Moving a Step of ChaCha in Syncopated Rhythm*

Shichang Wang[1,2], Meicheng Liu[1,2(✉)], Shiqi Hou[1,2], and Dongdai Lin[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, P. R. China
{wangshichang,liumeicheng,houshiqi,ddlin}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, P. R. China

Abstract. The stream cipher ChaCha is one of the most widely used ciphers in the real world, such as in TLS, SSH and so on. In this paper, we study the security of ChaCha via differential cryptanalysis based on probabilistic neutrality bits (PNBs). We introduce the *syncopation* technique for the PNB-based approximation in the backward direction, which significantly amplifies its correlation by utilizing the property of ARX structure. In virtue of this technique, we present a new and efficient method for finding a good set of PNBs. A refined framework of key-recovery attack is then formalized for round-reduced ChaCha. The new techniques allow us to break 7.5 rounds of ChaCha without the last XOR and rotation, as well as to bring faster attacks on 6 rounds and 7 rounds of ChaCha.

Keywords: Stream Ciphers · ChaCha · Differential Cryptanalysis · PNB · Syncopation.

1 Introduction

Symmetric key cryptosystems play an indispensable role in cryptography. Owing to the significant performance advantage of symmetric cryptographic primitives, there are massive deployment of symmetric primitives, including stream ciphers, block ciphers, hash functions, and cryptographic permutations, in virtually all real-world applications and scenarios related to cryptography. Among them, ARX-based design is one important and attractive branch with simplicity and efficiency in both software and hardware implementations, especially in consideration of the persistent focus on lightweight cryptography. ARX is short for modular Addition, Rotation and bit-wise XOR. ARX-based designs not only have very high efficiency, but also provide good security properties. The algebraic degree of ARX ciphers is usually high after a very few rounds since the function of carry bit within one modular addition already reaches almost maximal

* Supported by the National Natural Science Foundation of China (Grant No. 62122085, 12231015 and 61936008) and the Youth Innovation Promotion Association of Chinese Academy of Sciences.

degree. With regard to differential [7] and linear [23] attacks, the probabilities of differentials and absolute correlations of linear approximations decrease very quickly when the numbers of rounds increase.

One of the most important set of ARX ciphers is the family consisting of Salsa20, ChaCha and their variants. In 2005, Bernstein proposed the stream cipher Salsa20 [5] as a candidate for the eSTREAM competition [17], and its 12-round variant was accepted into the final software portfolio. Later, as a variant of Salsa20, ChaCha [4] was proposed by Bernstein in 2008 to provide better diffusion and cryptanalytic resistance without slowing down encryption. The stream cipher ChaCha has a total of 20 rounds. In addition, the ChaCha family also includes reduced-round ciphers—the 8-round version ChaCha8 and the 12-round version ChaCha12. In the following, we mainly concentrate on the version of ChaCha with a 256-bit key due to its wide deployment.

ChaCha, along with the message authentication code Poly1305 [3], is adopted as one of the cipher suites of Transport Layer Security (TLS) [20], which has been supported by Google on both Chrome and Android. Also, the RFC 7634 [25] proposes the use of ChaCha along with Poly1305 for IKEv2 and IPsec. In addition, it has been implemented in many other protocols such as SSH, Noise, QUIC, WireGard, S/MIME 4.0 and so on. Apart from the usage of encryption, ChaCha has also been used as a pseudo-random number generator in the operating system with Linux kernel. Up to now, there are plenty of protocols and software which have implemented the stream cipher ChaCha, and please refer to [18] for details. In a nutshell, ChaCha is one of the most widely-used ciphers in practice.

Related works. Because of the wide range of usage and deployment of ChaCha, it is crucial to systematically and deeply analyze its security. And indeed, cryptanalysts have advanced lots of important and profound works on the security evaluation of round reduced ChaCha.

At FSE 2008, Aumasson et al. [1] proposed a significant improvement on the differential cryptanalysis of both Salsa20 and ChaCha with introducing a new concept called *probabilistic neutral bits* (shortcut PNBs). In most cases, the attacks are launched by a meet-in-the-middle approach, in which for forward direction one applies some input difference to the initial state to observe the output difference after certain rounds, and for backward direction once a set of key bits with less influence on the output difference is identified using the PNB method, attackers can obtain the output difference at the middle from the final state. Since most of the attacks on Salsa and ChaCha to date have been carried out under based on PNBs, this idea will be reviewed detailedly in Section 3.1.

Several further enhancements have been proposed following this line of research. In 2012, Shi et al. [26] achieved some incremental advancements for both Salsa20 and ChaCha by introducing the concept of *column chaining distinguisher* (CCD). Maitra [22] provided the idea of chosen IV cryptanalysis to obtain certain improvements in the key-recovery attacks of both ciphers. In 2017, Choudhuri and Maitra [8] improved the attacks with the idea of extending single-bit differentials to multi-bit differentials using linear approximations derived theoretically,

which is essentially a differential-linear cryptanalysis [19]. Dey and Sarkar [15] improved the attacks by giving an algorithm for constructing a good PNB set, and then in [16] they provided a theoretical justification of the distinguishers of both ciphers. At CRYPTO 2020, Beierle, Leander and Todo [2] provided the first 3.5-round single-bit distinguisher, and improved the attacks for ChaCha7 with time complexity of $2^{230.86}$. Later, Coutinho and Souza Neto [10] presented a few more distinguishers for 3.5 rounds of ChaCha, and provided a further improvement on the differential-linear attacks including the distinguishers and key-recovery attacks for 7 rounds of ChaCha. However, in a recent work [12], Dey et al. showed that the correlation of differential at 3.5-th round in [10] is much smaller than their claim which results to inaccurate complexities of the attacks. The authors of [10] immediately corrected this mistake and reevaluated the complexities in [9]. In [12], Dey et al. also provided a theoretical explanation of the issue that not all keys have available IV's to satisfy the given differential, called *strong keys* at [2], mainly for two special cases, and provided experimental results for the other six cases without theoretical explanations. Subsequently, at EUROCRYPT 2022, Dey et al. [14] further improved the attack on ChaCha7 with time complexity of $2^{221.95}$ by introducing a concept of *exploitable keys* and constructing a list to store combinations of exploitable keys with a favorable IV for each key. At ASIACRYPT 2022, Coutinho et al. [11] presented a differential-linear distinguisher against 7 rounds of ChaCha by improving its linear part, with both time and data complexities of 2^{214}. Very recently, Dey et al. [13] presented a full key recovery attack on ChaCha6 with a complexity of $2^{99.48}$.

Since the first published analysis of ChaCha in 2008 by Aumasson et al. [1], its valid attacks in the literatures have stayed within 7 rounds [26,8,15,2,14]. In [24], Miyashita et al. presented an attack on 7.25 rounds of ChaCha with time complexity of $2^{255.62}$ and success probability of 0.5. However, as declared in [24], this attack is less efficient than a brute force attack.

Contributions. So far, the best results of key-recovery attacks against ChaCha are obtained by the PNB-based differential cryptanalysis, which is composed of two parts: forward truncated differential and backward PNB-based approximation. For the two parts, we have proposed several new techniques and improvements, which are summarized as follows.

Backward PNB-based approximation: In the PNB-based approximation, the essential problem is how to find numerous PNBs with a large correlation of backward approximation. Unfortunately, more PNBs and larger correlation are usually contradictory. To simplify the problem, we first show that the backward approximation $\Delta f = \Delta g$ connected with the forward differential corresponding to two initializations can be seen as applying twice an approximation $f = g$ in a single initialization, and thus its correlation is the square of the correlation of the latter (see also Proposition 1). This fact is simple and was not illuminated in the previous analysis but it is surprising that it significantly simplifies calculations of the backward correlation. By this observation, we can treat the PNB-based approximation similar to

the linear part in differential-linear cryptanalysis. However, the PNB-based approximation is much more complicated. We then introduce the technique called *syncopation* that takes advantage of ARX structures to analyze the propagation characteristic of PNBs and amplify the approximation correlation. To further simplify the analysis, the approximated part f is divided into two parts f_0 and f_1, that is, $f = f_1 \circ f_0$. The PNB-based approximating function g is a restriction of f with PNBs to be a fixed value, and this relationship also applies to g_0 and f_0. The *syncopation* technique is used in the g_0 (to say f_0) part for reducing its dependence on PNBs so that the correlation between g_0 and f_0 is high. We analyze the properties of syncopation in the basic operations of ARX designs, especially including modular subtraction and modular addition. We also present a tool for determining syncopations, which leads to a new and efficient method for finding a good set of PNBs with a large correlation by restrictions on non-PNBs. In virtue of the new observations and techniques, a refined framework of the PNB-based differential attack is formalized as described in Section 4.

Forward differential: By analysis of the equations that control differential propagation, we achieve two main improvements in the differential part. Firstly, we show several useful observations on these differential equations, and propose a concise and effective way to satisfy the given differential. This treatment leads to a save of time by 2^{-2} in the key recovery at the cost of 2^4 times of data, compared with the work of Beierle, Leander and Todo [2]. Compared with Dey et al.'s technique [14], it does not require restrictions on exploitable keys (which shrink the space of all possible PNBs) and saves up to 2^{22} times of data. Secondly, a more comprehensive theoretical explanation is presented for the proportion of *strong keys* (about 30%) reported in [2], which covers all the remaining six (out of eight) cases lacked in the previous theoretical analysis [12].The details are shown in Section 5.

Comparison of results. As applications of our techniques, we present several attacks on the round-reduced versions of ChaCha with a 256-bit key. To be more specific, for ChaCha6, we show a partial key-recovery attack with time complexity of $2^{75.7}$ and data complexity of $2^{73.7}$, where nine more key bits (total 45 key bits) can be recovered with less time complexity than the previous best known partial key recovery [2]. With regard to ChaCha7, several attacks are obtained with different complexities. Specifically, we can launch a key-recovery attack on ChaCha7 with a time complexity of $2^{210.3}$ using a data complexity of $2^{103.3}$. With a data complexity of $2^{68.9}$, our attack on ChaCha7 takes a time complexity of $2^{216.9}$. The data and time complexity of this attack are both less than the previous best known attack with a time complexity of $2^{221.95}$ [14]. Towards a closer analysis of ChaCha8, we present two attacks against ChaCha7.5$^{\oplus}$, which is defined as a round-reduced version without the last bit-wise XOR and rotation compared with ChaCha7.5. Our attack on ChaCha7.5$^{\oplus}$ is launched with the time complexity of $2^{244.9}$ and data complexity of $2^{104.9}$. We also obtain another attack on ChaCha7.5$^{\oplus}$, which requires time complexity of $2^{242.9}$ with an increasing data complexity of $2^{125.8}$.

As far as we know, these attacks are the best known (partial) key-recovery attacks on ChaCha6, ChaCha7 and ChaCha7.5$^{\oplus}$. In particular, our attacks on ChaCha7 and ChaCha7.5$^{\oplus}$ are both several thousands times faster than the previous best known key-recovery attack or a brute force attack. Moreover, this is the first time that a key recovery attack is achieved beyond 7 rounds of ChaCha. As mentioned earlier, since the first 7-round attack was presented by Aumasson et al. [1] in 2008, nearly fifteen years have passed while no public cryptanalytic results superior to a brute force have been proposed more than 7 rounds of ChaCha. In consequence, we take a small but difficult step towards the security analysis of ChaCha.

Our key-recovery attacks on round-reduced ChaCha are summarized in Table 1, with a comparison of the best known attacks that are better than a brute force attack. Last but not least, it is important to emphasize that our attacks do not pose any threat on ChaCha with full 20 rounds in the actual deployment.

Table 1. Key-recovery attacks on ChaCha

Round-reduced ChaCha	Time	Data	Reference
ChaCha6	2^{139}	2^{30}	[1]
	2^{136}	2^{28}	[26]
	$2^{127.5}$	$2^{37.5}$	[8]
	$2^{99.48}$	-	[13]
	$2^{77.4}$†	2^{58}	[2]
	$2^{75.7}$†	$2^{73.7}$	Sect. 6.3
ChaCha7	2^{248}	2^{27}	[1]
	$2^{246.5}$	2^{27}	[26]
	$2^{237.7}$	2^{96}	[8]
	$2^{230.86}$	$2^{48.83}$	[2]
	$2^{221.95}$	$2^{90.20}$‡	[14]
	$2^{216.9}$	$2^{68.9}$	Sect. 6.1
	$2^{210.3}$	$2^{103.3}$	Sect. 6.1
ChaCha7.5$^{\oplus}$	$2^{244.9}$	$2^{104.9}$	Sect. 6.2
	$2^{242.9}$	$2^{125.8}$	Sect. 6.2

† In the analysis of ChaCha6, the partial key-recovery attacks are launched where 36 key bits are restored in [2], and nine more bits (i.e. 45 bits) are recovered in our attack.

‡ By our evaluation, it can be cut down to $2^{75.89}$.

The full version of this paper and all codes of verification experiments can be found at https://github.com/desert-oasis/chacha_syncopation.git. To demonstrate the proposed techniques and verify our attacks, we have implemented experiments on practical key-recovery attacks against 5 rounds and 6 rounds of ChaCha with 64-bit secret key. The details of these experiments can be found in Supplementary Material C.4.

Organization of the paper. In Sect. 2, we briefly recall the specification of ChaCha, and list the notations used throughout this paper. In Sect. 3, we review the framework of differential attacks based on PNB method, and recent advances on cryptanalysis of ChaCha. Then in Sect. 4, we introduce the *syncopation* technique for the PNB-based approximation in the backward direction and refine the PNB-based differential attack taking this technique into account. Next in Sect. 5, the analysis of differential part in the forward direction is provided. In Sect. 6, we present the attacks on ChaCha7, ChaCha7.5$^{\oplus}$ and ChaCha6 with our proposed techniques. Finally, Sect. 7 concludes this paper.

2 Preliminaries

In this section, we first take a look at the stream cipher ChaCha, and then summarize the notations used throughout this paper.

In 2008, the stream cipher ChaCha [4] was introduced by Bernstein, as a variant of Salsa20 aiming at bringing better diffusion for similar performance. To start with, we give a brief description of ChaCha. The cipher operates on 32-bit words, takes as input a 256-bit secret key $k = (k_0, k_1, \cdots, k_7)$, a 96-bit nonce $v = (v_0, v_1, v_2)$ and a 32-bit block counter t_0, and produces a sequence of 512-bit keystream blocks. In the subsequent content, we will refer to the nonce and block counter words together as IV words. The operations consist of bit-wise XOR ($\oplus$), left rotation ($\lll$) and addition modulo 2^{32} ($\boxplus$). The state of ChaCha consists of sixteen 32-bit words, which can be represented as a 4×4 matrix

$$\mathbf{X} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & v_0 & v_1 & v_2 \end{pmatrix}. \tag{1}$$

As for the initial state, it is built by placing the four predefined constants (c_0, c_1, c_2, c_3), secret key $k = (k_0, k_1, \cdots, k_7)$, block counter t_0 and nonce (v_0, v_1, v_2) as in the above matrix, where the four constants for the version with a 256-bit key are $c_0 =$ 0x61707865, $c_1 =$ 0x3320646e, $c_2 =$ 0x79622d32, $c_3 =$ 0x6b206574.

The round function *Round* consists of 4 parallel applications of the function *quarterround*. The function *quarterround* uses 4 modular additions and 4 XOR's and 4 rotations to update four 32-bit state words. Specifically, the function *quarterround* transforms a 4-word vector (a, b, c, d) into (a'', b'', c'', d'') via an intermediate vector (a', b', c', d'):

$$\begin{array}{ll} a' = a \boxplus b; & d' = (d \oplus a') \lll 16; \\ c' = c \boxplus d'; & b' = (b \oplus c') \lll 12; \\ a'' = a' \boxplus b'; & d'' = (d' \oplus a'') \lll 8; \\ c'' = c' \boxplus d''; & b'' = (b' \oplus c'') \lll 7. \end{array}$$

As depicted in Fig. 1, this transformation is invertible and updates each word twice, so the round function of ChaCha is also invertible. Let $X^{(r)}$ denote the

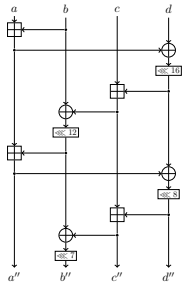


Fig. 1. The function *quarterround* of ChaCha.

state after r rounds, i.e. $X^{(r)} = Round^r(X^{(0)})$. The inverse of round function is denoted as $Round^{-1}$, then $X^{(0)} = Round^{-r}(X^{(r)})$. In the odd rounds, called column rounds, the function *quarterround* is applied to the four columns (x_0, x_4, x_8, x_{12}), (x_1, x_5, x_9, x_{13}), $(x_2, x_6, x_{10}, x_{14})$ and $(x_3, x_7, x_{11}, x_{15})$. In the even rounds, called diagonal rounds, the *quarterround* function is applied to the four diagonals $(x_0, x_5, x_{10}, x_{15})$, $(x_1, x_6, x_{11}, x_{12})$, (x_2, x_7, x_8, x_{13}) and (x_3, x_4, x_9, x_{14}). The keystream block Z of ChaCha$R$ is obtained as the word-wise modular addition of the initial state and the state after R rounds, which means $Z = X^{(0)} \boxplus X^{(R)}$ and $X^{(R)} = Round^R(X^{(0)})$.

Next, we list the notations mainly used throughout this paper in Table 2.

Table 2. Notations.

