Improved Cryptanalysis of ChaCha: Beating PNBs with Bit Puncturing

Antonio Flórez-Gutiérrez[0000−0001−7749−8925] and
Yosuke Todo[0000−0002−6839−4777]

NTT Social Informatics Laboratories, Japan
antonio.florez@ntt.com, yosuke.todo@ntt.com

Abstract. ChaCha is a widely deployed stream cipher and one of the most important symmetric primitives. Due to this practical importance, many cryptanalysis have been proposed. Until now, Probabilistic Neutral Bits (PNBs) have been the most successful. Given differential-linear distinguishers, PNBs are the technique for key recovery relying on an experimental backward correlation obtained through blackbox analysis. A careful theoretical analysis exploiting the round function design may find a better attack and improve our understanding, but the complicated nature of the ARX structure makes such analysis difficult. We propose a theoretical methodology inspired by bit puncturing, which was recently proposed at Eurocrypt 2024. Our method has a theoretical foundation and is thus fundamentally different from PNBs, to which it is the first effective alternative. As a result, we significantly improved the attack complexity for 6, 7, and 7.5-round ChaCha. The 7-round attack is about 2^{40} times faster than the previous best. Furthermore, we propose the first 7.5-round attack with a non-negligible advantage over an exhaustive search.

Keywords: ChaCha, Walsh spectrum puncturing, Differential-linear attack

1 Introduction

Symmetric key primitives play a fundamental role in most if not all practical security applications: due to their performance, they are used to encrypt most sensitive data. Block ciphers are the most common such primitive, with AES being the most famous. Another common primitive is the stream cipher, which has throughput advantages over block ciphers. ChaCha [Ber08] is currently one of the most popular stream ciphers, and has been deployed in several operating systems and TLS. Reflecting its practical importance, the security of ChaCha has been widely discussed since its introduction and until today [AFK+08,CM16,BLT20,DGSS22,DGSS23,WLHL23,BGG+23,Dey24,XXTQ24].

ChaCha belongs to a design principle called Add-Rotate-XOR (ARX). As the name implies, the main operations in an ARX cipher are bitwise XORs, bit rotations, and additions modulo 2^n, which are in most processors' instruction sets. Targeting software efficiency on several processors of all performance

categories, many modern ARX ciphers adopt 32-bit registers. Multiple 32-bit registers represent the state, which are mixed with ARX operations.

PNBs. Among the cryptanalysis of ChaCha (see Table 1), the most actively discussed technique is Probabilistic Neutral Bits (PNBs), introduced by Aumasson et al [AFK+08]. They are deployed in a variant of differential-linear cryptanalysis. We divide the cipher E into two parts, $E_2 \circ E_1$ and set a differential-linear distinguisher for E_1 with input difference Δ and output linear mask Γ. Usually, key recovery requires guessing all key bits which influence $\langle \Gamma, E_2^{-1} \rangle$. Unfortunately, this is problematic in ARX ciphers because almost all key bits are activated after a few operations. The idea behind PNBs is that not all the key bits are equally important, so we can separate the key guess into significant and non-significant bits. Non-significant bits are removed from the guess and set to a fixed value, e.g., all zero. Such a key recovery map is no longer accurate but highly correlated to the original one. As a result, we have an *approximated key-recovery map*. The data complexity increases in order to compensate the smaller correlation, but the time complexity decreases due to the reduced key guess.

Although several new differential-linear distinguishers have been proposed [BLT20,CN20,BGG+23], the key recovery relies on PNBs in almost all existing works. Most recent cryptanalysis of ChaCha uses enhanced PNBs, e.g., syncopation technique [WLHL23], proper assignment of non-significant bits [DGSS23], or linear decomposition [Dey24]. However, no alternatives to PNBs have been proposed since the seminal 2008 work by Aumasson et al., particularly for attacks on 7 or more rounds.

