we performed another test to gather data about the behavior of PNB for each combination of rotation distances. It turns out that the computation necessary for this test increases significantly because we must test not only for each pair of input-output bits but also for each key bit individually. Fortunately, we empirically verified that, for ChaCha, the set of neutral bits are roughly the same for a particular output bit for any input bit. Thus, we can drastically reduce the necessary computation by randomly choosing the single-bit input differential. We computed the average neutrality for each output bit by performing 2^{16} iterations for each key bit, obtaining an array of 512

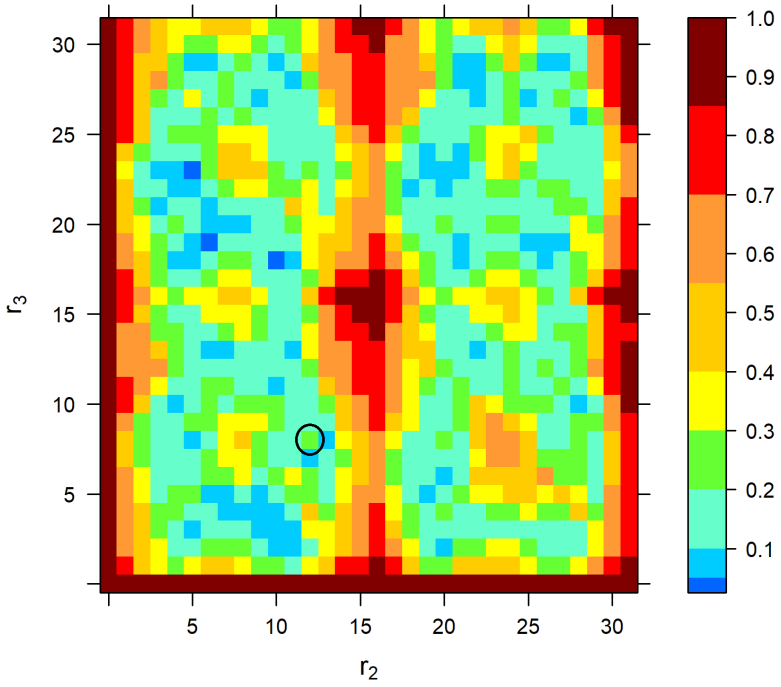

Figure 2. The biases we obtained for 3 rounds of ChaCha using rotations $r_1 = 16$ and $r_4 = 7$ and varying all values for r_2 and r_3. The color of the figure indicates the maximum absolute bias obtained for each combination of rotations. The value obtained for the original ChaCha is depicted inside a black circle.

values. Our final statistic is defined as the maximum value in this array. We performed this test considering 7 rounds of ChaCha.

After these tests, we chose our set rotation distances as $r_1 = 19$, $r_2 = 17$, $r_3 = 25$ and $r_4 = 11$, which are the values that minimize the product between both statistics. In particular, for this combination of rotation distances, we obtained 0.01497 for the bias of the differential path and 0.221 for the worst average neutrality. In the next section, we will show that this choice does improve the security of ChaCha against known attacks.

5. Security comparison

5.1. Estimating the Complexity of the PNB Attack

In [Aumasson et al. 2008], the authors reported an attack on 256-bit ChaCha20/6 and ChaCha20/7. For ChaCha20/6, they used the differential $(\Delta^3_{11,0}|\Delta^0_{13,13})$ with $|\varepsilon^\star_d| = 0.026$. The $\mathcal{OD}$ is observed after working 3 rounds backward from a 6-round keystream block. For the threshold $\gamma = 0.6$ they found a set of 147 non-significant key bits, with $|\varepsilon| = 0.00048$. This results in an attack in time 2^{139} and data 2^{30}. For ChaCha20/7, they used the same differential. The $\mathcal{OD}$ is observed after working 4 rounds backward from a 7-round keystream block. For the threshold $\gamma = 0.5$, they found a set of 35 non-significant key bits with $|\varepsilon| = 0.00059$. This results an attack in time 2^{248} and data 2^{27}.