Notation	Description
x	An n-bit word, i.e., an n-bit vector $x = (x[n-1], x[n-2], \cdots, x[0]) \in \mathbb{F}_2^n$.
$X^{(r)}$	State matrix after applications of r round functions.
x_i or $X[i]$	The i-th word of state matrix X.
$x_i[j_2 : j_1]$	The consecutive $(j_2 - j_1 + 1)$-bit vector of the i-th word of state matrix X, starting from $x_i[j_1]$ ending to $x_i[j_2]$, $j_2 \geq j_1$.
$x \boxplus y$	Addition of x and y modulo 2^n.
$x \boxminus y$	Subtraction of x and y modulo 2^n.
$x \oplus y$	Bit-wise XOR of x and y.
$x \lll l$	Rotation of x by l bits to the left.
Δx	XOR difference of x and x', i.e. $\Delta x = x \oplus x'$.

3 Reviewing Differential Cryptanalysis of ChaCha

In this section, we first review the differential attack proposed by Aumasson et al. [1], which is based on the technique called *probabilistic neutral bits* (shortcut PNBs). Then, we summarize the recent advances on cryptanalysis of ChaCha.

3.1 Differential Attack Based on PNB Method

In general, the differential attacks based on PNB method in [1] use a meet-in-the-middle idea. Usually, one selects a truncated differential with some (single-bit) input/output differences after certain rounds for forward direction. Then one identifies the set of PNBs with less influence over the output difference from backward direction. As for the significant key bits, we just guess and recover them in the key recovery phase of the attacks. The framework of differential attack based on PNB method is illustrated in Fig. 2, where we use the shortcuts ID and OD for input and output difference.

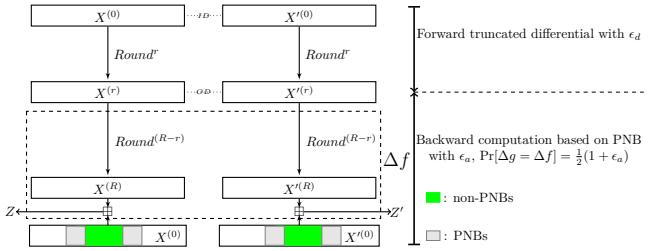


Fig. 2. Framework of PNB-based differential attack.

Assume that we use a truncated differential with a single-bit input difference at $X^{(0)}[i][j]$ of IV and a single-bit output difference at $X^{(r)}[p][q]$ of middle state after r rounds, denoted by $(\Delta X^{(r)}[p][q], \Delta X^{(0)}[i][j])$, which is actually a differential-linear (DL) approximation, and its correlation ε_d is defined by

$$\Pr_{v,t}\left\{\Delta X^{(r)}[p][q] = 0 \mid \Delta X^{(0)} = e_{ij}\right\} = \frac{1}{2}\left(1+\varepsilon_d\right), \tag{2}$$

where e_{ij} is a state whose j-th bit of i-th word is one and other bits are zero. Note that regarding key as a random variable, the median of correlation over all keys is used in the following analysis.

The corresponding output Z is observed for a nonce v, counter t and the given secret key k. Having k, v and t, one can invert the operations to access internal state from backward direction. Given the above forward differential, let

$$\Delta f(k,v,t,Z,Z') \triangleq (Round^{-(R-r)}(Z \boxminus X^{(0)}) \oplus Round^{-(R-r)}(Z' \boxminus X'^{(0)}))[p][q], \tag{3}$$

then $\Delta X^{(r)}[p][q] = \Delta f(k,v,t,Z,Z')$. To obtain a more efficient attack than exhaustive search over all possibilities of secret key, we need to find an approximation Δg of Δf which effectively depends on $m = 256 - n$ key bits. More formally, let k^1 correspond to the subkey of m bits of secret key $k = (k^1, k^0)$ and Δf be correlated to Δg with correlation ε_a,

$$\Pr_{v,t}\left\{\Delta f\left(k,v,t,Z,Z'\right) = \Delta g\left(k^1,v,t,Z,Z'\right)\right\} = \frac{1}{2}\left(1+\varepsilon_a\right) \tag{4}$$

Denote the correlation of Δg by ε, i.e. $\Pr_{v,t}\left\{\Delta g(k^1, v, t, Z, Z') = 0\right\} = \frac{1}{2}(1+\varepsilon)$. Under a reasonable independence assumption, the equality $\varepsilon = \varepsilon_d \cdot \varepsilon_a$ holds.

Probabilistic neutral bits. Generally speaking, PNBs are these non-significant key bits with less influence (of size n), and on the opposite side non-PNBs are these significant key bits with great influence (of size m). To identify the set of PNBs (or non-PNBs), Aumasson et al. [1] gave a formal definition of a suitable measure for the amount of influence which every key bit has on output of Δf.

Definition 1 (Neutrality measure [1]). *The neutrality measure of $k[i]$ with respect to the function $\Delta f(k, W)$ is defined as γ_i, where $W = (v, t, Z, Z')$ and $\Pr\{\Delta f(k, W) = \Delta f(k \oplus e_i, W)\} = \frac{1}{2}(1+\gamma_i)$, i.e. the probability (over all k and W) that complementing the key bit $k[i]$ does not change the output of $\Delta f(k, W)$, where e_i is the unit vector of which i-th bit is one and other bits are all zero.*

In practice, a threshold γ is set, and $PNB = \{i \mid |\gamma_i| > \gamma\}$ and $Non\text{-}PNB = \{i \mid |\gamma_i| \leq \gamma\}$. Assuming that k^1 denotes the significant key bits, i.e. non-PNBs, we simply define $\Delta g(k^1, W)$ as $\Delta f(k, W)$ with setting all PNBs into a fixed value (e.g. all zeros). Then the correlation ϵ_a is measured experimentally by using enough random IVs under many randomly chosen keys.

The Neyman-Pearson decision theory gives the results for estimating the number of samples N required to get the bounds on probabilities of false alarm p_{fa} and non-detection p_{nd}. It can be shown that $N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-(\varepsilon_a \varepsilon_d)^2}}{\varepsilon_a \varepsilon_d}\right)^2$ samples suffices to achieve that $p_{nd} = 1.3 \times 10^{-3}$ and $p_{fa} = 2^{-\alpha}$. With using the median correlation ε^* in the above equation, we have a success probability of at least $\frac{1}{2}(1 - p_{nd}) \approx \frac{1}{2}$ for the attack.

Subsequently, a two-step key-recovery procedure can be launched, where the non-PNBs are first recovered according to the given single-bit differential and then PNBs are found by exhaustive search. The time complexity of this key-recovery attack is calculated as $T = 2^m N + (2^m p_{fa} + (1 - p_{nd}))2^n \approx 2^m N + 2^{256-\alpha} + 2^n$, where $m + n = 256$. For details about this procedure, please refer to [1]. As applications of the PNB-based differential cryptanalysis, Aumasson et al. [1] presented the analysis of Salsa, ChaCha and Rumba, especially including the first attacks on 6 and 7 rounds of ChaCha.

3.2 Recent Advances in Cryptanalysis of ChaCha

In the following, we recall recent advances in cryptanalysis of ChaCha in [2,14].

At CRYPTO 2020, Beierle et al. [2] presented the first 3.5-round differential-linear (DL) approximation for ChaCha, leading to further improvements on attacks on ChaCha6 and ChaCha7. More precisely, the DL distinguisher is divided into two parts, the differential characteristic in E_1 and the middle DL approximation in E_m. For the new distinguisher of ChaCha in [2], the first part E_1 consists of one round and the second part E_m covers 2.5 rounds. The four *quarterround* functions are independent in the first round and we

pay our attention on only one of them in the following analysis. The differential characteristics of E_1 can be presented over one *quarterround* function as $\Delta_{in} = ([], [], [], [6]) \rightarrow \Delta_m = ([2], [5, 9, 17, 29], [10, 22, 30], [10, 30])$, where Δ_m has the minimal Hamming weight of 10. Then with Δ_m as input difference of middle part E_m, they found the following DL approximation with probability $\Pr[(\Delta X_j^{(1)}, \Delta X_{j+4}^{(1)}, \Delta X_{j+8}^{(1)}, \Delta X_{j+12}^{(1)}) = \Delta_m \rightarrow \Delta X_{(j+1) \bmod 4}^{(3.5)}[0]] = \frac{1}{2}(1+\varepsilon_d)$, for $j \in \{0, 1, 2, 3\}$, where the correlation is evaluated experimentally as $\varepsilon_d = 2^{-8.3}$. The experimental distinguisher of E_m is combined with the differential characteristic of E_1, and the resulting distinguisher covers 3.5 rounds with single-bit input difference and output linear mask, which is summarized in Table 3 for the case $j = 1$. To find a right pair in the first round, one can repeatedly choose random IVs with some fixed iterations. As evaluated in [2], the probability that pairs with the input difference Δ_{in} satisfy the given differential in the first round is $p = 2^{-5}$ on average for about 70% of the keys. Therefore, on average we need repeat the procedure of attacks for $p^{-1} = 2^5$ times with varying IVs in the input difference column (ID column) on which the input difference is imposed.

To avoid the cost of iterations for getting a right pair, Dey et al. [14] proposed another alternative way to find the right pairs with the help of a pre-computed list. Actually, one should pre-compute all the needed right pairs offline and keep them in a list. To achieve this goal, they introduced the concept of *exploitable key*, which was restated as the following in the case of ChaCha.

Definition 2 (Exploitable key [14]). *Let $k_{ID} \in \mathbb{F}_2^{64}$ and a subspace $\mathbb{S}_{K_{nmem}} = \{(a_{63}, a_{62}, \cdots, a_0) \mid a_i = 0$ for $i \notin K_{nmem}\} \subseteq \mathbb{F}_2^{64}$, if there exists at least one IV $v \in \mathbb{F}_2^{32}$ such that for any key $k'_{ID} \in k_{ID} \oplus \mathbb{S}_{K_{nmem}}$, $((k'_{ID}, v), (k'_{ID}, v'))$ forms a right pair, k_{ID} is an exploitable key with respect to the subspace $\mathbb{S}_{K_{nmem}}$, and the corresponding v is a favorable IV for k_{ID}.*

They constructed a subset K_{nmem} with dimension 18, and evaluated that there are approximately 62% exploitable keys among all the keys. Then one should construct a pre-computed list to store about $0.62 \times 2^{64-18} = 2^{45.31}$ possible exploitable keys along with their favorable IVs. They used the same forward differential of [2] for the attack on ChaCha7. With their systematic three-step strategy, they found a set of 79 PNBs with the backward correlation $\varepsilon_a = 0.00057$. When $\alpha = 38.8$, this gives $N = 2^{44.89}$, and the time complexity is $2^{221.95}$. Since there are $2^{45.31}$ exploitable keys in the ID column, they estimated the overall data complexity as $2^{45.31} \times N = 2^{90.2}$.[3]

4 The PNB-based Attack with Syncopation

In this paper, we describe a refined framework of differential attack based on the PNB method for ARX ciphers. In this section, we focus on the PNB-based

[3] Noting that there are at most 2^{32} IVs in the ID column, the number N_{ID} of different IV pairs stored in the pre-computed list is less than 2^{31}, and therefore the overall data complexity can be cut down to $2^{31} \times N = 2^{75.89}$.

Table 3. The 3.5-round DL approximation in [2].

ID	OD	$\lvert\varepsilon_d\rvert$
$\Delta X_{13}^{(0)}[6]$ satisfying $\Delta_{in} \to \Delta_m$	$\Delta X_2^{(3.5)}[0]$	$2^{-8.3}$

approximation in the backward direction. The analysis of differential part in the forward direction will be discussed in the next section.

In the following, we will discuss the calculation of approximation correlation and searching for PNBs using properties of the ARX structure in the PNB-based approximation, along with the attack framework.

First, we show a simple but useful observation about the correlation of backward approximation, which significantly cuts down the workload of estimation of backward correlation. More precisely, we observe that the backward approximation of the differential described in (4) is actually the sum of the same approximation with two different inputs in initialization, and thus its correlation is the square of the correlation of the latter, which is shown in Proposition 1. This fact is simple and was not illuminated in the previous analysis, but it is surprising that it is powerful in the calculation of backward correlation.

After that, a technique called *syncopation* is introduced for the PNB-based approximation of ARX ciphers, to amplify the correlation of the approximation. For simplifying the analysis, the approximated part of an ARX cipher, denoted by f, is divided into two parts f_0 and f_1, that is, $f = f_1 \circ f_0$. The PNB-based approximating function g is a function of f with assigning PNBs to be a fixed value, and the same relationship holds between g_0 and f_0. The *syncopation* technique is applied to the part g_0 (to say f_0) for reducing its dependence on PNBs so that the correlation between g_0 and f_0 is high. The syncopations split the input of f_0 into different segments such that each segment of output of f_0 only depends on the same segment of the input under some restrictions. A syncopation consists of a few non-PNBs following by PNBs, and a split segment consists of several non-PNBs. In the other words, the output of f_0 excluding the syncopations does not depend on PNBs, and therefore the approximation correlation between g_0 and f_0 is high. In this section, we analyze the properties of syncopation in modular subtraction and modular addition, and present a tool for determining syncopations, which leads to a new and efficient method for finding a good set of PNBs with a large correlation.

The attack framework is depicted as in Fig. 3.

In the backward computations with PNBs, we observe that computing the backward correlation of output difference at middle states can be simplified into the computation of the correlation of the middle state in one initialization. This treatment is always used in the linear part when one constructs a differential-linear approximation. More precisely, let

$$f(k, v, t, Z) \triangleq Round^{-(R-r)}(Z \boxminus X^{(0)})[p][q], \tag{5}$$

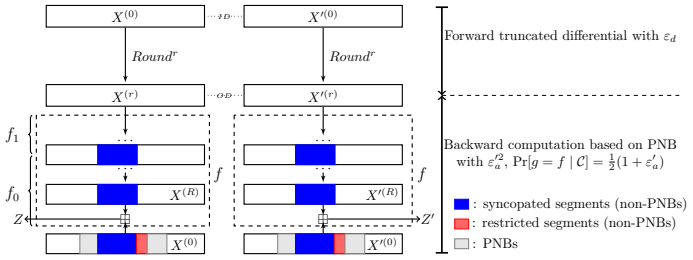


Fig. 3. Refined framework of PNB-based differential attack.

and then by (3) we have $\Delta f(k, v, t, Z, Z') = f(k, v, t, Z) \oplus f(k, v', t', Z')$. Let g an approximation of f which effectively depends on $m = 256 - n$ key bits, more formally, let k^1 correspond to the m-bit subkey of secret key $k = (k^1, k^0)$, in general, g is constructed by assigning a fixed value $\hat{k}^0$ to k^0, i.e.,

$$g(k^1, v, t, Z) \triangleq f(k, v, t, Z) \mid_{k^0 = \hat{k}^0} . \tag{6}$$

Then we can use g to construct an approximation for the function Δf, i.e. $g(k^1, v, t, Z) \oplus g(k^1, v', t', Z')$, which is always the same to the one constructed directly by Δg. Under the assumption of independence, the square relationship is established between the correlation of this treatment and the traditional one, which is summarized in the following.

Proposition 1. *Let f and g be the functions as defined in* (5) *and* (6)*, where Round is the round function, $X^{(0)}$ the state as defined in* (1) *and k^1 corresponds to the m-bit subkey of secret key k. Let ϵ_a be the correlation of f and g,*

$$\Pr_{v,t}\{f(k, v, t, Z) = g(k^1, v, t, Z)\} = \frac{1}{2}(1 + \epsilon_a), \tag{7}$$

and ε_a the correlation of backward approximation in the differential using g,

$$\Pr_{v,t}\{f(k, v, t, Z) \oplus f(k, v', t', Z') = g(k^1, v, t, Z) \oplus g(k^1, v', t', Z')\} = \frac{1}{2}(1 + \varepsilon_a).$$

If Z and Z' are statistically independent, then it holds that $\varepsilon_a = \epsilon_a^2$.

Proof. If Z and Z' are statistically independent, then $Z \boxminus X^{(0)}$ and $Z' \boxminus X'^{(0)}$ are statistically independent. Therefore, by the definitions of f and g, $f(k, v, t, Z) \oplus g(k^1, v, t, Z)$ and $f(k, v', t', Z') \oplus g(k^1, v', t', Z')$ are statistically independent, and both equal to zero with probability of $\frac{1}{2}(1 + \epsilon_a)$. By Piling-up Lemma, $f(k, v, t, Z) \oplus g(k^1, v, t, Z)) \oplus (f(k, v', t', Z') \oplus g(k^1, v', t', Z') = 0$ holds with probability of $\frac{1}{2}(1 + \epsilon_a^2)$. Thus, we have $\varepsilon_a = \epsilon_a^2$. □

By Proposition 1, we can detect the correlation ϵ_a instead of ε_a in practice. It takes about the square root of the amount of computations to detect the correlation ϵ_a in experiments compared with detecting ε_a by using the conventional method. In the following, we usually refer to ϵ_a as the correlation of backward single approximation $f = g$ in one initialization, and correspondingly, to ε_a as the correlation of backward double approximation $\Delta f = \Delta g$ in the differential.

4.1 Syncopation Technique

In the PNB-based approximation, the essential problem is how to find numerous PNBs with a large correlation. Unfortunately, more PNBs and larger correlation are contradictory in most cases. In the literatures, as far as we know, the methods for handling this problem are limited to the naive threshold rule [1] and greedy-like method recently shown in [15,14], though the seminal attack was proposed fifteen years ago. Both methods treat the cipher as a black box, and yet the ARX structure of the cipher has been unexploited. In the existing similar work, this structure has been utilized by the partitioning technique [6,21,2] in linear cryptanalysis and differential-linear cryptanalysis. The idea of partitioning technique is to divide the input or output space into different subsets each of which corresponds to a good differential or linear characteristic. The PNB-based approximation is much more complicated than the cases of differential and linear characteristic, and the technique is not applicable to the PNB-based attack. Even in linear and differential-linear cryptanalysis, some ARX designs might be too complicated to use the partitioning technique, as stated by Beierle et al. in [2], *"unfortunately, 7-round ChaCha is too complicated to apply our (partitioning) technique for the linear part"*. However, we find that some similar properties of the ARX structure utilized by the partitioning technique can be used in the PNB-based approximation though with a different way.