Walsh Spectrum Puncturing. Recently, Flórez-Gutiérrez and Todo introduced *Walsh spectrum puncturing* for linear key recovery attacks [FT24]. They focus on the Walsh spectrum of the key recovery function instead of the function itself. By setting some coefficients of the Walsh spectrum to zero, a new (real-valued) highly-correlated key recovery map can be constructed. This correlation can be deduced from the punctured coefficients.

Our Contribution. So far, only PNBs and its variants have been applied to 7-round attacks on ChaCha. PNBs evaluate the correlation after removing the non-significant bits experimentally, so careful analysis of the key recovery function is not as crucial as with other attacks, which is helpful because ARX makes the key recovery map quite complicated. On the other hand, we expect that careful theoretical analysis may find better attacks and enhance our understanding.

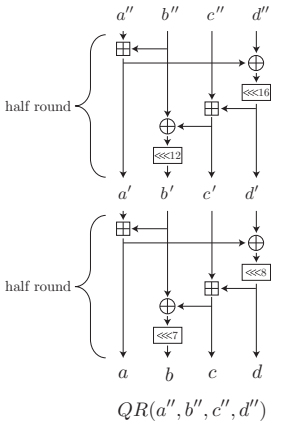
We focus on Walsh spectrum puncturing. Similar to PNBs, it aims to construct an approximated key-recovery map. In particular, one specific variant is *bit puncturing*, in which some input bits of the key recovery map are ignored by puncturing all the Walsh coefficients involving them. The resulting (real-valued) pseudoboolean function returns the average over all choices of the punctured bits. Unlike with PNBs, where a likely value (either 0 or 1) is returned, bit puncturing returns a real number. In fact, bit puncturing is the optimal (maximum correlation) approximation of the key recovery function.

Table 1: Comparison of attacks against reduced-round ChaCha.

Round	Data	Time	Notes	Ref.
6	2^{30}	2^{139}	Introduction of PNBs	[AFK+08]
	$2^{37.5}$	$2^{127.5}$	Linear-trail extension of DL, PNBs	[CM16]
	2^{58}	$2^{77.4}$	First 3.5-round distinguisher, partitioning	[BLT20,BBC+22]
	$2^{73.7}$	$2^{75.7}$	PNBs with syncopation	[WLHL23]
	$2^{41.6}$	$2^{71.0}$	PNBs with linear decomposition	[Dey24]
	$2^{51.0}$	$2^{61.4}$	Puncturing with distillation	Sect.5
	$2^{55.7}$	$2^{57.4}$	Puncturing with distillation	Sect.5
7	2^{27}	2^{248}	Introduction of PNBs	[AFK+08]
	2^{96}	$2^{237.7}$	Linear-trail extension of DL, PNBs	[CM16]
	$2^{48.8}$	$2^{230.9}$	3.5-round distinguisher with 1-bit mask, PNBs	[BLT20,BBC+22]
	$2^{90.2}$	$2^{222.0}$	PNBs using memory/non-memory subspace	[DGSS22]
	$2^{87.2}$	$2^{218.9}$	Improved version of [DGSS22]	[DGSS23]
	$2^{68.9}$	$2^{216.9}$	PNBs with syncopation	[WLHL23]
	$2^{103.3}$	$2^{210.3}$	PNBs with syncopation	[WLHL23]
	$2^{110.8}$	$2^{206.8}$	New distinguisher, PNBs	[BGG+23]
	$2^{100.8}$	$2^{192.9}$	PNBs with linear decomposition	[Dey24]
	$2^{102.6}$	$2^{189.7}$	DL hull of the [BGG+23] distinguisher, PNBs	[XXTQ24]
	$2^{127.7}$	$2^{148.2}$	Puncturing with distillation	Sect.6
	$2^{102.9}$	$2^{154.2}$	Puncturing with distillation	Sect.6
7.5	$2^{32.6}$	$2^{255.2}$	PNBs with linear decomposition	[Dey24]
	$2^{127.1}$	$2^{250.2}$	Puncturing with distillation	Sect.7

In theory, puncturing can be a promising alternative to PNBs. However, applying it to ARX is not trivial in practice. As almost all bits are quickly activated, the Walsh spectrum has huge dimensions, and computing it is infeasible in practice. To solve this problem, we use a *trail enumeration* technique, which recovers some approximated spectrum coefficients in practical time. We experimentally verified the correctness of the complexity estimation when using punctured Walsh spectra computed through trail enumeration.