We ran the attacks for ChaCha again, obtaining very similar complexity results. Using the same program, we ran the attack for ChaCha with rotation distances $19, 17, 25, 11$, showing that we get a stronger cipher. In fact, for 7 rounds, we did not find any attack with time $< 2^{256}$, see Table 2 for the results.

Algorithm	$\mathcal{ID}$	$\mathcal{OD}$	ε_d^*	γ	n	ϵ^*	Data	Time
ChaCha20/6	$\Delta^{(0)}_{12,21}$	$\Delta^{(3)}_{2,0}$	-0.1973	0.6	134	-0.0039	$2^{23.9}$	$2^{145.9}$
ChaCha20/7	$\Delta^{(0)}_{12,21}$	$\Delta^{(3)}_{2,0}$	-0.1977	0.4	20	-0.0097	$2^{17.8}$	2^{254}
*ChaCha20/6	$\Delta^{(0)}_{14,17}$	$\Delta^{(3)}_{1,0}$	-0.0059	0.8	111	-0.0019	$2^{25.6}$	$2^{170.6}$
*ChaCha20/7	–	–	–	–	–	–	–	–

Table 2. Best attacks obtained for ChaCha and for its modified version with rotation distances $19, 17, 25, 11$, denoted here by *ChaCha. We could not find any attacks for the modified version of ChaCha with 7 rounds.

5.2. Multi-bit differential

In [Choudhuri and Maitra 2016], the authors provides several different attacks for ChaCha20/4, ChaCha20/5, ChaCha20/6, and ChaCha20/7. For ChaCha20/4, Lemma 1 is used. Considering the first row of Table 3, we note a bias $\varepsilon_d = 0.1984$ and thus $\frac{1}{\varepsilon_d^2/2} < 51$. That is, with 2^6 samples it is enough to distinguish 4-round ChaCha from a uniform random source. However, when changing the rotation distances, the best bias we get is $\varepsilon_d = -0.009179$ and thus $\frac{1}{\varepsilon_d^2/2} < 23738$. That is, with 2^{15} samples it is enough to distinguish 4-round ChaCha with rotation distances $19, 17, 25, 11$ from a uniform random source.

For ChaCha20/5, if we define $\mathcal{ID}$ at $\Delta x^0_{13,13}$ and $\mathcal{OD}$ at $\Delta x^3_{11,0}$, we obtain $\varepsilon_d = -0.0272$. By Lemma 1, we can extend this bias to 4 rounds, and by Lemma 2, we can further extend this bias to 5 rounds with probability $3/4$, or $\varepsilon_L = 1/2$. This gives a total differential-linear 5-th round bias of $\varepsilon = \varepsilon_d \varepsilon_L^2 = -0.0068$ thus $\frac{1}{\varepsilon^2/2} < 43253$. That is, with 2^{16} samples it is enough to distinguish 5-round ChaCha from a uniform random source. However, changing the rotation distances and if we define $\mathcal{ID}$ at $\Delta^0_{14,12}$ and $\mathcal{OD}$ at $\Delta^3_{8,0}$, we obtain $\varepsilon_d = -0.000915$, and from Lemmas 1 and 2, we get a total differential-linear 5-th round bias of $\varepsilon = \varepsilon_d \varepsilon_L^2 = -0.00022875$ thus $\frac{1}{\varepsilon^2/2} < 38221506$. That is, with 2^{26} samples it is enough to distinguish 5-round ChaCha with rotation distances $19, 17, 25, 11$ from a uniform random source.