Inspired by the partitioning technique, we introduce the *syncopation* technique for the PNB-based attack, which exploits the properties of ARX structure. As mentioned above, in linear or differential-linear attacks, to increase the correlation of linear approximation, one can use the partitioning technique to choose corresponding linear masks for different subsets of ciphertexts. In the PNB-based attack, a much more complicated function is used to approximate the targeted function. For simplifying the analysis, we divide the approximating function g into two parts g_0 and g_1, that is, $g = g_1 \circ g_0$. The *syncopation* technique is applied to the first part g_0 for reducing its dependence on PNBs so that the correlation of the approximation g_0 is high with the corresponding part f_0.

First the notations and definitions are provided for description of *syncopation* technique. Let $Y = H(X)$ be an ARX function. For $X = (X^o, X^{\#})$ and $Y = (Y^o, Y^{\#})$, if Y^o only depends on X^o under the condition that a system $Q^{\mathcal{R}}(X^o)$ of equations on X^o is satisfied, then $X^{\#}$ and $Y^{\#}$ are called the *syncopation bits*, X^o and Y^o the *syncopated bits*, and the bits of X^o appearing in $Q^{\mathcal{R}}$ the *restricted bits*, denoted by $X^{\mathcal{R}}$. The consecutive bits in each word operated by the ARX function is called a *segment*. For a state S, each segment of its syncopation bits $S^{\#}$ that are consecutive in the state S is called a *syncopation*, and each segment of its syncopated bits S^o that are consecutive in the state S is called a *syncopated segment*. The pattern of syncopations and syncopated segments is called the *syncopation property*. A simple example for syncopation property is given in Fig. 4, and the terms and notations are illustrated in Fig. 4b.

Let the secret key k consist of two parts, the PNBs k^0 and non-PNBs k^1, as defined previously. For the initial state, we set the PNBs to syncopation bits and the other bits to syncopated bits. Let u be a positive integer. We denote by $k^{\mathcal{R}}$

the set of the least u significant bits of a non-PNB segment that are adjacent to PNBs in a word. We restrict conditions on non-PNBs $k^{\mathcal{R}}$ so that the syncopated segment of the output of a PNB-based approximating function does not depend on PNBs, and therefore the approximation correlation will be amplified.

To analyze the propagation of syncopation property for ARX ciphers, like differential and linear characteristic, we provide the basic properties of modular addition and subtraction which are the nonlinear operations in ARX designs.

Property of Modular Addition. Let us consider the operation of modular addition $x = s \boxplus k$ where k is a word of the secret key and s is a public word, as depicted in Fig. 4. Assuming that $k[t+w-1:t]$ is a non-PNB segment and known in backward computation, we wonder under what conditions $x[t+w-1:t]$ is independent of the PNBs and can be determinedly computed from s and non-PNBs of k. To this end, Lemma 1 is derived.

Lemma 1. *Let $x = s \boxplus k$ and $x, k, s \in \mathbb{F}_2^n$. If $k[t-1:t-u] \oplus s[t-1:t-u] \neq \vec{1}$ for $1 \leq u \leq t$, then we have*

$$x[t+w-1:t] = s[t+w-1:t] + k[t+w-1:t] + Carry[t] - Carry[t+w] \cdot 2^w,$$

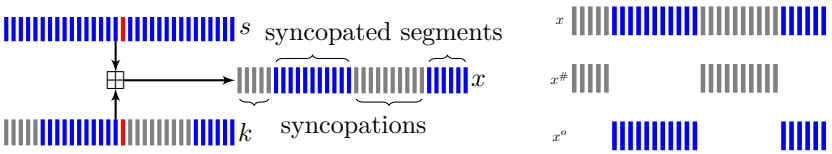
$$Carry[t+w] = \begin{cases} 0 & if \quad s[t+w-1:t] + k[t+w-1:t] + Carry[t] < 2^w, \\ 1 & if \quad s[t+w-1:t] + k[t+w-1:t] + Carry[t] \geq 2^w, \end{cases}$$

$Carry[t] = s[t-\iota]$ *if* $k[t-1:t-\iota+1] \oplus s[t-1:t-\iota+1] = \vec{1}$ *and* $k[t-\iota] \oplus s[t-\iota] = 0$

where the width of segment $w \geq 0$, $1 \leq \iota \leq u$ and $Carry[i]$ denotes the i-th bit of carry vector, i.e. $s \boxplus k = s \oplus k \oplus Carry$.

For completeness, the proof of Lemma 1 is provided in Supplementary Material A.1. Note that the parameter u in Lemma 1 is the number of bits of restricted segments, and ι is an auxiliary parameter to identify the minimum of index such that the premise of lemma holds. For $w = 0$, only $Carry[t]$ is computed in the above lemma. Actually, Lemma 1 generalizes the property of modular addition utilized by the partitioning technique in [21] which can be seen as a special case of Lemma 1 when $w = 1$ and $u = 2$, see also Supplementary Material A.1. In Fig. 4a, according to Lemma 1, the syncopated segment $x[26:16]$ is independent of the syncopation bits, specially including $k[14:6]$, under the condition $k[15] = s[15]$, where $t = 16$, $w = 11$ and $u = 1$.

Property of Modular Subtraction. Let us consider the operation $F_k : \mathbb{F}_2^n \to \mathbb{F}_2^n, x \mapsto s = x \boxplus k$ where k is a word of the secret key and s is a public word. For instance, F_k serves as the last operation of branches b and c of ChaCha. Assuming that $k[t+w-1:t]$ is a non-PNB segment and known, we are interested in the case that $x[t+w-1:t]$ can be determinedly computed from s and non-PNBs of k. Similarly, the property of modular subtraction is derived as Lemma 2.



(a) Propagation for modular addition

(b) x^o: syncpated bits obtained by removing $x^\#$ from x

Fig. 4. An example for syncopation property, where the blue lines represent for syncopated bits, red lines for restricted bits and gray lines for syncopation bits.

Lemma 2. *Let $s = x \boxplus k$ and $x, k, s \in \mathbb{F}_2^n$. If $k[t-1:t-u] \oplus s[t-1:t-u] \neq \vec{0}$ for $1 \leq u \leq t$, then we have*

$$x[t+w-1:t] = Carry[t+w] \cdot 2^w + s[t+w-1:t] - k[t+w-1:t] - Carry[t],$$

$$Carry[t+w] = \begin{cases} 0 & if \quad s[t+w-1:t] > k[t+w-1:t], \\ Carry[t] & if \quad s[t+w-1:t] = k[t+w-1:t], \\ 1 & if \quad s[t+w-1:t] < k[t+w-1:t], \end{cases}$$

$Carry[t] = s[t-\iota] \oplus 1$ *if* $k[t-1:t-\iota+1] \oplus s[t-1:t-\iota+1] = \vec{0}$ *and* $k[t-\iota] \oplus s[t-\iota] = 1$

where the width of segment $w \geq 0$, $1 \leq \iota \leq u$, and $Carry[i]$ denotes the i-th bit of carry vector, i.e. $x \boxplus k = x \oplus k \oplus Carry$.

The proof of Lemma 2 is provided in Supplementary Material A.2 for completeness. Lemma 2 generalizes the property of modular subtraction utilized by the partitioning technique in linear or differential-linear attacks. More exactly, Lemma 2 in [2] falls into the case of Lemma 2 when $w = 1$ and $u = 2$, and please refer to Supplementary Material A.2 for details.

According to Lemma 1 and 2, given the segments $k[t+w-1:t-u]$ and $s[t+w-1:t-u]$, the segment $x[t+w-1:t]$ is known under the condition that the equation $Q^{\mathcal{R}}(k[t-1:t-u], s[t-1:t-u])$ is satisfied, where $Q^{\mathcal{R}}(k[t-1:t-u], s[t-1:t-u])$ denotes $k[t-1:t-u] \oplus s[t-1:t-u] \neq \vec{1}$ and $k[t-1:t-u] \oplus s[t-1:t-u] \neq \vec{0}$ for modular addition and subtraction respectively. Based on the properties of modular addition and subtraction, we summarize the propagation of syncopation property for modular addition and subtraction in the following.

Propagation for Modular Addition and Subtraction. For the operation of modular addition or subtraction on k and s, if all the bits of s are syncopated bits, then, compared with k, the syncopation property of the output is unchanged except that it brings the conditions on restricted bits and converts the restricted bits to syncopation bits. As depicted in Fig. 4a, the sum of top augend and bottom addend has a similar syncopation property with the bottom addend, where all the bits of top augend are syncopated bits. Here a condition

on restricted bits $k[15] = s[15]$ is introduced by modular addition and $x[15]$ is converted to a syncopation bit. This property is useful in our attacks.

Usually, a syncopated segment is obtained by imposing conditions $Q^{\mathcal{R}}$ on restricted bits, which is called as profitable syncopated segment. Hereinafter, assuming that there is a set of profitable syncopated segments of internal state with carries, denoted by $\mathcal{X} = \{(x[t+w-1:t], Carry[t+w])\}$, and let $\mathcal{C}$ be the set of conditions related to $\mathcal{X}$, that is, $\mathcal{C} = \{Q^{\mathcal{R}}(k[t-1:t-u], s[t-1:t-u])\}$.

Propagation for Rotation and XOR. For the rotation operation, the propagation of syncopation property is equivalent in terms of rotation, that is, the bit-wise pattern of syncopation property after rotation is equivalent to rotation of the one before rotation. The propagation of syncopation property on the XOR operation has the following property: the set of the syncopations is the union of input syncopations and the set of the syncopated segments is the intersection of input syncopated segments.

By analysis of the propagation of syncopation property on basic operations, we can get a general property for ARX structures in PNB-based approximation.

Proposition 2. *Let $f(k, s)$ be a function consisting of ARX operations, where the secret key k consists of two parts, the PNBs k^0 and non-PNBs k^1. Let g be a restriction of f with k^0 to be a fixed value. Let the PNBs k^0 be syncopation bits, the other bits of (k, s) be syncopated bits, and $k^{\mathcal{R}}$ be restricted bits. If $k^{\mathcal{R}}$ satisfies the syncopation property propagating from (k, s) to $f(k, s)$, that is, the restricted equations introduced by modular operations are satisfied, then any syncopated segment in the output of f is independent with k^0 and thus equals the corresponding segment in the output of g.*

Instead of finding a linear mask in the partitioning technique, the goal of syncopation technique is to find a PNB set consisting of as many as possible PNBs (to say syncopation bits), as well as maximize the overall length of syncopated segments of internal state in the ARX function for a fixed size of PNB set.

The first question is how to determine the syncopated segments in the PNB-based attack. Next, we present an algorithm to determine the profitable syncopated segments of internal state which are obtained by imposing conditions on the restricted segments.

A Tool for determining syncopated segments and restricted segments. First we find all the possible positions of secret key on which the syncopation technique can be applied, including the syncopated segments and restricted segments of secret key. Then according to the specific structure of target cipher, the profitable syncopated segments of internal state and corresponding conditions are determined by the syncopated segments and restricted segments of secret key. Algorithm 1 summarizes the procedure, and note that the cases of $u = 1$ or $u = 2$ are usually used in our attacks. Note that, for a state S, $S^{\mathcal{R}}$ is a part of S^o by their definitions, but for convenience by k^o hereinafter we mean the syncopated bits of the secret key k that exclude the restricted bits $k^{\mathcal{R}}$.

Algorithm 1 Determining syncopated segments given a set of PNBs

Input: A set of PNBs.

Output: The syncopated segments with conditions on u-bit restricted segments.

1: **for** $0 \leq i \leq 7$ **do**
2: **for** $0 \leq j < 31 - u$ **do**
3: **if** $k_i[j]$ is a PNB and $k_i[j+u : j+1]$ are all non-PNBs **then**
4: $k^{\mathcal{R}} \leftarrow k^{\mathcal{R}} \cup \{k_i[j+u : j+1]\}$
5: **end if**
6: **end for**
7: $k^o \leftarrow k^o \cup k_i^o$, where $k_i^o = \{k_i[j_2 : j_1] \mid k_i[j_2 : j_1]$ are all non-PNBs with $k_i[j_1 - 1]$ (if $j_1 - 1 \geq 0$) and $k_i[j_2 + 1]$ (if $j_2 + 1 \leq 31$) being both PNBs.$\}$
8: **end for**
9: Update k^o by excluding $k^{\mathcal{R}}$ from k^o.
10: According to k^o and $k^{\mathcal{R}}$, find the profitable syncopated segments $\mathcal{X}$ and conditions $\mathcal{C}$ on restricted segments with analyzing the specific structure of cipher. ▷ Refer to Sections 6.1, 6.2 and 6.3 for details.
11: Return $\mathcal{X}$ and $\mathcal{C}$.

Example of Algorithm 1. As shown in Fig. 4a, the bits $\{k[31:27], k[14:6]\}$ are PNBs and the bits $\{k[26:15], k[5:0]\}$ are non-PNBs. After running Step 2–9 of Algorithm 1 with $u = 1$, we get the syncopated segments $k^o = \{k[26:16], k[5:0]\}$ and restricted segments $k^{\mathcal{R}} = \{k[15]\}$. The structure in Fig. 4a illustrates Step 10 of Algorithm 1 for $x = s \boxplus k$ where s is public and known. According to the propagation for modular addition, the set of profitable syncopated segments of internal state is $\mathcal{X} = \{(x[26:16], Carry[27])\}$ and its condition is $\mathcal{C} = \{k[15] = s[15]\}$.

4.2 Refined PNB-based Attack with Syncopation

First, we present a new and efficient method to find good set of PNBs with greedy algorithm. Then, a refined procedure of key-recovery attack is summarized, along with the analysis of complexities.

New efficient method to construct PNB set. In our method, we experimentally estimate the backward correlation of $f = g$ as described in Proposition 1, i.e. the correlation ϵ_a in (7) instead of ε_a in (4). This trick allows us to save lots of computations as explained before and makes our method very efficient. With the tool of determining syncopated segments, we take the syncopation technique into account, which results in a new criteria of identifying PNBs. The resulting key bits by new criteria are called the conditional PNBs (CPNBs for short). Specifically, we first generalize the definition of neutrality measure to conditional neutrality measure, and present a greedy algorithm to find good set of CPNBs.

Definition 3 (Conditional neutrality measure). *The conditional neutrality measure of the key bit $k[i]$ with respect to the function $f(k, v, t, Z)$ and condition $\mathcal{C}_i(k, v, t, Z)$ is defined as γ_i', and* $\Pr\{f(k, v, t, Z) = f(k \oplus e_i, v, t, Z) \mid \mathcal{C}_i(k, v, t,$

Algorithm 2 A greedy algorithm for searching a conditional PNB set

Input: The size n of CPNB set and $CPNB^{shortlist}$.
Output: The CPNB set, its correlation, syncopated segments and conditions.

Initialize a set $CPNB_0 = \emptyset$.
for $i \in \{0, 1, \cdots, n-1\}$ **do**
  for $j \in CPNB^{shortlist}$ *and* $j \notin CPNB_i$ **do**
    $CPNB_{temp} \leftarrow CPNB_i \cup \{j\}$. ▷ The bit positions of k^0 in the key k.
    Update $\mathcal{X}_{i,j}$ and $\mathcal{C}_{i,j}$ by performing Algorithm 1 with $CPNB_{temp}$ as input.
    Under the conditions $\mathcal{C}_{i,j}$, estimate backward correlation $\epsilon_a'^{(i,j)}$ in (7) with assigning k^0 by setting key bits of $CPNB_i$ to zero and flipping key bit j.
  end for
  Choose index j_i with maximal correlation, and $CPNB_{i+1} \leftarrow CPNB_i \cup \{j_i\}$.
end for
Determine $\mathcal{X}_n$ and $\mathcal{C}_n$ of $CPNB_n$ by Algorithm 1.
Under the conditions $\mathcal{C}_n$, estimate backward correlation ϵ_a' in (7) with assigning k^0 by setting key bits of $CPNB_n$ to zero.
Return $CPNB_n$, ϵ_a', $\mathcal{X}_n$ and $\mathcal{C}_n$.

$Z)\} = \frac{1}{2}(1 + \gamma_i')$ *i.e. the probability (over all* k *and* (v, t)*) that complementing the key bit* $k[i]$ *does not change the output of* $f(k, v, t, Z) \mid_{\mathcal{C}_i(k,v,t,Z)}$*. Here* e_i *is the unit vector where the* i*-th bit is one and other bits are all zeros.*

In practice, the condition $\mathcal{C}_i(k, v, t, Z)$ for CPNB $k[i]$ is obtained by Algorithm 1 with treating only $k[i]$ as a PNB and other key bits as non-PNBs. As in the traditional way, one could construct a set of CPNBs by setting a threshold γ for conditional neutrality measure, i.e. $CPNB = \{i \mid |\gamma_i'| \geq \gamma\}$. However, there may raise the incompatibility among CPNBs. Suppose that the key bits $k[i]$ and $k[j]$ are separately derived as CPNBs, but they should not be CPNBs at the same time if the condition $\mathcal{C}_i(k, v, t, Z)$ for $k[i]$ being a CPNB depends on $k[j]$. Therefore, we treat all the CPNBs as a whole when constructing a set of CPNBs.