Table 1 summarizes cryptanalysis of ChaCha. The time complexity of the attack is significantly improved. For 6 rounds, the attack complexity is now below 2^{60}. For 7 rounds, the time complexity is reduced by about 40 bits, which is a significant jump compared to recent improvements. Dey recently proposed the first 7.5-round attack [Dey24], but its time complexity is almost equal to the exhaustive search. We propose an attack with complexity below 2^{250}.



$$(v_0^{r+1}, v_4^{r+1}, v_8^{r+1}, v_{12}^{r+1}) = QR(v_0^r, v_4^r, v_8^r, v_{12}^r)$$
$$(v_1^{r+1}, v_5^{r+1}, v_9^{r+1}, v_{13}^{r+1}) = QR(v_1^r, v_5^r, v_9^r, v_{13}^r)$$
$$(v_2^{r+1}, v_6^{r+1}, v_{10}^{r+1}, v_{14}^{r+1}) = QR(v_2^r, v_6^r, v_{10}^r, v_{14}^r)$$
$$(v_3^{r+1}, v_7^{r+1}, v_{11}^{r+1}, v_{15}^{r+1}) = QR(v_3^r, v_7^r, v_{11}^r, v_{15}^r)$$

$$(v_0^{r+2}, v_5^{r+2}, v_{10}^{r+2}, v_{15}^{r+2}) = QR(v_0^{r+1}, v_5^{r+1}, v_{10}^{r+1}, v_{15}^{r+1})$$
$$(v_1^{r+2}, v_6^{r+2}, v_{11}^{r+2}, v_{12}^{r+2}) = QR(v_1^{r+1}, v_6^{r+1}, v_{11}^{r+1}, v_{12}^{r+1})$$
$$(v_2^{r+2}, v_7^{r+2}, v_8^{r+2}, v_{13}^{r+2}) = QR(v_2^{r+1}, v_7^{r+1}, v_8^{r+1}, v_{13}^{r+1})$$
$$(v_3^{r+2}, v_4^{r+2}, v_9^{r+2}, v_{14}^{r+2}) = QR(v_3^{r+1}, v_4^{r+1}, v_9^{r+1}, v_{14}^{r+1})$$

Fig. 1: The odd and even round functions of ChaCha.

2 Preliminaries

2.1 ChaCha

The internal state of ChaCha is represented as a 4×4 matrix whose entries are 32-bit vectors. The input state for the rth round function is denoted as

$$V^r = \begin{pmatrix} v_0^r & v_1^r & v_2^r & v_3^r \\ v_4^r & v_5^r & v_6^r & v_7^r \\ v_8^r & v_9^r & v_{10}^r & v_{11}^r \\ v_{12}^r & v_{13}^r & v_{14}^r & v_{15}^r \end{pmatrix}.$$

In odd and even-numbered rounds, the quarter-round QR function is applied on every column and diagonal, respectively. The term *branch* is used for the QR inputs a, b, c and d. A *half round* can also be applied to the state as depicted in Fig. 1. Let $v^{r.5}$ be the internal state after applying a half round on v^r.

The initial state V^0 is initialised with four predefined constants $c_0, \ldots, c_3$, the 256-bit key $k_0, \ldots, k_7$, a 32-bit block counter and a 96-bit nonce $t_0, \ldots, t_4$. Note that we do not distinguish between the counter and the nonce in this paper.

$$V^0 = \begin{pmatrix} v_0^0 & v_1^0 & v_2^0 & v_3^0 \\ v_4^0 & v_5^0 & v_6^0 & v_7^0 \\ v_8^0 & v_9^0 & v_{10}^0 & v_{11}^0 \\ v_{12}^0 & v_{13}^0 & v_{14}^0 & v_{15}^0 \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & t_1 & t_2 & t_3 \end{pmatrix}.$$

The output keystream block Z is $Z = V^0 + V^r$, where "+" denotes wordwise modular addition, i.e., $z_i = v_i^0 \boxplus v_i^r$ for all $i = 0$ to $i = 15$. Note that we can immediately obtain v_i^r for $i \in \{0, 1, 2, 3, 12, 13, 14, 15\}$. For simplicity, we introduce $\bar{Z}$, where $\bar{z}_i = v_i^r$ if $i \in \{0, 1, 2, 3, 12, 13, 14, 15\}$ and $\bar{z}_i = z_i$ otherwise.