Algorithm	$\mathcal{ID}$	$\mathcal{OD}$	Bias
ChaCha	$\Delta x^{(0)}_{12,20}$	$\Delta x^{(4)}_{2,0} \oplus \Delta x^{(4)}_{7,7} \oplus \Delta x^{(4)}_{7,19} \oplus \Delta x^{(4)}_{8,12} \oplus \Delta x^{(4)}_{13,0}$	0.1984
ChaCha	$\Delta x^{(0)}_{14,20}$	$\Delta x^{(4)}_{0,0} \oplus \Delta x^{(4)}_{5,7} \oplus \Delta x^{(4)}_{5,19} \oplus \Delta x^{(4)}_{10,12} \oplus \Delta x^{(4)}_{15,0}$	0.1979
ChaCha	$\Delta x^{(0)}_{15,20}$	$\Delta x^{(4)}_{1,0} \oplus \Delta x^{(4)}_{6,7} \oplus \Delta x^{(4)}_{6,19} \oplus \Delta x^{(4)}_{11,12} \oplus \Delta x^{(4)}_{12,0}$	0.1973
ChaCha	$\Delta x^{(0)}_{13,20}$	$\Delta x^{(4)}_{3,0} \oplus \Delta x^{(4)}_{4,7} \oplus \Delta x^{(4)}_{4,19} \oplus \Delta x^{(4)}_{9,12} \oplus \Delta x^{(4)}_{14,0}$	0.1972
*ChaCha	$\Delta x^{(0)}_{14,1}$	$\Delta x^{(4)}_{0,0} \oplus \Delta x^{(4)}_{5,11} \oplus \Delta x^{(4)}_{5,28} \oplus \Delta x^{(4)}_{10,17} \oplus \Delta x^{(4)}_{15,0}$	-0.009179
*ChaCha	$\Delta x^{(0)}_{15,16}$	$\Delta x^{(4)}_{0,0} \oplus \Delta x^{(4)}_{5,11} \oplus \Delta x^{(4)}_{5,28} \oplus \Delta x^{(4)}_{10,17} \oplus \Delta x^{(4)}_{15,0}$	-0.009133
*ChaCha	$\Delta x^{(0)}_{15,1}$	$\Delta x^{(4)}_{1,0} \oplus \Delta x^{(4)}_{6,11} \oplus \Delta x^{(4)}_{6,28} \oplus \Delta x^{(4)}_{11,17} \oplus \Delta x^{(4)}_{12,0}$	-0.009122
*ChaCha	$\Delta x^{(0)}_{14,16}$	$\Delta x^{(4)}_{3,0} \oplus \Delta x^{(4)}_{4,11} \oplus \Delta x^{(4)}_{4,28} \oplus \Delta x^{(4)}_{9,17} \oplus \Delta x^{(4)}_{14,0}$	-0.009099

Table 3. The best multi-bit differentials for ChaCha and for its modified version with rotation distances $19, 17, 25, 11$, denoted here by *ChaCha. Notice that we can reduce the bias significantly.

Extending the linear approximation for 3 rounds comes at a cost. As discussed in

[Choudhuri and Maitra 2016], for 6 rounds, the linear bias after expanding any equation from Lemma 2 is $\varepsilon_L = 1/(2 \cdot 1 + 3 \cdot 4 + 5 \cdot 1 + 3 \cdot 2 + 2 \cdot 1) = 1/2^{26}$. To use this extension, we searched for the input differential which maximizes the differential bias for $\Delta x^{(3)}_{8,0}, \Delta x^{(3)}_{9,0}, \Delta x^{(3)}_{10,0}$ or $\Delta x^{(3)}_{11,0}$, which leads to the differential pair $(\Delta x^{(3)}_{9,0} | \Delta x^{(0)}_{15,12})$ with $\varepsilon_d = 0.000792$. This leads to a 6-round bias of $\varepsilon_L^2 \varepsilon_d \approx \frac{1}{2^{62.3}}$ and a distinguisher with complexity of 2^{125}.

For a key recovery attack against 6 rounds of ChaCha, we must use PNB. The best attack we obtained when considering the proposed rotation distances uses 5 rounds forward and then 1 round backward. For this the $\mathcal{ID}$ is $\Delta^{(0)}_{12,12}$ and the $\mathcal{OD}$ in the third round is $\Delta^{(3)}_{10,0}$, thus, using the third equation of Lemma 2, we can mount an attack. We got 157 PNBs using $\gamma = 0.6$ from which we estimated $\varepsilon = -0.000024$, $\varepsilon^\star = -0.000023$ leading to an attack with data complexity $2^{38.7}$ and time complexity $2^{137.7}$. For 7 rounds of ChaCha with the proposed rotation distances, we did not find any significant attacks. Also, we could not use the equations from Lemma 4 since we could not find any significant bias for a double output differential bias. Table 4 summarizes our findings.