In general, the procedure of finding a good set of CPNBs is divided into two steps. First, in the preprocessing step, those key bits are excluded whose conditional neutrality measure is less than a threshold γ_{cNeutr}, which will not become good CPNBs. As a result, we get a preliminary shortlisting of CPNBs, denoted by $CPNB^{shortlist}$ whose elements are the key bits with conditional neutrality measure greater than γ_{cNeutr}. Note that the value of γ_{cNeutr} is set to quite lower than the ones in the conventional method, e.g. $\gamma_{cNeutr} = 2^{-5.0}$. In the second step, a greedy algorithm is launched to find a good set of CPNBs from $CPNB^{shortlist}$, which is described in detail below.

Greedy algorithm with new criteria. Once the size of CPNB set is specified, we construct the set of CPNBs by selecting PNB one by one. In the i-th iteration, the index of key bit with the maximal conditional correlation in backward direction, i.e. minimizing the time complexity, is selected and added into the CPNB set. To be accurate, we declare a temporary CPNB set $CPNB_{temp}$ to include a new key bit j, and update the set of profitable syncopated segments $\mathcal{X}_{i,j}$ and the

conditions $\mathcal{C}_{i,j}$ by performing Algorithm 1 with $CPNB_{temp}$ as input. Then under the conditions $\mathcal{C}_{i,j}$, the correlation $\epsilon_a'^{(i,j)}$ in (7) is estimated with assigning k^0 by flipping the new bit j and setting the other bits in $CPNB_{temp}$ to zeros. This iteration is repeated until enough CPNBs are identified. The detailed procedure is presented in Algorithm 2, where ϵ_a' is the conditional backward correlation in a single initialization, that is, $\Pr[f = g \mid \mathcal{C}_n] = \frac{1}{2}(1 + \epsilon_a')$.

When there are large numbers of equations in conditions $\mathcal{C}$ of syncopated segments $\mathcal{X}$, it takes a lot of time overhead to satisfy them, and we will not be able to explore further the impact of conditions on the backward approximation. However, according to the propagation of modular addition and subtraction (Lemma 1 and 2), we know that under the conditions $\mathcal{C}$ of $\mathcal{X}$, the values of syncopated segments $\mathcal{X}$ are independent on PNBs and can be determinedly computed. In practice, we then estimate theoretically the backward correlation at Step 6 and 11 of Algorithm 2 on the premise that the values of $\mathcal{X} = \{(x[t+w-1 : t], Carry[t+w])\}$ are known, that is, estimating the correlation between g_1 and f_1 with fixing the values of $\mathcal{X}$ in g_0 to the ones in f_0. Moreover, our experiments show that the theoretically estimated correlation is very close to the actual one, and we can always use the theoretical value instead of the actual one. It is worth noting that the use of theoretical evaluation of conditional backward correlation makes our method to find good set of CPNBs very efficient.

New key-recovery attack with syncopation technique. In a chosen-IV attack, an attacker can get the keystream blocks by xoring the plaintexts and ciphertexts. Therefore, the accessible in-and-out data consists of public values (e.g., constants and IV) and keystream block Z, which are collectively denoted by M for simplicity. Assume that a set of n CPNBs is generated by our new method, along with the set of syncopated segments $\mathcal{X}$ and corresponding conditions $\mathcal{C}$. Once obtaining a pair of accessible in-and-out data (M, M'), according to its values of the restricted bits $S^{\mathcal{R}}$, the attacker classifies it into the corresponding subset. In the online phase, since the non-PNBs of secret key are guessed, the values of key bits in $k^{\mathcal{R}}$ are known due to the bits of $k^{\mathcal{R}}$ are syncopated bits (that is, non-PNBs). The pairs are then filtered under the conditions $\mathcal{C}$. Suppose that N pairs are given, about $N^* = qN$ pairs will be remaining for subsequent statistic testing, where $q = (\prod(1 - \frac{1}{2^u})^{\theta_u})^2$ according to the premises of Lemma 1 and 2 and θ_u is the number of syncopated segments with an u-bit restricted segment. The detailed procedure of key-recovery attack is summarized as Algorithm 3.

Analysis of complexities. The total time complexity of Algorithm 3 is $T = T_0 + T_1$, where the complexity T_0 of Step 1 is N, and the complexity T_1 of Step 2–8 can be estimated as done in [1]. Therefore,

$$T = N + 2^m N^* + 2^{\kappa-\alpha} + 2^n,$$

where $N = \frac{1}{q}N^*$, $N^* \approx (\frac{\sqrt{\alpha \log 4}+3\sqrt{1-(\varepsilon_a\varepsilon_d)^2}}{\varepsilon_a\varepsilon_d})^2$, $q = (\prod(1 - \frac{1}{2^u})^{\theta_u})^2$, θ_u is the number of syncopated segments with an u-bit restricted segment, ε_d is the

Algorithm 3 Recovery of the full secret key

Input: N pairs $\{(M, M')\}$ of accessible data with IV satisfying ID, set of n CPNBs denoted by k^0 and corresponding conditions $\mathcal{C}$.

Output: The full secret key.

1: Classify N pairs into subsets according to the values of $S^{\mathcal{R}}$. ▷ Pre-processing data where no key bits are involved.
2: **for all** $\hat{k}^1 \in \{0,1\}^m$ **do** ▷ Identify the key $k = (k^1, k^0)$, $|k^1| = m$ and $|k^0| = n$.
3: Set $\hat{k} = (\hat{k}^1, 0^n)$, according to the values of $k^{\mathcal{R}}$, choose the subsets of $\{(M, M')\}$ whose values of $S^{\mathcal{R}}$ satisfying equations in conditions $\mathcal{C}$, with about $N^* = qN$ pairs.
4: Compute the correlation at OD with the filtered N^* pairs.
5: **if** the optimal decision rule legitimates $\hat{k}^1$ as a (possibly) correct one **then**
6: Perform an additional exhaustive search over the remaining n-bit subkey k^0. Once that the right key is found, output it.
7: **end if**
8: **end for**

forward correlation, $\varepsilon_a = \epsilon_a'^2$ is the conditional backward correlation utilizing the syncopation technique, and κ is the size of secret key and $\kappa = 256$ for ChaCha. The data complexity is of N. The memory complexity is mainly consumed in Step 1 of at most N, and if we only keep these subsets used later, it takes memory of $2^{|k^{\mathcal{R}}|}N^*$. For concrete attacks, we refer to Sect. 6 for details.

5 Theoretical Analysis of Differential Equations

In this section, we describe our improvements on forward differential, which are obtained by an in-depth analysis of the equations controlling differential propagation in ARX structure. More precisely, a concise and effective way is found to satisfy the given differential, resulting in a reduced complexity in key-recovery attacks. Besides, a more comprehensive theoretical explanation of *strong keys* is presented, which covers all the remaining six (out of eight) cases lacked in previous analysis.

According to the experiments of [2], for about 70% of the keys, called *weak keys*, there exists at least one IV satisfying the differential characteristic $\Delta_{in} \to \Delta_m$ in the first round, which is the differential characteristic of *quarterround* function described in Section 3.2. On the flip side, their experiments imply the existence of *strong keys* for which we can not find such IV. Recently, Dey et al. [12] provided a theoretical explanation about the differential characteristic mainly for two (out of eight) special cases, especially pointing out the case in which 12.5% of all keys becomes strong keys. But, for the other six cases, there is no theoretical explanation now. In the direction of finding a right pair, Dey et al. [14] proposed to achieve the right pairs by constructing a list consisting of at least one favorable IV for these *exploitable keys*. However, this treatment views the differential propagation as a black box, which results in a lot of data and storage overhead. In the following, it is shown that we fill in the missing parts

of the previous work and overcome the shortcomings of existing method to some extent based on the deeper analysis of differential equations of ARX structure.

For the differential characteristic $\Delta_{in} \rightarrow \Delta_m$ in the first round which is described in Section 3.2, there are totally five differential equations to satisfy:

$$\begin{cases} c[22] = Carry_{(c,d')}[22], & (8) \\ a'[2] = Carry_{(a',b')}[2], & (9) \\ c'[10] = Carry_{(c',d'')}[10], & (10) \\ c'[30] = Carry_{(c',d'')}[30], & (11) \\ d''[22] = Carry_{(c',d'')}[22], & (12) \end{cases}$$

where $Carry_{(x,y)}[i]$ denotes the i-th bit of carry vector $Carry_{(x,y)}$ of modular addition between x and y, i.e. $x \boxplus y = x \oplus y \oplus Carry_{(x,y)}$. As shown in Fig. 1, Eqs. (8) and (9) guarantee that the difference is not propagated to next bits in the second and third addition modular of *quarterround* function respectively, Eqs. (10), (11) and (12) guarantee that the difference is not propagated to next bits in the fourth addition of *quarterround* function.

In the initialization of ChaCha, a fixed constant is loaded into branch a, an unknown secret key is loaded into branches b and c. Therefore, we can not control branches a, b and c. Only branch d can be varied to satisfy the differential equations given the secret key. First, it is observed that at least one bit of d is almost linearly involved in Eqs. (10), (11) and (12). Therefore, we can always flip the linear bits to satisfy corresponding equations. We summarized this observation in the following.

Observation of last three differential equations.

- Eq. (10): $d[26] = Carry_{(c',d'')}[10] \oplus Carry_{(c,d')}[10] \oplus c[10] \oplus a'[26]$.
- Eq. (11): $d[14] = Carry_{(c',d'')}[30] \oplus Carry_{(c,d')}[30] \oplus c[30] \oplus a'[14]$.
- Eq. (12): $d[30] \oplus d[18] = Carry_{(c',d'')}[22] \oplus Carry_{(a',b')}[14] \oplus Carry_{(c,d')}[2] \oplus a'[30] \oplus a'[14] \oplus b[2] \oplus c[2] \oplus a'[18]$.

Thus, based on the above observation, the last three equations are satisfied by flipping $d[26]$, $d[14]$ and $d[30]$, which is verified by experiments given later.

As for the first two differential equations, no obvious linear bits of b are involved in equations. Therefore, we could not simply flip some bits of d to satisfy the two equations. It is noted that the two equations can not always be satisfied, and specially, Dey et al. [12] discussed two special cases where Eq. (9) is always established and must not be established, i.e. $a'[2:0] \in \{000, 100\}$. For details about their results, please refer to Theorems 5 and 6 in [12]. Next, we give a more in-depth analysis of Eqs. (8) and (9) by considering them together, which results in a theoretical interpretation for other six complex cases. The analysis stems from an observation of interaction between the first two equations which control differential propagation of two consecutive modular addition.

Observation of the first two differential equations. Considering an internal variable $c'[21:20]$, we further analyze the interaction between constraints of Eqs. (8) and (9). Since the relation $b' = (b \oplus c') \lll 12$, we

can obtain the range of $c'[21:20]$ from the constraints of $b'[1:0]$ derived by the assumption that Eq. (9) holds. Precisely, assume that Eq. (9) holds, we have that $c'[21:20] \in C'^{b[21:20]}_{a'[2:0]} \triangleq \{b'[1:0] \oplus b[21:20] \mid a'[2] = \lfloor (a'[1:0] + b'[1:0])/2^2 \rfloor\}$. Due to the fact that $Carry_{(c,d')}[22] = \begin{cases} 0, & if \quad c'[21:20] > c[21:20], \\ 1, & if \quad c'[21:20] < c[21:20], \\ Carry_{(c,d')}[20], & if \quad c'[21:20] = c[21:20], \end{cases}$ Eq. (8) also restricts the range of $c'[21:20]$, i.e. if $c[22] = 1$ then $c'[21:20] \leq c[21:20]$; otherwise $c[22] = 0$ then $c'[21:20] \geq c[21:20]$. Therefore, the two constraints of Eq. (8) and (9) may contradict.

Taking the two equations into account together, it is first identified that the cases Eqs. (8) and (9) can not hold at the same time. For this, a function $sign(\cdot,\cdot) : \{\alpha \mid \alpha \in \mathbb{Z}\} \times \mathbb{Z} \to \{1, -1, 0\}$ is defined as

$$sign(S, \beta) = \begin{cases} 1 \; if \quad \forall\, \alpha \in S, \alpha > \beta, \\ -1 \; if \quad \forall\, \alpha \in S, \alpha < \beta, \\ 0 \; otherwise, \end{cases}$$

where $sign(\cdot) \neq 0$ implies a contradiction between Eqs. (8) and (9). Otherwise, a heuristically optimal setting of d is given such that as few bits of b and c as possible are involved in. Formally, we write in a form of lemma below.

Lemma 3. *Given two words $a'[2:0] \in \{0,1\}^3$ and $b[21:20] \in \{0,1\}^2$, let $C'^{b[21:20]}_{a'[2:0]} \triangleq \{b'[1:0] \oplus b[21:20] \mid a'[2] = \lfloor (a'[1:0] + b'[1:0])/2^2 \rfloor\}$.*

- *For $a'[2:0] = 000$ and $\forall\, b[21:20] \in \{0,1\}^2$, $a'[2:0] \in \{111, 001\}$ and $b[21:20] \in \{01, 10\}$, no matter what value of $c[22:20]$, Eqs. (8) and (9) can hold simultaneously, specially with the following setting.*
  If $c[21:20] = min(C'^{b[21:20]}_{a'[2:0]})$ and $c[22] = 1$, or $c[21:20] = max(C'^{b[21:20]}_{a'[2:0]})$ and $c[22] = 0$,

$$\begin{cases} d[5:4] = a'[5:4] \oplus (3 \times c[22]) \\ d[3-i] = a'[3-i] \oplus c[19-i] \oplus 1, \quad i \in \{0, 1, \cdots, t-1\} \\ d[3-t] = a'[3-t] \oplus c[19-t] \end{cases} \tag{13}$$

  where $t \in \{0, 1, \cdots, 19\}$ is the minimum integer such that $c[19-t] = c[22]$. Otherwise,

$$\begin{cases} d[5:4] = a'[5:4] \oplus (2 \times c[22] + 1 - c[19]) \\ d[3] = a'[3] \oplus c[19] \end{cases} \tag{14}$$

- *For $a'[2:0] = 100$, Eq (9) must not be satisfied.*
- *For $a'[2:0] \in \{101, 110, 010, 011\}$ and $\forall\, b[21:20] \in \{0,1\}^2$, $a'[2:0] \in \{111, 001\}$ and $b[21:20] \in \{00, 11\}$,*
  - *When $c[21:20] \notin C'^{b[21:20]}_{a'[2:0]}$, if $c[22] = \frac{1}{2}(1 + sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20]))$ then Eqs. (8) and (9) can not hold simultaneously; otherwise, then Eqs.*

(8) and (9) can hold simultaneously, specially with the following setting,

$$\begin{cases} d[5:4] = a'[5:4] \oplus \left(min(C'^{b[21:20]}_{a'[2:0]}) - c[21:20] + c[22] \times 4 - c[19]\right) \\ d[3] = a'[3] \oplus c[19] \end{cases} \tag{15}$$

- *When* $c[21:20] \in C'^{b[21:20]}_{a'[2:0]}$*, Eqs. (8) and (9) can hold simultaneously, specially with the following setting. If* $c[21:20] = min(C'^{b[21:20]}_{a'[2:0]})$ *and* $c[22] = 1$*, or* $c[21:20] = max(C'^{b[21:20]}_{a'[2:0]})$ *and* $c[22] = 0$*, the setting is same to Eqs. (13); otherwise, the setting is same to Eqs. (14).*

For completeness, the proof of Lemma 3 is given in Supplementary Material B.1. In an attack, it is necessary to know how many unknown bits of secret key need to guess and how many public bits are consumed to satisfy the first two equations. For this purpose, the following corollary is directly derived by the above lemma.

Corollary 1. *For* $a'[2:0] \in \mathbb{F}_2^3$ *and* $a'[2:0] \neq 100$*, with the setting of Lemma 3,*

- *In Eq. (13), there are at most 22 bits of* d *i.e.* $d[5:0]$ *and* $d[31:16]$*, and at most 32 bits of* b *and 23 bits of* c *i.e.* $c[22:0]$ *are needed to guess. Note that this setting is used only when* $c[21:20] = min(C'^{b[21:20]}_{a'[2:0]})$ *and* $c[22] = 1$*, or* $c[21:20] = max(C'^{b[21:20]}_{a'[2:0]})$ *and* $c[22] = 0$*, which are* 25% *of all cases.*
  If $t \leq 3$*, with probability* $\frac{15}{16}$*, at most 6 bits of* d *i.e.* $d[5:0]$ *are restricted, and at most 8 bits of* b *i.e.* $a'[5:0]$ *and* $b[21:20]$ *and 7 bits of* c *i.e.* $c[22:16]$ *are needed to guess.*
- *In Eqs. (14) and (15), there are 3 bits of* d *i.e.* $d[5:3]$*, and at most 8 bits of* b *i.e.* $a'[5:0]$ *and* $b[21:20]$ *and 4 bits of* c *i.e.* $c[22:19]$ *needed to guess.*

5.1 Reducing Complexities

Based on our observations of the five differential equations, we propose to use a concise and efficient way to generate a right pair for the given differential $\Delta_{in} \to \Delta_m$ in the first round. Given a weak key, from Corollary 1, we know that with probability greater than $\frac{15}{16}$ one can use at most six IV bits $\{0:5\}$ to satisfy the first two equations. And we can flip three IV bits $\{14, 26, 30\}$ to satisfy the last three equations based on the aforementioned observation. Inspired by the analysis of five differential equations, a proposition is presented as follows.