In differential-linear attacks, we can only introduce input differences to the 128-bit value $t_0, \ldots, t_3$. We use the following notation to represent a difference.

$$\Delta^r = (\Delta^r_{i_1}[j_1], \Delta^r_{i_2}[j_2], \ldots, \Delta^r_{i_m}[j_m]),$$

which consists of the indicated m bits and thus has Hamming weight m. We write $(\Delta^r_{i_1}[j_1], \ldots, \Delta^r_{i_m}[j_m])$ as $(\Delta^r_{i_1}[j_1, \ldots, j_m])$ in short when $i_1 = i_2 = \cdots = i_m$. Similarly, we use the following notation to represent a linear mask.

$$\Gamma^r = (\Gamma^r_{i_1}[j_1], \Gamma^r_{i_2}[j_2], \ldots, \Gamma^r_{i_m}[j_m]),$$

which has Hamming weight m and $\langle \Gamma^r, V^r \rangle = v^r_{i_1}[j_1] \oplus v^r_{i_2}[j_2] \oplus \cdots \oplus v^r_{i_m}[j_m]$. Similarly to a difference, we write shortly when there are common active branches.

We use an equivalent representation for the convenience of discussing the key recovery. As mentioned above, an attacker can compute $\bar{Z}$. Let $(\bar{z}_a, \bar{z}_b, \bar{z}_c, \bar{z}_d)$ be four branches of $\bar{Z}$ corresponding to each QR function appliaction. Let k_b and k_c is secret key for $\bar{z}_b$ and $\bar{z}_c$, respectively. Then, we have

$$\begin{aligned}
c' &= (\bar{z}_c - k_c) - \bar{z}_d = (\bar{z}_c - \bar{z}_d) - k_c = \bar{z}'_c - k_c, \\
c'' &= (\bar{z}_c - k_c) - (\bar{z}_a \oplus (\bar{z}_d \ggg 8)) = (\bar{z}_c - (\bar{z}_a \oplus (\bar{z}_d \ggg 8)) - k_c = \bar{z}''_c - k_c, \\
d' &= \bar{z}'_d = \bar{z}_a \oplus (\bar{z}_d \ggg 8).
\end{aligned}$$

The attacker can compute $\bar{z}'_c$, $\bar{z}''_c$, and $\bar{z}'_d$. Besides, c' and c'' are evaluated by subtracting k_c from $\bar{z}'_c$ and $\bar{z}''_c$, respectively. We use the notation, $\bar{z}'_i$ and $\bar{z}''_i$, when we compute them without guessing the key.

2.2 Differential-linear Cryptanalysis

Differential-linear cryptanalysis was introduced by Langford and Hellman [LH94].

Definition 1 (Differential-linear distinguisher). *Let $E : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a vectorian Boolean function, and let $\Delta \in \mathbb{F}_2^n$ and $\Gamma \in \mathbb{F}_2^n$ be an input difference and an output linear mask, respectively. Such a pair is called a* differential-linear distinguisher*, and its performance is measured through the* autocorrelation

$$\mathrm{Aut}_E(\Delta, \Gamma) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \Gamma, E(x) \rangle \oplus \langle \Gamma, E(x \oplus \Delta) \rangle}.$$

If E is a keyed instance of a block cipher, and given a collection of around $1/\mathrm{Aut}_E(\Delta, \Gamma)^2$ pairs of the form $(E(x), E(x \oplus \Delta))$, it is possible to distinguish the cipher from a random permutation. There are several methods to estimate the autocorrelation and to find good distinguishers (Δ, Γ). Since this paper focuses on key recovery and relies on existing distinguishers, they fall outside its scope.