Algorithm	Rounds	Data	Time	Type	Reference
ChaCha	4	2^6	2^6	Distinguisher	[Choudhuri and Maitra 2016]
	5	2^{16}	2^{16}	Distinguisher	[Choudhuri and Maitra 2016]
	6	2^{75}	2^{75}	Distinguisher	[Coutinho and Neto 2020]
	6	2^{56}	$2^{102.2}$	Key recovery	[Coutinho and Neto 2020]
	7	2^{50}	$2^{231.9}$	Key recovery	[Coutinho and Neto 2020]
*ChaCha	4	2^{15}	2^{15}	Distinguisher	This work
	5	2^{26}	2^{26}	Distinguisher	This work
	6	2^{125}	2^{125}	Distinguisher	This work
	6	$2^{38.7}$	$2^{137.7}$	Key recovery	This work
	7	–	–	–	–

Table 4. Attacks obtained considering the techniques presented in Section 3.2. Notice that the complexity of the attacks for ChaCha with rotation distances $19, 17, 25, 11$, denoted here by *ChaCha, are higher, thus, the proposed modification is more secure against these attacks.

6. Conclusion

In this work, we proposed a modification for the stream cipher ChaCha. This was done by considering the best attacks in the literature and trying to minimize the differential biases generated by 3 rounds of the algorithm. This analysis resulted in the optimal rotation distances for ChaCha against differential-linear cryptanalysis, which are $r_1 = 19$, $r_2 = 17$, $r3 = 25$, and $r_4 = 11$. We computed the complexity of the two most successful attacks against ChaCha presented in the literature, showing that the proposed modification leads to attacks with higher complexity for 4, 5, and 6 rounds (see Table 4). For 7 rounds, ChaCha with the proposed rotation distances is no longer vulnerable to differential attacks. For future work, it remains to test other types of attacks known in the literature, in particular, related key attacks and the attack of Beierle et al. [Beierle et al. 2020], published after the submission of this work.

Acknowledgements

This work was supported in part by CNPq - Brazilian National Research Council (Grants 312180/2019-5 PQ-2, BRICS2017-591 LargEWiN, and 465741/2014-2 INCT on Cybersecurity), in part by CAPES - Brazilian Higher Education Personnel Improvement Coordination (Grants PROAP PPGEE/UnB, 23038.007604/2014-69 FORTE, and 88887.144009/2017-00 PROBRAL), in part by FAP-DF - Brazilian Federal District Research Support Foundation (Grant 0193.001366/2016 UIoT, and Grant 0193.001365/2016 SSDDC), in part by the Brazilian Ministry of the Economy (Grant 005/2016 DIPLA, and Grant 083/2016 ENAP), in part by the Institutional Security Office of the Presidency of Brazil (Grant ABIN 002/2017), in part by the Administrative Council for Economic Defense (Grant CADE 08700.000047/2019-14), and in part by the General Attorney of the Union (Grant AGU 697.935/2019).

References

[Aumasson et al. 2008] 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*, pages 470–488. Springer.

[Beierle et al. 2020] Beierle, C., Leander, G., and Todo, Y. (2020). Improved differential-linear attacks with applications to ARX ciphers. In *Annual International Cryptology Conference*, pages 329–358. Springer.

[Bernstein 2005] Bernstein, D. J. (2005). The Poly1305-AES message-authentication code. In *International Workshop on Fast Software Encryption*, pages 32–49. Springer.

[Bernstein 2008a] Bernstein, D. J. (2008a). ChaCha, a variant of Salsa20. In *Workshop Record of SASC*, volume 8, pages 3–5.