Proposition 3. *Let* $k_{ID} \in \mathbb{F}_2^{64}$*, if* k_{ID} *is a weak key that there exists at least one* $v \in \mathbb{F}_2^{32}$ *satisfying the given differential, then with probability greater than* $\frac{15}{16} = 93.75\%$*, one can find* v *in a subspace* $\mathbb{S}_{V_{nfree}} \subseteq \mathbb{F}_2^{32}$ *with* 2^9 *IVs such that* $((k_{ID}, v), (k_{ID}, v'))$ *forms a right pair, where the subspace* $\mathbb{S}_{V_{nfree}} = \{(a_{31}, a_{30}, \cdots, a_0) \mid a_i = 0 \text{ for } i \notin V_{nfree}\}$ *and the index subset* $V_{nfree} = \{0:5, 14, 26, 30\}$.

We performed experiments to verify the proposition. As a result, for 66.4% keys among all the keys we find one v in the subspace in Proposition 3 that satisfies the given differential, which confirms Proposition 3, i.e., $\geq 70\% \times \frac{15}{16} = 65.6\%$.

5.2 Theoretical Interpretation of Strong Key

According to Lemma 3, we derive a lower bound for the percentage of strong keys $k_{ID} \in \mathbb{F}_2^{64}$ that there is no IV $v \in \mathbb{F}_2^{32}$ satisfying the given differential $\Delta_{in} \to \Delta_m$ in the first round for each case of $a'[2:0]$.

Theorem 1. *Among all the keys, there are at least* 29.6875% *strong keys* $k_{ID} \in \mathbb{F}_2^{64}$ *that there is no IV* $v \in \mathbb{F}_2^{32}$ *satisfying the given differential, for different* $a'[2:0]$ *we have*

$$\tau = \begin{cases} 0 & \text{if} \quad a'[2:0] = 000, \\ 0.0625 & \text{if} \quad a'[2:0] = 001, \\ 0.25 & \text{if} \quad a'[2:0] = 010, \\ 0.375 & \text{if} \quad a'[2:0] = 011, \end{cases} \quad \begin{cases} 1 & \text{if} \quad a'[2:0] = 100, \\ 0.375 & \text{if} \quad a'[2:0] = 101, \\ 0.25 & \text{if} \quad a'[2:0] = 110, \\ 0.0625 & \text{if} \quad a'[2:0] = 111. \end{cases}$$

where τ *denotes the percentage of strong keys derived by Lemma 3.*

For completeness, we give the proofs of Theorem 1 in Supplementary Material B.2, along with two specific examples of strong keys. With analyzing six out of eight more complex cases, our theoretical derivation of strong keys is basically consistent with the experimental result 30%.

6 Applications: ChaCha7/7.5$^{\oplus}$/6

In this section, we present several key-recovery attacks on ChaCha7, 7.5$^{\oplus}$ and ChaCha6 utilizing the new proposed techniques in Sections 4 and 5. In the attacks, we use the forward differential characteristic in Table 3.

6.1 Attacks against ChaCha7

First the inverse structure of ChaCha7 is analyzed, which is used to determine all the syncopated segments of internal state in Algorithm 1. The last round of ChaCha7 is illustrated in Fig. 5. For the inversion of last *quarterround* function, since the following relationships,

$$\begin{cases} x_7^{(7)} = z_7^{(7)} \boxminus k_3 \\ x_{11}^{(7)} = z_{11}^{(7)} \boxminus k_7 \\ y_{11}^{(6)} = (z_{11}^{(7)} \boxminus \underset{\cdot\cdot\cdot\cdot}{x_{15}^{(7)}}) \boxminus k_7 \\ x_{11}^{(6)} = (z_{11}^{(7)} \boxminus \underset{\cdot\cdot\cdot\cdot}{x_{15}^{(7)}} \boxminus \underset{\cdot\cdot\cdot\cdot}{y_{15}^{(6)}}) \boxminus k_7 \end{cases}, \quad \begin{cases} \underset{\cdot\cdot\cdot\cdot}{x_3^{(7)}} = z_3^{(7)} \boxminus c_3 \\ \underset{\cdot\cdot\cdot\cdot}{x_{15}^{(7)}} = z_{15}^{(7)} \boxminus v_2 \\ \underset{\cdot\cdot\cdot\cdot}{y_{15}^{(6)}} = (\underset{\cdot\cdot\cdot\cdot}{x_{15}^{(7)}} \ggg 8) \oplus \underset{\cdot\cdot\cdot\cdot}{x_3^{(7)}} \end{cases}$$

the syncopation technique can be applied on words $x_7^{(7)}$, $x_{11}^{(7)}$, $y_{11}^{(6)}$, and $x_{11}^{(6)}$ with using the property of modular subtraction (Lemma 2), where the words underlined by $\underset{\cdot}{x}$ are known since they can be derived from constants, IV and keystream blocks. The analysis is similar for the other three *quarterround* functions, and we mark all the words by red in Fig. 5 which the syncopation technique can

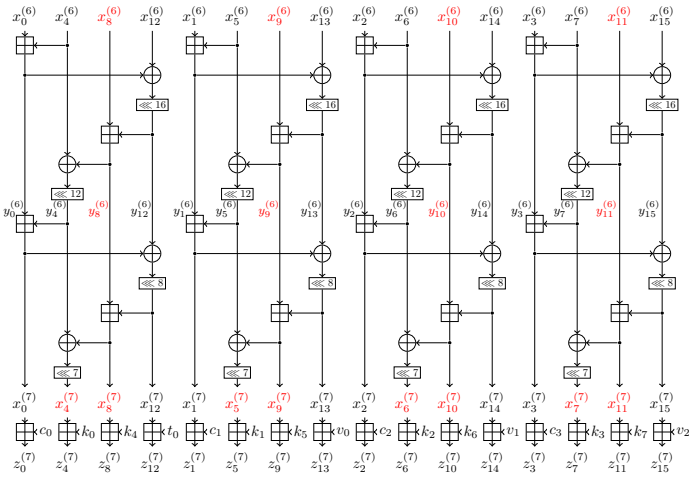


Fig. 5. Last round of ChaCha7.

be applied on. Given a (C)PNB set, Algorithm 1 is performed, and for the words marked by red in Fig. 5, the syncopated segments of internal state and corresponding conditions on restricted bits are determined according to the syncopated segments and restricted segments of secret key. The attacks with better parameters are obtained for ChaCha7 by setting $u = 1$, i.e., only using the syncopated segments with a one-bit restricted segment.

The set of CPNBs. In the first step, we obtain a preliminary shortlisting of 151 possible CPNBs by setting a threshold 2^{-5} for the conditional neutrality measure, which is given in Supplementary Material C.1. Then Algorithm 2 is executed with input of the preliminary shortlisting, and the size of CPNB set is exhaustively searched. As a result, two sets with 74 CPNBs and 89 CPNBs are selected by minimizing time complexity under the requirement that data complexity does not exceed the amount of available IV. The set of 74 CPNBs is $\{2, 3, 4, 5, 47, 48, 49, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 90, 91, 102, 103, 104, 105, 106, 107, 108, 109, 110, 123, 124, 125, 126, 127, 155, 156, 157, 158, 159, 168, 169, 191, 192, 193, 194, 199, 200, 207, 208, 211, 212, 219, 220, 221, 222, 223, 224, 225, 226, 244, 245, 246, 247, 255\}$. In Fig. 6, the 74 gray lines represent 74 CPNBs, i.e. syncopation bits according to their indexes of secret key, and the blue and red lines represent the non-CPNBs where the syncopated segments k^o and restricted segments $k^{\mathcal{R}}$ of secret key are denoted by areas of blue and red lines respectively. The profitable syncopated segments for the 74 CPNBs are summarized in Table 4 along with 27 conditions, where we ignore the carry bits after syncopated segments. The set of 89 CPNBs is $\{2, 3, 4, 5, 6, 14, 15, 26, 47, 48, 49, 51, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 90, 91, 95, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 135, 136, 147, 155, 156, 157, 158, 159, 168, 169, 170, 191, 192,$

Table 4. Syncopated segments of 74 CPNBs, where "-" denotes $w = 0$.

syncopated segment $\leftarrow$ condition	(i, j_1, j_2)
T0: $x_4^{(7)}[j_2 : j_1] \leftarrow k_0[i] \neq z_4^{(7)}[i]$	$(6, 7, 31)$
T0: $x_5^{(7)}[j_2 : j_1] \leftarrow k_1[i] \neq z_5^{(7)}[i]$	$(18, 19, 31)$
T0: $x_6^{(7)}[j_2 : j_1] \leftarrow k_2[i] \neq z_6^{(7)}[i]$	$\{(12, \text{-}, \text{-}), (24, 25, 25), (28, 29, 31)\}$
T0: $x_7^{(7)}[j_2 : j_1] \leftarrow k_3[i] \neq z_7^{(7)}[i]$	$(15, 16, 26)$
T0: $x_9^{(7)}[j_2 : j_1] \leftarrow k_5[i] \neq z_9^{(7)}[i]$ T1: $y_9^{(6)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7)} \boxminus x_{13}^{(7)})[i]$ T2: $x_9^{(6)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7)} \boxminus x_{13}^{(7)} \boxminus y_{13}^{(6)})[i]$	$(10, 11, 30)$
T0: $x_{10}^{(7)}[j_2 : j_1] \leftarrow k_6[i] \neq z_{10}^{(7)}[i]$ T1: $y_{10}^{(6)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7)} \boxminus x_{14}^{(7)})[i]$ T2: $x_{10}^{(6)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7)} \boxminus x_{14}^{(7)} \boxminus y_{14}^{(6)})[i]$	$\{(3, 4, 6), (9, 10, 14),$ $(17, 18, 18), (21, 22, 26)\}$
T0: $x_{11}^{(7)}[j_2 : j_1] \leftarrow k_7[i] \neq z_{11}^{(7)}[i]$ T1: $y_{11}^{(6)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7)} \boxminus x_{15}^{(7)})[i]$ T2: $x_{11}^{(6)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7)} \boxminus x_{15}^{(7)} \boxminus y_{15}^{(6)})[i]$	$\{(3, 4, 19), (24, 25, 30)\}$

$193, 194, 195, 199, 200, 207, 208, 211, 212, 219, 220, 221, 222, 223, 224, 225, 226, 227, 232, 244, 245, 246, 247, 255\}$. The profitable syncopated segments of the 89 CPNBs are given in Table 5 along with 40 conditions.

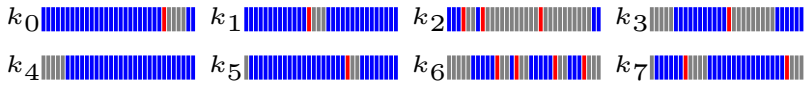


Fig. 6. The classification of all key bits for 74 CPNBs, where gray line represents CPNBs, and blue and red lines represent non-PNBs, from left to right for $k_i[31 : 0]$.

From Table 4, we remove 13 inessential conditions whose coordinates in table are $\{(\text{T0}, 8), (\text{T0}, 11), (\text{T1}, 8), (\text{T1}, 9), (\text{T1}, 10), (\text{T1}, 11), (\text{T1}, 12), (\text{T2}, 6), (\text{T2}, 8), (\text{T2}, 9), (\text{T2}, 10), (\text{T2}, 11), (\text{T2}, 12)\}$, where $(\text{T}i, j)$ locates a condition with type Ti and top-to-bottom order j in right column of table, $0 \leq i \leq 2$ and $0 \leq j \leq 12$. For example, coordinate $(\text{T0}, 8)$ locates the syncopated segment and its condition $x_{10}^{(7)}[14 : 10] \leftarrow k_6[9] \neq z_{10}^{(7)}[9]$. In the experiment of 2^{10} random keys and $2^{16.3}$ IVs for each key, the conditional correlation is theoretically estimated as $\epsilon_a' = 0.2365$. We have conducted a verification experiment for this theoretical calculation, in which 2^6 random keys are used with $2^{32.6}$ IVs for each key. The result of experiment shows that the conditional correlation is $\epsilon_a' = 0.2369$, which is very close to the theoretical one and justifies theoretical calculation of conditional correlation. Therefore, the conditional backward correlation is $\varepsilon_a = 2^{-4.2}$, and with $n = 74$, $\theta_1 = 14$ and $\alpha = 48$, the time and data complexities are $T = 2^{213.9}$ and $N = 2^{59.9}$ respectively.

Table 5. Syncopated segments of 89 CPNBs, where "-" denotes $w = 0$.

syncopated segment $\leftarrow$ condition	(i, j_1, j_2)
T0: $x_4^{(7)}[j_2 : j_1] \leftarrow k_0[i] \neq z_4^{(7)}[i]$	$\{(7, 8, 13), (16, 17, 25), (27, 28, 31)\}$
T0: $x_5^{(7)}[j_2 : j_1] \leftarrow k_1[i] \neq z_5^{(7)}[i]$	$\{(18, -, -), (20, 21, 31)\}$
T0: $x_6^{(7)}[j_2 : j_1] \leftarrow k_2[i] \neq z_6^{(7)}[i]$	$\{(12, -, -), (24, 25, 25), (28, 29, 30)\}$
T0: $x_7^{(7)}[j_2 : j_1] \leftarrow k_3[i] \neq z_7^{(7)}[i]$	$\{(16, 17, 18), (20, 21, 26)\}$
T0: $x_8^{(7)}[j_2 : j_1] \leftarrow k_4[9] \neq z_8^{(7)}[9]$ T1: $y_8^{(6)}[j_2 : j_1] \leftarrow k_4[9] \neq (z_8^{(7)} \boxminus x_{12}^{(7)})[9]$ T2: $x_8^{(6)}[j_2 : j_1] \leftarrow k_4[9] \neq (z_8^{(7)} \boxminus x_{12}^{(7)} \boxminus y_{12}^{(6)})[9]$	$\{(9, 10, 18), (20, 21, 26)\}$
T0: $x_9^{(7)}[j_2 : j_1] \leftarrow k_5[i] \neq z_9^{(7)}[i]$ T1: $y_9^{(6)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7)} \boxminus x_{13}^{(7)})[i]$ T2: $x_9^{(6)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7)} \boxminus x_{13}^{(7)} \boxminus y_{13}^{(6)})[i]$	$(11, 12, 30)$
T0: $x_{10}^{(7)}[j_2 : j_1] \leftarrow k_6[i] \neq z_{10}^{(7)}[i]$ T1: $y_{10}^{(6)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7)} \boxminus x_{14}^{(7)})[i]$ T2: $x_{10}^{(6)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7)} \boxminus x_{14}^{(7)} \boxminus y_{14}^{(6)})[i]$	$\{(4, 5, 6), (10, 11, 14),$ $(17, 18, 18), (21, 22, 26)\}$
T0: $x_{11}^{(7)}[j_2 : j_1] \leftarrow k_7[i] \neq z_{11}^{(7)}[i]$ T1: $y_{11}^{(6)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7)} \boxminus x_{15}^{(7)})[i]$ T2: $x_{11}^{(6)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7)} \boxminus x_{15}^{(7)} \boxminus y_{15}^{(6)})[i]$	$\{(4, 5, 7), (9, 10, 19), (24, 25, 30)\}$

Analysis of complexities. The analysis in Section 5 is used to satisfy the differential in the first round for the forward distinguisher in Table 3. According to Corollary 1, we should know fifteen key bits $\{32 : 37, 52, 53, 176 : 182\}$ to satisfy the first two differential equations, and they are not involved in the set of 74 CPNBs. Therefore, we can freely satisfy the first two differential equations by setting at most six IV bits $\{0 : 5\}$ as done in Lemma 3. The overall attack process is as follows: in Step 1 of Algorithm 3 one prepares 2^6 structures of data with N pairs in each structure by traversing six IV bits $\{0 : 5\}$, and in Step 3 of Algorithm 3 one first selects the structure which satisfies the first two differential equations according to the value of guessed key, and then as specified in Algorithm 3 chooses the subsets satisfying 28 equations in conditions and then performs the statistic testing. To satisfy the last three differential equations, this process needs to be performed 2^3 times. The N pairs should be generated by varying the other three branches at which no input difference is imposed, i.e. $N \leq 2^{96}$, without affecting the propagation of differential in the first round. As a result, the total time complexity is $2^{213.9+3} = 2^{216.9}$, and data complexity is $2^{59.9+9} = 2^{68.9}$. As for the memory complexity is of $2^{6+|k^{\mathcal{R}}|}N^* = 2^{48.9}$ since we only store the subsets used later in each structure with ignoring the subsets that must not satisfy 28 equations in conditions.

Similarly, we remove 13 inessential conditions from the 40 conditions of 89 CPNBs, and their coordinates in Table 5 are $\{(\mathrm{T0}, 14), (\mathrm{T0}, 17), (\mathrm{T1}, 14), (\mathrm{T1}, 15), (\mathrm{T1}, 16), (\mathrm{T1}, 18), (\mathrm{T1}, 19), (\mathrm{T2}, 11), (\mathrm{T2}, 12), (\mathrm{T2}, 15), (\mathrm{T2}, 16), (\mathrm{T2}, 18), (\mathrm{T2}, 19)\}$, where $(\mathrm{T}i, j)$ locates a condition with type Ti and top-to-bottom order j in right

column of table, $0 \le i \le 2$ and $0 \le j \le 19$. By the experiment with 2^{10} random keys and $2^{20.3}$ IVs for each key, the conditional correlation ϵ'_a is theoretically estimated as $2^{-4.14}$. Therefore, the conditional backward correlation is $\varepsilon_a = 2^{-8.28}$, and with $n = 89$, $\theta_1 = 27$ and $\alpha = 54$, we obtain another attack with less time complexity of $2^{207.3+3} = 2^{210.3}$ and data complexity of $2^{94.3+9} = 2^{103.3}$.

6.2 Attacks against ChaCha7.5$^{\oplus}$

To get closer to ChaCha8, ChaCha7.5$^{\oplus}$ is defined as a round-reduced version without the last bit-wise XOR and left rotation in last round function compared with ChaCha7.5.

Since the analysis of ChaCha7.5$^{\oplus}$ is similar to the one of ChaCha7, we only focus on different points and present the main parameters of attacks against ChaCha7.5$^{\oplus}$ in the following. The last round of ChaCha7.5$^{\oplus}$ is illustrated in Fig. 7 where the notation $1^{\oplus}$ means no last XOR and left rotation in one round function. For the inversion of last *quarterround* function in the 7-th round, since the relationship $x_3^{(7)} = (\overbrace{x_3^{(7.5^{\oplus})}}^{\cdots\cdots} \boxminus z_4^{(7.5^{\oplus})}) \boxplus k_0$ and $\overbrace{x_3^{(7.5^{\oplus})}}^{\cdots\cdots} = z_3^{(7.5^{\oplus})} \boxminus c_3$, the syncopation technique can be applied on $x_3^{(7)}$ with using the property of modular addition (Lemma 1). In Fig. 7, we mark all the words on which the syncopation technique can be applied with using the property of modular addition (Lemma 1) by green and subtraction (Lemma 2) by red. Similarly, we set $u = 1$ and use the syncopated segments with a one-bit restricted segment in Algorithm 1.

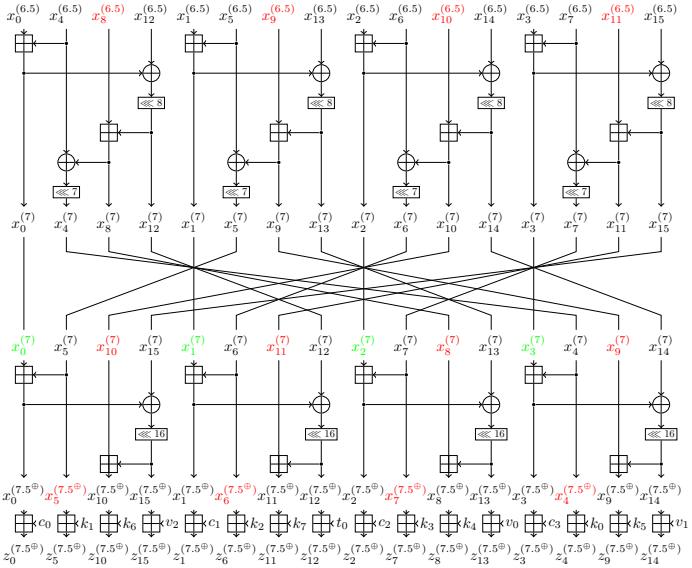


Fig. 7. Last $1^{\oplus}$ round of ChaCha7.5$^{\oplus}$.

The set of CPNBs. A preliminary shortlisting of 119 possible CPNBs is obtained by setting 2^{-5} as a threshold for the conditional neutrality measure, which is given in Supplementary Material C.2. Then Algorithm 2 is executed with the preliminary shortlisting, and a set of 54 CPNBs is selected to launch the key-recovery attack. The 54 CPNBs are $\{74, 75, 83, 90, 91, 92, 95, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 155, 156, 157, 158, 159, 168, 169, 170, 191, 192, 193, 194, 195, 199, 200, 204, 207, 208, 211, 212, 213, 216, 219, 220, 221, 222, 223, 224, 225, 226, 227, 232, 244, 245, 246, 247, 255\}$, and its syncopated segments and 30 conditions are given in Supplementary Material C.2.

Analysis of complexities. Two inessential conditions are removed from the 30 conditions of 54 CPNBs. By the experiment with 2^{10} random keys and $2^{20.3}$ IVs for each key, the conditional correlation ϵ'_a is theoretically estimated as $2^{-4.3}$. Therefore, the conditional backward correlation is $\varepsilon_a = 2^{-8.6}$, and with $n = 54$, $\theta_1 = 28$ and $\alpha = 19$, an attack is obtained with time complexity of $2^{241.9+3} = 2^{244.9}$ and data complexity of $2^{95.9+9} = 2^{104.9}$.

Another attack on ChaCha7.5$^{\oplus}$. There is another alternative way to find the right pairs in the first round with help of a pre-computed list, as proposed in [14]. The attacker pre-computes all the needed right pairs offline, and keep them in a list. In order to find a favorable IV in the pre-computed list for every exploitable key which is defined in Section 3.2, we should ensure that all the CPNBs of ID column, denoted by $CPNB_{ID}$, satisfy $CPNB_{ID} \subseteq K_{nmem}$, i.e. $K_{mem} \subseteq Non\text{-}CPNB_{ID}$. Next, we list the main results of the way of pre-computing a list to find right pairs. A set of 49 CPNBs is selected, that is $CPNB = \{74, 75, 83, 90, 91, 95, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 155, 156, 157, 158, 159, 168, 169, 170, 191, 192, 193, 194, 199, 204, 207, 208, 211, 212, 216, 219, 220, 221, 222, 223, 224, 225, 226, 232, 244, 245, 246, 247, 255\}$. There are four CPNBs in the ID column, i.e. $CPNB_{ID} = \{168, 169, 170, 191\}$. A greedy algorithm is used to construct K_{nmem} such that K_{nmem} includes as many key bits as possible, especially for the key bits in $CPNB_{ID}$, while the probability of exploitable keys is high. A subset of size 20 is obtained, i.e. $K_{nmem} = \{168, 191, 169, 171, 183, 172, 173, 184, 174, 175, 185, 163, 176, 40, 41, 164, 59, 177, 186, 160\}$. Among of them three bits are from $CPNB_{ID}$, and one key bit should be excluded from the CPNB set for a high probability of exploitable keys, i.e. $K_{exclude} = \{170\}$. As a result, there are about $p_{exp} = 55.4\%$ exploitable keys among all keys with respect to the subspace $\mathbb{S}_{K_{nmem}}$. Since the dimension of $\mathbb{S}_{K_{mem}}$ is $64 - 20 = 44$, we should construct a pre-computed list of approximately $p_{exp} \times 2^{44} \approx 2^{43.1}$ possible exploitable keys and their favorable IVs. The final set of 48 CPNBs is $CPNB_{final} = CPNB \setminus K_{exclude}$, and its syncopated segments and 30 conditions are given in Supplementary Material C.2.

Analysis of complexities. The conditional backward correlation of final set of 48 CPNBs is theoretically estimated as $\varepsilon_a = 2^{-6.1}$. Therefore, with $n = 48$, $\theta_1 = 30$ and $\alpha = 17$, the corresponding data and time complexities of attack are $2^{94.8+31} = 2^{125.8}$ and $2^{242.9}$ respectively.

6.3 Attack against ChaCha6

Here, we give a new analysis of ChaCha6 under the refined framework. Since the analysis of ChaCha6 is similar to the one of ChaCha7, next we only show the main results of the attack, and for details please refer to Supplementary Material C.3. A partial key-recovery attack is presented in [2], where 36 key bits can be restored. For a better comparison, only the significant key bits are recovered in our attack as depicted in Step 1–5 of Algorithm 3. As estimated in [2], the attack should be repeated for 2^5 times to find a right pair in the first round. There are two main parameters which affect the complexities of attack, the number m of significant key bits, and the backward correlation ε_a. Given the main parameters, we determine $N^* = (\frac{\sqrt{(m+5)\log 4}+3\sqrt{1-(\varepsilon_a\varepsilon_d)^2}}{\varepsilon_a\varepsilon_d})^2$, the data complexity $N = 2^5 \times \frac{1}{q} \times 2N^*$, and time complexity $T = 2^5 \times \frac{1}{q} \times 2N^* + 2^5 \times 2N^* \times 2^{m}$ [4], where $q = ((\frac{1}{2})^{\theta_1}(\frac{3}{4})^{\theta_2})^2$, θ_1 and θ_2 are the numbers of syncopated segments with one-bit and two-bit restricted segments respectively. By performing Algorithm 2 with a greedy criteria of minimizing the time complexity, we obtain an attack of ChaCha6 which has 211 CPNBs with 27 syncopated segments with conditions ($\theta_1 = 18$, $\theta_2 = 9$). Our experiment has verified the conditional backward correlation of 211 CPNBs, that is $|\varepsilon_a| = 0.81$. Therefore, with $m = 45$, $\theta_1 = 18$ and $\theta_2 = 9$, our attack on ChaCha6 takes time and data complexities of $2^{75.7}$ and $2^{73.7}$ respectively, which is faster and recovers nine more key bits, i.e. 45 key bits, than previous one [2]. The specific significant key bits and corresponding CPNB's syncopated segments and conditions are given in Supplementary Material C.3.

7 Conclusion

In this paper, we present PNB-based differential cryptanalysis of ChaCha with *syncopation*. Specifically, we propose the *syncopation* technique, and refine the key-recovery attack based on PNB method with this new technique. The attacks on ChaCha6 and ChaCha7 are improved using the new technique in the PNB-based cryptanalysis. Furthermore, ChaCha7.5$^{\oplus}$ is defined for getting closer to ChaCha8, and several attacks are presented on ChaCha7.5$^{\oplus}$. As far as we know, this is the first attack on ChaCha7.5$^{\oplus}$ which is much more efficient than brute force. In a nutshell, we move a small but important step forward towards the analysis of ChaCha8. It is interesting and worth to consider how to apply the new techniques on the other ARX ciphers, such as Salsa and Chaskey.

Acknowledgments. We are very grateful to the anonymous reviewers for their detailed comments and helpful suggestions of our manuscript. We would also like to thank Chengan Hou for his careful proofreading of this manuscript.

[4] The complexity was doubled in the analysis of [2], we multiply it by a factor of two here for a fair comparison.

References

1. Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of latin dances: Analysis of salsa, chacha, and rumba. In: Nyberg, K. (ed.) Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 470–488. Springer (2008). https://doi.org/10.1007/978-3-540-71039-4_30, https://doi.org/10.1007/978-3-540-71039-4_30
2. Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 329–358. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_12, https://doi.org/10.1007/978-3-030-56877-1_12
3. Bernstein, D.J.: The poly1305-aes message-authentication code. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3557, pp. 32–49. Springer (2005). https://doi.org/10.1007/11502760_3, https://doi.org/10.1007/11502760_3
4. Bernstein, D.J.: Chacha, a variant of salsa20 (2008), https://cr.yp.to/chacha/chacha-20080128.pdf, https://cr.yp.to/chacha/chacha-20080128.pdf
5. Bernstein, D.J.: The salsa20 family of stream ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986, pp. 84–97. Springer (2008). https://doi.org/10.1007/978-3-540-68351-3_8, https://doi.org/10.1007/978-3-540-68351-3_8
6. Biham, E., Carmeli, Y.: An improvement of linear cryptanalysis with addition operations with applications to FEAL-8X. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8781, pp. 59–76. Springer (2014). https://doi.org/10.1007/978-3-319-13051-4_4, https://doi.org/10.1007/978-3-319-13051-4_4
7. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990). https://doi.org/10.1007/3-540-38424-3_1, https://doi.org/10.1007/3-540-38424-3_1
8. Choudhuri, A.R., Maitra, S.: Significantly improved multi-bit differentials for reduced round salsa and chacha. IACR Trans. Symmetric Cryptol. **2016**(2), 261–287 (2016). https://doi.org/10.13154/tosc.v2016.i2.261-287, https://doi.org/10.13154/tosc.v2016.i2.261-287
9. Coutinho, M., Neto, T.C.S.: Improved linear approximations to arx ciphers and attacks against chacha. Cryptology ePrint Archive, Paper 2021/224 (2021), https://eprint.iacr.org/2021/224, https://eprint.iacr.org/2021/224
10. Coutinho, M., Neto, T.C.S.: Improved linear approximations to ARX ciphers and attacks against chacha. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21,

2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 711–740. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_25, https://doi.org/10.1007/978-3-030-77870-5_25
11. Coutinho, M., Passos, I., Vásquez, J.C.G., de Mendonça, F.L.L., de Sousa, R.T., Borges, F.: Latin dances reloaded: Improved cryptanalysis against salsa and chacha, and the proposal of forró. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13791, pp. 256–286. Springer (2022). https://doi.org/10.1007/978-3-031-22963-3_9, https://doi.org/10.1007/978-3-031-22963-3_9
12. Dey, S., Dey, C., Sarkar, S., Meier, W.: Revisiting cryptanalysis on chacha from crypto 2020 and eurocrypt 2021. IEEE Trans. Inf. Theory **68**(9), 6114–6133 (2022). https://doi.org/10.1109/TIT.2022.3171865, https://doi.org/10.1109/TIT.2022.3171865
13. Dey, S., Garai, H.K., Maitra, S.: Cryptanalysis of reduced round chacha- new attack and deeper analysis. Cryptology ePrint Archive, Paper 2023/134 (2023), https://eprint.iacr.org/2023/134, https://eprint.iacr.org/2023/134
14. Dey, S., Garai, H.K., Sarkar, S., Sharma, N.K.: Revamped differential-linear cryptanalysis on reduced round chacha. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13277, pp. 86–114. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_4, https://doi.org/10.1007/978-3-031-07082-2_4
15. Dey, S., Sarkar, S.: Improved analysis for reduced round salsa and chacha. Discret. Appl. Math. **227**, 58–69 (2017). https://doi.org/10.1016/j.dam.2017.04.034, https://doi.org/10.1016/j.dam.2017.04.034
16. Dey, S., Sarkar, S.: Proving the biases of salsa and chacha in differential attack. Des. Codes Cryptogr. **88**(9), 1827–1856 (2020). https://doi.org/10.1007/s10623-020-00736-9, https://doi.org/10.1007/s10623-020-00736-9
17. eSTREAM: The ecrypt stream cipher project., https://www.ecrypt.eu.org/stream/
18. IANIX: Chacha usage & deployment., https://ianix.com/pub/chacha-deployment.html
19. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 17–25. Springer (1994). https://doi.org/10.1007/3-540-48658-5_3, https://doi.org/10.1007/3-540-48658-5_3
20. Langley, A., Chang, W., Mavrogiannopoulos, N., Strömbergson, J., Josefsson, S.: Chacha20-poly1305 cipher suites for transport layer security (TLS). RFC **7905**, 1–8 (2016). https://doi.org/10.17487/RFC7905, https://doi.org/10.17487/RFC7905
21. Leurent, G.: Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 344–371. Springer (2016).

https://doi.org/10.1007/978-3-662-49890-3_14, https://doi.org/10.1007/978-3-662-49890-3_14
22. Maitra, S.: Chosen IV cryptanalysis on reduced round chacha and salsa. Discret. Appl. Math. **208**, 88–97 (2016). https://doi.org/10.1016/j.dam.2016.02.020, https://doi.org/10.1016/j.dam.2016.02.020
23. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33, https://doi.org/10.1007/3-540-48285-7_33
24. Miyashita, S., Ito, R., Miyaji, A.: Pnb-focused differential cryptanalysis ofăchacha stream cipher. In: Nguyen, K., Yang, G., Guo, F., Susilo, W. (eds.) Information Security and Privacy. pp. 46–66. Springer International Publishing, Cham (2022)
25. Nir, Y.: Chacha20, poly1305, and their use in the internet key exchange protocol (IKE) and ipsec. RFC **7634**, 1–13 (2015). https://doi.org/10.17487/RFC7634, https://doi.org/10.17487/RFC7634
26. Shi, Z., Zhang, B., Feng, D., Wu, W.: Improved key recovery attacks on reduced-round salsa20 and chacha. In: Kwon, T., Lee, M., Kwon, D. (eds.) Information Security and Cryptology - ICISC 2012 - 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7839, pp. 337–351. Springer (2012). https://doi.org/10.1007/978-3-642-37682-5_24, https://doi.org/10.1007/978-3-642-37682-5_24

Supplementary Material

A Proofs for Syncopation

In the section, the complete proofs for Lemma 1 and 2 are given, along with their corollaries.

A.1 Proof of Property for Modular Addition

Here, a complete proof for Lemma 1 is presented.

Proof. By considering modular addition $x = s \boxplus k$, the following holds

$$\begin{aligned} x[t+w-1:t] &= (s[t+w-1:t] + k[t+w-1:t] + Carry[t]) \ \ mod\ 2^w, \\ Carry[t+w] &= \lfloor (s[t+w-1:t] + k[t+w-1:t] + Carry[t])\,/2^w \rfloor, \end{aligned}$$

where $Carry[i]$ is i-th bit of carry vector, i.e. $Carry[i] \triangleq (s \boxplus k)[i] \oplus s[i] \oplus k[i]$. That is

$$x[t+w-1:t] = s[t+w-1:t] + k[t+w-1:t] + Carry[t] - Carry[t+w] \cdot 2^w,$$

and

$$Carry[t+w] = \begin{cases} 0 & if \quad s[t+w-1:t] + k[t+w-1:t] + Carry[t] < 2^w, \\ 1 & if \quad s[t+w-1:t] + k[t+w-1:t] + Carry[t] \geq 2^w. \end{cases}$$

With representing $x = s \boxplus k$ in a bit level, we have

$$x[t] = s[t] \oplus k[t] \oplus Carry[t],$$

where $Carry[t] = MAJ(Carry[t-1], s[t-1], k[t-1])$, and the majority function $MAJ(a, b, c) = ab \oplus ac \oplus bc$.