[Bernstein 2008b] Bernstein, D. J. (2008b). The Salsa20 family of stream ciphers. In *New stream cipher designs*, pages 84–97. Springer.

[Choudhuri and Maitra 2016] Choudhuri, A. R. and Maitra, S. (2016). Significantly improved multi-bit differentials for reduced round Salsa and Chacha. *IACR Transactions on Symmetric Cryptology*, pages 261–287.

[Coutinho et al. 2020] Coutinho, M., De Sousa, R. T., and Borges, F. (2020). Continuous diffusion analysis. *IEEE Access*.

[Coutinho and Neto 2020] Coutinho, M. and Neto, T. C. S. (2020). New multi-bit differentials to improve attacks against ChaCha. Cryptology ePrint Archive, Report 2020/350. `https://eprint.iacr.org/2020/350`.

[Crowley 2006] Crowley, P. (2006). Truncated differential cryptanalysis of five rounds of Salsa20. *The State of the Art of Stream Ciphers SASC*, 2006:198–202.

[Dey et al. 2019] Dey, S., Roy, T., and Sarkar, S. (2019). Revisiting design principles of Salsa and ChaCha. *Advances in Mathematics of Communications*, 13(4).

[Dey and Sarkar 2017] Dey, S. and Sarkar, S. (2017). Improved analysis for reduced round Salsa and ChaCha. *Discrete Applied Mathematics*, 227:58–69.

[Ding 2019] Ding, L. (2019). Improved related-cipher attack on Salsa20 stream cipher. *IEEE Access*, 7:30197–30202.

[Fischer et al. 2006] Fischer, S., Meier, W., Berbain, C., Biasse, J.-F., and Robshaw, M. J. (2006). Non-randomness in eSTREAM candidates Salsa20 and TSC-4. In *International Conference on Cryptology in India*, pages 2–16. Springer.

[Hernandez-Castro et al. 2008] Hernandez-Castro, J. C., Tapiador, J. M., and Quisquater, J.-J. (2008). On the Salsa20 core function. In *International Workshop on Fast Software Encryption*, pages 462–469. Springer.

[IANIX 2020] IANIX (2020). ChaCha usage & deployment. `https://ianix.com/pub/chacha-deployment.html`. Accessed: 2020-01-13.

[Ishiguro et al. 2011] Ishiguro, T., Kiyomoto, S., and Miyake, Y. (2011). Latin dances revisited: new analytic results of Salsa20 and ChaCha. In *International Conference on Information and Communications Security*, pages 255–266. Springer.

[Langley et al. 2016] Langley, A., Chang, W., Mavrogiannopoulos, N., Strombergson, J., and Josefsson, S. (2016). ChaCha20-Poly1305 cipher suites for transport layer security (TLS). *RFC 7905*, (10).

[Maitra 2016] Maitra, S. (2016). Chosen IV cryptanalysis on reduced round ChaCha and Salsa. *Discrete Applied Mathematics*, 208:88–97.

[Maitra et al. 2015] Maitra, S., Paul, G., and Meier, W. (2015). Salsa20 cryptanalysis: New moves and revisiting old styles. In *the Ninth International Workshop on Coding and Cryptography*.

[Mouha and Preneel 2013] Mouha, N. and Preneel, B. (2013). A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptology ePrint Archive*, 2013:328.

[Shi et al. 2012] Shi, Z., Zhang, B., Feng, D., and Wu, W. (2012). Improved key recovery attacks on reduced-round Salsa20 and ChaCha. In *International Conference on Information Security and Cryptology*, pages 337–351. Springer.

[Torvalds 2016] Torvalds, L. (2016). Linux kernel source tree. `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=818e607b57c94ade9824dad63a96c2ea6b21baf3`.

[Tsunoo et al. 2007] Tsunoo, Y., Saito, T., Kubo, H., Suzaki, T., and Nakashima, H. (2007). Differential cryptanalysis of Salsa20/8. In *Workshop Record of SASC*, volume 28.