If $k[t-1] = s[t-1]$, then we have $Carry[t] = s[t-1]$. As for the case $k[t-1] \neq s[t-1]$, we have $Carry[t] = Carry[t-1]$. Moreover, if $k[t-2] = s[t-2]$, then we have $Carry[t] = Carry[t-1] = s[t-2]$. By analogy, we have that $Carry[t] = s[t-\iota]$ where ι is such that $k[t-1:t-\iota+1] \oplus s[t-1:t-\iota+1] = \vec{1}$ *and* $k[t-\iota] \oplus s[t-\iota] = 0$. □

Then we show the corollary of Lemma 1 when $w = 1$ and $u = 2$, which actually covers the previous result in [21].

Corollary 2 ([21]). *Let $t \geq 1$, and let $x, k, s \in \mathbb{F}_2^n$ and $x = s \boxplus k$. Then,*

$$\begin{aligned} x[t] &= s[t] + k[t] + Carry[t] - Carry[t+1] \cdot 2, \\ Carry[t+1] &= \begin{cases} 0 & if \quad s[t] + k[t] + Carry[t] < 2, \\ 1 & if \quad s[t] + k[t] + Carry[t] \geq 2, \end{cases} \end{aligned}$$

that is

$$x[t] = s[t] \oplus k[t] \oplus Carry[t],$$

where

$$Carry[t] = \begin{cases} s[t-1] & \textit{if} \quad k[t-1] = s[t-1], \\ s[t-2] & \textit{if} \quad k[t-1] \neq s[t-1] \ \textit{and} \ k[t-2] = y[t-2]. \end{cases}$$

A.2 Proof of Property for Modular Subtraction

In the following, we give a complete proof for Lemma 2.

Proof. By considering modular addition $s = x \boxplus k$, the following holds

$$\begin{gathered} s[t+w-1:t] = (x[t+w-1:t] + k[t+w-1:t] + Carry[t]) \ \ mod\ 2^w, \\ Carry[t+w] = \lfloor (x[t+w-1:t] + k[t+w-1:t] + Carry[t])\ /2^w \rfloor \end{gathered}$$

where $Carry[i]$ is i-th bit of carry vector, i.e. $Carry[i] \triangleq (x \boxplus k)[i] \oplus x[i] \oplus k[i]$. That is

$$x[t+w-1:t] + k[t+w-1:t] + Carry[t] = Carry[t+w] \cdot 2^w + s[t+w-1:t],$$

then we have

$$x[t+w-1:t] = Carry[t+w] \cdot 2^w + s[t+w-1:t] - k[t+w-1:t] - Carry[t].$$

Since $0 \leq x[t+w-1:t] < 2^w$, by contradiction, we can easily conclude that

$$Carry[t+w] = \begin{cases} 0 & \textit{if} \quad s[t+w-1:t] > k[t+w-1:t], \\ Carry[t] & \textit{if} \quad s[t+w-1:t] = k[i+w-1:i], \\ 1 & \textit{if} \quad s[t+w-1:t] < k[t+w-1:t]. \end{cases}$$

With representing $s = x \boxplus k$ in a bit level, we have

$$s[t] = x[t] \oplus k[t] \oplus Carry[t]$$

where $Carry[t] = MAJ(Carry[t-1], x[t-1], k[t-1])$, and the majority function $MAJ(a,b,c) = ab \oplus ac \oplus bc$.

Since $s[t-1] = x[t-1] \oplus k[t-1] \oplus Carry[t-1]$, if $k[t-1] \neq s[t-1]$, then we have $x[t-1] \neq Carry[t-1]$, and $Carry[t] = k[t-1] = s[t-1] \oplus 1$. As for the case $k[t-1] = s[t-1]$, we have $x[t-1] = Carry[t-1]$, and $Carry[t] = Carry[t-1]$. Moreover, if $k[t-2] \neq s[t-2]$, then we have $x[t-2] \neq Carry[t-2]$, and $Carry[t] = Carry[t-1] = k[t-2] = s[t-2] \oplus 1$. By analogy, we have $Carry[t] = s[t-\iota] \oplus 1$ where ι is such that $k[t-1:t-\iota+1] \oplus s[t-1:t-\iota+1] = \vec{0}$ *and* $k[t-\iota] \oplus s[t-\iota] = 1$. □

The corollary of Lemma 2 when $w = 1$ and $u = 2$ is presented in the following, which actually covers the result of Lemma 2 in [2].

Corollary 3 ([2]). *Let $t \geq 1$, and let $x, k, s \in \mathbb{F}_2^n$ and $s = x \boxplus k$. Then,*

$$x[t] = Carry[t+1] \cdot 2 + s[t] - k[t] - Carry[t],$$

$$Carry[t+1] = \begin{cases} 0 & if \quad s[t] > k[t], \\ Carry[t] & if \quad s[t] = k[t], \\ 1 & if \quad s[t] < k[t], \end{cases}$$

that is

$$x[t] = s[t] \oplus k[t] \oplus Carry[t],$$

where

$$Carry[t] = \begin{cases} s[t-1] \oplus 1 & if \quad k[t-1] \neq s[t-1], \\ s[t-2] \oplus 1 & if \quad k[t-1] = s[t-1] \text{ and } k[t-2] \neq s[t-2]. \end{cases}$$

B Proofs for Theoretical Analysis of Differential Equations

In this section, we give the proofs of Lemma 3 and Theorem 1, along with two examples of strong keys.

B.1 Proof of Lemma 3

Here, we give the proof of Lemma 3 as the following.

Proof. For modular addition of c and d', the following holds

$$c[21:20] + d'[21:20] + Carry_{(c,d')}[20] = c'[21:20] + Carry_{(c,d')}[22] \times 4.$$

Taking Eq. (8) and(9) into the above formula, we have that

$$\begin{gathered} c[21:20] + d'[21:20] + Carry_{(c,d')}[20] = c'[21:20] + c[22] \times 4, \\ \text{for } c'[21:20] \in C'^{b[21:20]}_{a'[2:0]} \end{gathered} \tag{16}$$

where $C'^{b[21:20]}_{a'[2:0]} \triangleq \{b'[1:0] \oplus b[21:20] \mid a'[2] = \lfloor (a'[1:0] + b'[1:0])/2^2 \rfloor\}$. Therefore, our goal is to find a value of $\sigma = d'[21:20] + Carry_{(c,d')}[20]$ satisfy Eq. (16), which can be controlled by varying branch d.

First, assume that the value of $sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20])$ is not equal to zero for some $c[21:20]$, if Eq. (9) is satisfied, then we have the assertion about the value of $Carry_{(c,d')}[22]$, i.e. $Carry_{(c,d')}[22] = \frac{1}{2}(1 - sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20]))$, where $sign(\cdot,\cdot) : \{\alpha \mid \alpha \in \mathbb{Z}\} \times \mathbb{Z} \to \{1, -1, 0\}$ is defined as

$$sign(S, \beta) = \begin{cases} 1 \; if & \forall\, \alpha \in S, \alpha > \beta, \\ -1 \; if & \forall\, \alpha \in S, \alpha < \beta, \\ 0 \; otherwise. \end{cases}$$

However Eq. (9) requires that $Carry_{(c,d')}[22] = c[22]$. Therefore, when $c[22] = \frac{1}{2}(1 + sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20]))$, Eq. (8) and (9) can not hold simultaneously.

Next, it is discussed that for different cases, whether the above situation happens, and if not, how we set d to satisfy Eq. (16), i.e. satisfying Eqs. (8) and (9) simultaneously.

- For $a'[2:0] = 000$ and $\forall\, b[21:20] \in \{0,1\}^2$, $a'[2:0] \in \{111, 001\}$ and $b[21:20] \in \{01, 10\}$, the value of $sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20])$ is equal to zero for any values of $c[21:20]$.
  If $c[22] = 1$, then Eq. (8) requires $c'[21:20] \leq c[21:20]$, and if $c[21:20] = min(C'^{b[21:20]}_{a'[2:0]})$, then with considering Eq. (9) together, we have that $c'[21:20]$ is only equal to $c[21:20]$. Putting $c'[21:20] = c[21:20]$ into Eq. (16), we have $d'[21:20] + Carry_{(c,d')}[20] = c[22] \times 4$, that is $d'[21:20] = c[22] \times 4 - c[22]$ and $Carry_{(c,d')}[20] = c[22]$. With expanding $Carry_{(c,d')}[20]$, we have $Carry_{(c,d')}[20] = c[19-t]$, if $d'[19-i] = c[19-i] \oplus 1$ for $0 \leq i \leq t-1$ and $d'[19-t] = c[19-t]$, where $0 \leq t \leq 19$. In order to involve as less bits of b and c as possible, we always choose the minimum integer t such that $c[19-t] = c[22]$ where $0 \leq t \leq 19$. Considering the relation $d' = (d \boxplus a') \lll 16$, Eqs. (8) and (9) are satisfied simultaneously with the setting of Eqs. (13). Similarly, the setting also applies in the case where $c[22] = 0$ and $c[21:20] = max(C'^{b[21:20]}_{a'[2:0]})$.
  Otherwise, if $c[22] = 1$, then we choose $c'[21:20] = c[21:20] - 1 < c[21:20]$; if $c[22] = 0$, then we choose $c'[21:20] = c[21:20] + 1 > c[21:20]$. Putting this setting of $c'[21:20]$ into Eq. (16), we have $d'[21:20] + Carry_{(c,d')}[20] = 2 \times c[22] + 1$. Note that there is no restriction for the value of $Carry_{(c,d')}[20]$ any more with this setting of $c'[21:20]$. Then with setting $d'[19] = c[19]$, and we have $Carry_{(c,d')}[20] = c[19]$. As a result, Eqs. (8) and (9) are satisfied simultaneously with the setting of Eqs. (14).
- For $a'[2:0] = 100$, it is easily derived that Eq (9) must not be satisfied.
- For $a'[2:0] \in \{101, 110, 010, 011\}$ and $\forall\, b[21:20] \in \{0,1\}^2$, $a'[2:0] \in \{111, 001\}$ and $b[21:20] \in \{00, 11\}$,
  - When $c[21:20] \notin C'^{b[21:20]}_{a'[2:0]}$, we have that $sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20])$ is not equal to zero, and if $c[22] = \frac{1}{2}(1 + sign(C'^{b[21:20]}_{a'[2:0]}, c[21:20]))$, then Eqs. (8) and (9) can not hold simultaneously; otherwise, we can choose $c'[21:20]$ into any value of $C'^{b[21:20]}_{a'[2:0]}$, especially $min(C'^{b[21:20]}_{a'[2:0]})$, and Eqs. (8) and (9) can hold simultaneously with the setting of Eqs. (15).
  - When $c[21:20] \in C'^{b[21:20]}_{a'[2:0]}$, we can discuss like done in the first item. Eqs. (8) and (9) can hold simultaneously, specially with the following setting. If $c[21:20] = min(C'^{b[21:20]}_{a'[2:0]})$ and $c[22] = 1$, or $c[21:20] = max(C'^{b[21:20]}_{a'[2:0]})$ and $c[22] = 0$, setting is same to Eqs. (13); otherwise, setting is same to Eqs. (14).

□

B.2 Proof of Theorem 1

The proof of Theorem 1 is presented along with two examples of strong keys.

Proof. According to Lemma 3, for $c'[2:0] \in \{101, 110, 010, 011\}$ and $\forall\, b[21:20] \in \{0,1\}^2$, $c'[2:0] \in \{111, 001\}$ and $b[21:20] \in \{00, 11\}$, if $c[21:20] \in \{0,1\}^2 \setminus C'^{b[21:20]}_{a'[2:0]}$ where $C'^{b[21:20]}_{a'[2:0]} \triangleq \{b'[1:0] \oplus b[21:20] \mid a'[2] = \lfloor (a'[1:0] + b'[1:0])/2^2 \rfloor\}$, then there always exists a value of $c[22]$ such that a key become the strong key. Therefore, we have $\tau = \frac{1}{2} \times \frac{4 - |C'^{b[21:20]}_{a'[2:0]}|}{4}$, where τ denote the percentage of strong keys derived by Lemma 3. More precisely, we have

$$\tau = \begin{cases} 0.0625 \ \textit{if} \ \ a'[2:0] = 001, & 0.375 \ \ \textit{if} \ \ a'[2:0] = 101, \\ 0.25 \ \ \textit{if} \ \ a'[2:0] = 010, & 0.25 \ \ \textit{if} \ \ a'[2:0] = 110, \\ 0.375 \ \ \textit{if} \ \ a'[2:0] = 011, & 0.0625 \ \textit{if} \ \ a'[2:0] = 111. \end{cases}$$

Besides, Eq. (9) must not be satisfied when $a'[2:0] = 100$. Totally, there are at least 29.6875% strong keys for all possibilities of $a'[2:0]$. □

Examples of strong keys. Here we show two specific examples derived by Lemma 3.

- For the case of key such that $a'[2:0] = 110$, $b[21:20] = 00$, when Eq. (9) holds, we have that $c'[21:20] \in \{10, 11\}$. If $c[21:20] \in \{00, 01\}$, then $Carry_{(c,d')}[22] = 0$. If $c[22] = 1$, Eq. (8) must not be satisfied and the corresponding key become a strong key.
- For the case of key such that $a'[2:0] = 010$, $b[21:20] = 00$, when Eq. (9) holds, we have that $c'[21:20] \in \{00, 01\}$. If $c[21:20] \in \{10, 11\}$, then $Carry_{(c,d')}[22] = 1$. If $c[22] = 0$, Eq. (8) must not be satisfied and the corresponding key become a strong key.

C Applications of Syncopation Technique

C.1 Application on ChaCha7

Preliminary shortlisting of 151 possible CPNBs. For ChaCha7, we list the preliminary shortlisting of 151 possible CPNBs whose conditional neutrality measure is greater than 2^{-5}, $\{2, 3, 4, 5, 6, 7, 10, 11, 14, 15, 16, 19, 22, 26, 27, 31, 33, 35, 39, 40, 41, 43, 47, 48, 49, 51, 52, 53, 59, 60, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 90, 91, 92, 93, 95, 96, 97, 99, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 115, 116, 117, 119, 120, 123, 124, 125, 126, 127, 128, 131, 132, 135, 136, 137, 140, 143, 147, 148, 152, 155, 156, 157, 158, 159, 160, 161, 162, 164, 165, 168, 169, 170, 174, 180, 181, 186, 188, 191, 192, 193, 194, 195, 196, 199, 200, 201, 202, 204, 205, 207, 208, 211, 212, 213, 214, 216, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 232, 233, 237, 240, 244, 245, 246, 247, 249, 250, 252, 255\}$.

C.2 Applications on ChaCha7.5$^{\oplus}$

Preliminary shortlisting of 119 possible CPNBs. For ChaCha7.5$^{\oplus}$, we list the preliminary shortlisting of 119 possible CPNBs whose conditional neutrality measure is greater than 2^{-5}, $\{2, 4, 22, 31, 40, 48, 49, 51, 59, 60, 64, 66, 67, 68, 72, 74, 75, 77, 78, 79, 80, 81, 83, 84, 85, 86, 90, 91, 92, 93, 95, 96, 99, 104, 105, 106, 108, 109, 110, 111, 112, 115, 116, 117, 120, 123, 124, 125, 126, 127, 128, 131, 135, 136, 137, 140, 143, 147, 148, 152, 155, 156, 157, 158, 159, 160, 161, 162, 164, 165, 168, 169, 170, 174, 180, 181, 186, 188, 191, 192, 193, 194, 195, 196, 199, 200, 201, 202, 204, 205, 207, 208, 211, 212, 213, 214, 216, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 232, 233, 237, 240, 244, 245, 246, 247, 249, 250, 252, 255\}$.

Syncopated segments of 54 CPNBs. The profitable syncopated segments for the 54 CPNBs are summarized in Table 6 with 30 conditions, where the two removed conditions are marked by red in both columns of table.

Table 6. Syncopated segments of 54 CPNBs.

syncopated segment $\leftarrow$ condition	(i, j_1, j_2)
$x_6^{(7.5^{\oplus})}[j_2 : j_1] \leftarrow k_2[i] \neq z_6^{(7.5^{\oplus})}[i]$ $x_1^{(7)}[j_2 : j_1] \leftarrow k_2[i] = (x_1^{(7.5^{\oplus})} \boxminus z_6^{(7.5^{\oplus})})[i]$	$\{(12, 13, 18), (20, 21, 25), (29, 30, 30)\}$
$x_7^{(7.5^{\oplus})}[j_2 : j_1] \leftarrow k_3[i] \neq z_7^{(7.5^{\oplus})}[i]$ $x_2^{(7)}[j_2 : j_1] \leftarrow k_3[i] = (x_2^{(7.5^{\oplus})} \boxminus z_7^{(7.5^{\oplus})})[i]$	$\{(16, 17, 18), (20, 21, 26)\}$
$x_9^{(7)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7.5^{\oplus})} \boxminus x_{14}^{(7.5^{\oplus})})[i]$ $x_9^{(6.5)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7.5^{\oplus})} \boxminus x_{14}^{(7.5^{\oplus})} \boxminus x_{13}^{(7)})[i]$	$(11, 12, 30)$
$x_{10}^{(7)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7.5^{\oplus})} \boxminus x_{15}^{(7.5^{\oplus})})[i]$ $x_{10}^{(6.5)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7.5^{\oplus})} \boxminus x_{15}^{(7.5^{\oplus})} \boxminus x_{14}^{(7)})[i]$	$\{(4, 5, 6), (9, 10, 11), (13, 14, 14), (17, 18, 18), (22, 23, 23), (25, 26, 26)\}$
$x_{11}^{(7)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7.5^{\oplus})} \boxminus x_{12}^{(7.5^{\oplus})})[i]$ $x_{11}^{(6.5)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7.5^{\oplus})} \boxminus x_{12}^{(7.5^{\oplus})} \boxminus x_{15}^{(7)})[i]$	$\{(4, 5, 7), (9, 10, 19), (24, 25, 30)\}$

Syncopated segments of 48 CPNBs in another attack. The profitable syncopated segments for the 48 CPNBs are summarized in Table 7 with 30 conditions.

C.3 Application on ChaCha6

First the inverse structure of ChaCha6 is analyzed, which is used to determine all the syncopated segments of internal state in Algorithm 1. The last round of ChaCha6 is illustrated in Fig. 8. At the end of this section, we list the significant key bits, corresponding CPNB's syncopated segments and conditions.

Table 7. Syncopated segments of 48 CPNBs.

syncopated segment $\leftarrow$ condition	(i, j_1, j_2)
$x_6^{(7.5^{\oplus})}[j_2 : j_1] \leftarrow k_2[i] \neq z_6^{(7.5^{\oplus})}[i]$ $x_1^{(7)}[j_2 : j_1] \leftarrow k_2[i] = (x_1^{(7.5^{\oplus})} \boxminus z_6^{(7.5^{\oplus})})[i]$	$\{(12, 13, 18), (20, 21, 25), (28, 29, 30)\}$
$x_7^{(7.5^{\oplus})}[j_2 : j_1] \leftarrow k_3[i] \neq z_7^{(7.5^{\oplus})}[i]$ $x_2^{(7)}[j_2 : j_1] \leftarrow k_3[i] = (x_2^{(7.5^{\oplus})} \boxminus z_7^{(7.5^{\oplus})})[i]$	$\{(16, 17, 18), (20, 21, 26)\}$
$x_9^{(7)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7.5^{\oplus})} \boxminus x_{14}^{(7.5^{\oplus})})[i]$ $x_9^{(6.5)}[j_2 : j_1] \leftarrow k_5[i] \neq (z_9^{(7.5^{\oplus})} \boxminus x_{14}^{(7.5^{\oplus})} \boxminus x_{13}^{(7)})[i]$	$(10, 11, 30)$
$x_{10}^{(7)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7.5^{\oplus})} \boxminus x_{15}^{(7.5^{\oplus})})[i]$ $x_{10}^{(6.5)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(7.5^{\oplus})} \boxminus x_{15}^{(7.5^{\oplus})} \boxminus x_{14}^{(7)})[i]$	$\{(3, 4, 6), (8, 9, 11), (13, 14, 14),$ $(17, 18, 18), (21, 22, 23), (25, 26, 26)\}$
$x_{11}^{(7)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7.5^{\oplus})} \boxminus x_{12}^{(7.5^{\oplus})})[i]$ $x_{11}^{(6.5)}[j_2 : j_1] \leftarrow k_7[i] \neq (z_{11}^{(7.5^{\oplus})} \boxminus x_{12}^{(7.5^{\oplus})} \boxminus x_{15}^{(7)})[i]$	$\{(3, 4, 7), (9, 10, 19), (24, 25, 30)\}$

For the inversion of the first *quarterround* function, since the following relationships,

$$\begin{cases} x_5^{(6)} = z_5^{(6)} \boxminus k_1 \\ x_{10}^{(6)} = z_{10}^{(6)} \boxminus k_6 \\ y_{10}^{(5)} = (z_{10}^{(6)} \boxminus \underset{\cdots}{x_{15}^{(6)}}) \boxminus k_6 \\ x_{10}^{(5)} = (z_{10}^{(6)} \boxminus \underset{\cdots}{x_{15}^{(6)}} \boxminus \underset{\cdots}{y_{15}^{(5)}}) \boxminus k_6 \end{cases}, \quad \begin{cases} \underset{\cdots}{x_0^{(6)}} = z_0^{(6)} \boxminus c_0 \\ \underset{\cdots}{x_{15}^{(6)}} = z_{15}^{(6)} \boxminus n_2 \\ \underset{\cdots}{y_{15}^{(5)}} = (\underset{\cdots}{x_{15}^{(6)}} \ggg 8) \oplus \underset{\cdots}{x_0^{(6)}} \end{cases},$$

the syncopation technique can be applied on words $x_5^{(6)}$, $x_{10}^{(6)}$, $y_{10}^{(5)}$, and $x_{10}^{(5)}$ with using the property of modular subtraction (Lemma 2). The analysis is similar for the other three *quarterround* functions, and we mark all these words by red in Fig. 8 which the syncopation technique can be applied on. Given a (C)PNB set, Algorithm 1 is performed, and for the words marked by red in Fig. 8, the syncopated segments of internal state and corresponding conditions on restricted bits are determined according to the syncopated segments and restricted segments of secret key. The parameter $u = 2$ and $u = 1$ are used, that is, first determining the syncopated segments with a two-bit restricted segment, and then ones with a one-bit restricted segment.

The significant key bits and syncopated segments. Here we list the 45 significant key bits i.e. non-CPNBs, which are used in the attack on ChaCha6, $\{6, 37, 38, 39, 40, 41, 42, 43, 44, 45, 50, 68, 69, 70, 75, 76, 77, 78, 89, 102, 108, 114, 133, 139, 153, 192, 193, 194, 195, 196, 197, 198, 203, 224, 225, 226, 227, 228, 229, 230, 231, 241, 242, 253, 254\}$. The profitable syncopated segments for corresponding 211 CPNBs are summarized in Table 8 along with 27 conditions ($\theta_1 = 18$, $\theta_2 = 9$).

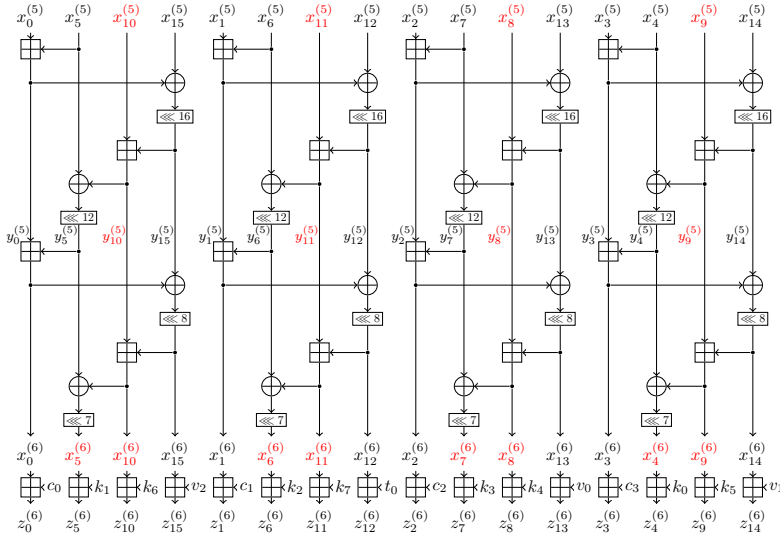


Fig. 8. Last round of ChaCha6.

Table 8. Syncopated segments of 211 CPNBs, where "-" denotes $w = 0$.

syncopated segment $\leftarrow$ condition	(i, j_1, j_2)
$x_4^{(6)}[j_2 : j_1] \leftarrow k_0[i] \neq z_4^{(6)}[i]$	$(6, \text{-}, \text{-})$
$x_5^{(6)}[j_2 : j_1] \leftarrow k_1[i : i-1] \oplus z_5^{(6)}[i : i-1] \neq 00$	$(6, 7, 13)$
$x_5^{(6)}[j_2 : j_1] \leftarrow k_1[i] \neq z_5^{(6)}[i]$	$(18, \text{-}, \text{-})$
$x_6^{(6)}[j_2 : j_1] \leftarrow k_2[i : i-1] \oplus z_6^{(6)}[i : i-1] \neq 00$	$\{(5, 6, 6), (12, 13, 14)\}$
$x_6^{(6)}[j_2 : j_1] \leftarrow k_2[i] \neq z_6^{(6)}[i]$	$(25, \text{-}, \text{-})$
$x_7^{(6)}[j_2 : j_1] \leftarrow k_3[i] \neq z_7^{(6)}[i]$	$\{(6, \text{-}, \text{-}), (12, \text{-}, \text{-}), (18, \text{-}, \text{-})\}$
$x_8^{(6)}[j_2 : j_1] \leftarrow k_4[i] \neq z_8^{(6)}[i]$ $y_8^{(5)}[j_2 : j_1] \leftarrow k_4[i] \neq (z_8^{(6)} \boxminus x_{13}^{(6)})[i]$ $x_8^{(5)}[j_2 : j_1] \leftarrow k_4[i] \neq (z_8^{(6)} \boxminus x_{13}^{(6)} \boxminus y_{13}^{(5)})[i]$	$\{(5, \text{-}, \text{-}), (11, \text{-}, \text{-}), (25, \text{-}, \text{-})\}$
$x_{10}^{(6)}[j_2 : j_1] \leftarrow k_6[i] \neq z_{10}^{(6)}[i]$ $y_{10}^{(5)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(6)} \boxminus x_{15}^{(6)})[i]$ $x_{10}^{(5)}[j_2 : j_1] \leftarrow k_6[i] \neq (z_{10}^{(6)} \boxminus x_{15}^{(6)} \boxminus y_{15}^{(5)})[i]$	$(11, \text{-}, \text{-})$
$x_{11}^{(6)}[j_2 : j_1] \leftarrow k_7[i : i-1] \oplus z_{11}^{(6)}[i : i-1] \neq 00$ $y_{11}^{(5)}[j_2 : j_1] \leftarrow k_7[i : i-1] \oplus (z_{11}^{(6)} \boxminus x_{12}^{(6)})[i : i-1] \neq 00$ $x_{11}^{(5)}[j_2 : j_1] \leftarrow k_7[i : i-1] \oplus (z_{11}^{(6)} \boxminus x_{12}^{(6)} \boxminus y_{12}^{(5)})[i : i-1] \neq 00$	$\{(18, \text{-}, \text{-}), (30, \text{-}, \text{-})\}$

C.4 Experiments of Practical Attacks

To demonstrate the syncopation technique, two practical key-recovery attack have been implemented on 5 rounds and 6 rounds of ChaCha with 64 unknown key bits. Specifically, in order to make the process of key recovery practical, it is assumed that under the condition that 192 out of 256 key bits have been obtained and known, the attack is performed to recover some of the remaining 64 bits of secret key.

Next, we summarize the main results of experiments on 5-round and 6-round ChaCha with 64 unknown key bits.

Environment of experiments. We have implemented the procedure of new (partial) key-recovery attack with the syncopation technique through C++ programming, that is, Step 1 to 5 of Algorithm 3. All the experiments are conducted on Linux version 6.2.9-arch1-1 with Intel Core i9-13900KF and RAM of 128 GB.

Attack on 5-round ChaCha For 5-round ChaCha, the attack uses a forward differential with 2.5 rounds, and approximates 2.5 rounds in backward direction with PNBs of 64 unknown key bits. More precisely, the median correlation of forward differential is $|\varepsilon_d| = 0.838$ with input difference at $\Delta x_{13}^{(0)}[6]$ and output difference at $\Delta x_2^{(2.5)}[0]$. The last round of 5-round ChaCha is illustrated in Fig. 9, where k_i ($i \geq 2$) are assumed to be known in backward approximation, and 56 PNBs of k_0 and k_1 are found. Accordingly, the 8 non-PNBs are $\{k_0[i],\, i \in \{6, 9, 10, 11, 12, 13, 18\}\}$ and $k_1[6]$. With the syncopation (sync. for short) technique, the conditional backward correlation is experimentally estimated as $|\varepsilon_a| = 0.752$ under the condition that the $\theta_1 = 4$ constraints $k_0[i] \neq z_4^{(5)}[i]$, $i \in \{6, 9, 18\}$ and $k_1[6] \neq z_5^{(5)}[6]$ are satisfied. However, without the syncopation technique, the median correlation of backward approximation is experimentally estimated as $|\varepsilon_a| = 2^{-7.6}$ by fixing 56 PNBs into zeros. Therefore, the backward correlation is significantly amplified by utilizing the syncopation technique.

To recover 8 non-PNBs of 64-bit secret key with the syncopation technique, we only need to implement Step 1 to 5 of Algorithm 3. According to the complexity analysis, the time complexity of key-recovery attack is $T = N + 2^8 \times N^*$, data complexity is $N = 2^{2\times 4} \times N^*$ and memory complexity is $M = 2^4 \times N^*$. The Neyman-Pearson decision theory gives the results about estimating the number of samples N^* required to get the bounds on probabilities of false alarm p_{fa} and non-detection p_{nd}. It can be shown that $N^* \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-(\varepsilon_a \varepsilon_d)^2}}{\varepsilon_a \varepsilon_d} \right)^2$ samples suffices to achieve $p_{nd} = 1.3 \times 10^{-3}$ and $p_{fa} = 2^{-\alpha}$. With $\alpha = 8$, the time complexity is $T = 2^{15.4}$, data complexity is $N = 2^{14.4}$ and memory complexity is $M = 2^{10.4}$. As pointed out in [1], with using median correlation in the above equation, we have a success probability of at least $\frac{1}{2}(1 - p_{nd}) \approx \frac{1}{2}$ for the attack.

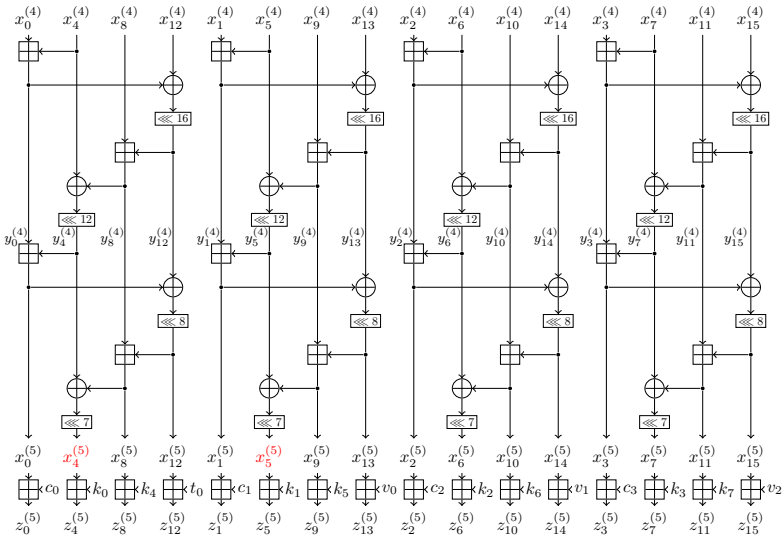


Fig. 9. Last round of 5-round ChaCha.

Result of experiments of 5-round ChaCha. It takes about 22 seconds to run the C++ program 10000 times. As a result, for 10000 randomly generated key, 7071 of them are successfully recovered 8 guessed key bits with the syncopation techniques, and thus the success probability is about 70.7%. The attacks on 5-round ChaCha are summarized in Table 9.

Table 9. Summary of attacks on 5-round ChaCha with 64 unknown key bits.

Attack method	Analysis	$\lvert\varepsilon_a\rvert$	T	N	#Random keys	Success probability
With sync.	Theoretical	-	$2^{15.4}$	$2^{14.4}$	-	≥ 0.5
	Experimental	0.752	2.2×2^{-3} s	$2^{14.4}$	10000	$\frac{7071}{10000} \approx 70.7\%$
Without sync.	Theoretical	-	$2^{29.1}$	$2^{21.1}$	-	≥ 0.5
	Experimental	$2^{-7.6}$	22.5 s	$2^{21.1}$	100	$\frac{16}{100} \approx 16\%$

Attack on 6-round ChaCha With regard to 6-round ChaCha, the attack is similar to the one of 5-round ChaCha, except that a forward differential with 3.5 rounds is used, and thus backward approximation consists of 2.5 rounds from the 6-th to 3.5-th round. Specifically, the forward differential takes $\Delta x_{13}^{(0)}[6]$ as the input difference and observes the output difference at $\Delta x_2^{(3.5)}[0]$. As shown in [2], the correlation of the forward characteristic is evaluated as $2^{-8.3}$ under that condition that the output difference after the first round has the minimum

Hamming weight 10, which takes about 2^5 iteration to achieve a right pair for the differential.

The 8 non-PNBs are $k_0[6]$ and $\{k_1[i],\, i \in \{6, 9, 10, 11, 12, 13, 18\}\}$. The conditional backward correlation is experimentally estimated as $|\varepsilon_a| = 0.759$ under the condition that the $\theta_1 = 4$ constraints $k_0[6] \neq z_4^{(6)}[6]$ and $k_1[i] \neq z_5^{(6)}[i]$, $i \in \{6, 9, 18\}$. According to the complexity analysis, the time complexity of key-recovery attack is $T = 2^5 \times (N + 2^8 \times N^*)$, data complexity is $N = 2^{5+2\times 4} \times N^*$ and memory complexity is $M = 2^4 \times N^*$, and with $\alpha = 13$, $T = 2^{37.1}$, $N = 2^{36.1}$ and $M = 2^{27.1}$.

Result of experiments of 6-round ChaCha. It takes about 1.6 hours to run once the C++ program with RAM about 24 GB for the key-recovery attack of 6-round ChaCha. As a result, for 16 randomly generated key, 10 of them are successfully recovered 8 guessed key bits with the syncopation techniques, and thus the success probability among all keys is about 62.5%. In another experiment with 32 randomly generated key, there are 21 weak keys among which 15 keys are successfully recovered 8 guessed key bits with the syncopation techniques, and thus the success probability of key-recovery attack in Algorithm 3 is about $\frac{15}{21} \approx 71.4\%$.