Differential-linear distinguishers are often used in key recovery attacks. Consider a function $E_2 : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ where $E = E_2 \circ E_1$, and $K \in \mathbb{F}_2^\kappa$ is the key, and a DL distinguisher on E_1. We assume it can be truncated to a function $\langle \Gamma, E_2^{-1}(y, K) \rangle = f(t, k)$, where $t \in \mathbb{F}_2^r$ and $k \in \mathbb{F}_2^s$ are shortened truncations of

$y = E(x)$ and K. In other words, we assume that $\langle \Gamma, E_2^{-1}(y, K)\rangle$ can be determined with r bits of y and s bits of K. Given N ciphertext pairs, the attacker can construct a table indexed on all the possible guesses of k and containing the values $\frac{1}{N}\sum(-1)^{\langle \Gamma, f(t_1,k)\oplus f(t_2,k)\rangle}$, where (t_1, t_2) are obtained by truncating two ciphertexts $y_1 = E(x)$ and $y_2 = E(x+\Delta)$. If the number of pairs is large enough, the correct value of k can be retrieved by the attacker as it should correspond to (one of) the highest value(s) in this table. The time complexity of the attack is $\mathcal{O}(N2^s)$. By using a distillation step which counts the occurrences of each value of (t_1, t_2), it can be carried out in time $\mathcal{O}(N) + \mathcal{O}\left(2^{2r+s}\right)$.

2.3 Key Recovery Attacks on ChaCha and other ARX

The core difficulty when applying the attack to ARX constructions is the quick diffussion of the key recovery map. Let $f(Z, Z', K) = \langle \Gamma, E_2^{-1}(Z, K) \oplus E_2^{-1}(Z', K)\rangle$ be the combined key recovery map for both ciphertexts. This function involves a lot of bits of K (s is very large) and leads to a high attack complexity.

Probabilistic Neutral Bits (PNBs). So far, the most successful method for ChaCha cryptanalysis is PNBs [AFK+08]. Let E be reduced-round ChaCha and $E = E_2 \circ E_1$, where E_1 has a differential-linear distinguisher from Δ to Γ with correlation c_d. Let $\bar{Z}$ and $\bar{Z}'$ denote the keystream blocks after peeling off the modular addition with known input branches. Then, the key recovery map is

$$f(\bar{Z}, \bar{Z}', K) = \langle \Gamma, E_2^{-1}(\bar{Z}, K) \oplus E_2^{-1}(\bar{Z}', K)\rangle, \tag{1}$$

and is expected to display correlation c_d for the right value of K.

With PNBs, we approximate the key recovery map by setting non-significant key bits to constant. Significant and non-significant key bits are identified through an experimental approach. We first evaluate $f(\bar{Z}, \bar{Z}', K)$ under a known random key, as well as $f(\bar{Z}, \bar{Z}', K \oplus \mathrm{e}_i)$ for all i, i.e., after flipping the ith bit of K. If $f(\bar{Z}, \bar{Z}', K \oplus \mathrm{e}_i)$ and $f(\bar{Z}, \bar{Z}', K)$ exhibit higher correlation than a threshold, i is considered non-significant. We denote the remaining significant key bits $\bar{K}$ and substitute f for $g(\bar{Z}, \bar{Z}', \bar{K})$, where g is the same function as f except that it assigned fixed values to non-significant bits of K. We experimentally determine the correlation between $f(\bar{Z}, \bar{Z}', K)$ and $g(\bar{Z}, \bar{Z}', \bar{K})$, and c_a denotes the correlation. With independent assumption, we derive the total correlation as $c_d \times c_a$ when we guess $\bar{K}$ correctly.

Partitioning. Another method is advanced partitioning [BLT20], which is an extension of [BC14,Leu16]. Unlike PNBs, it relies on the properties of the modular addition. Existing works use a *2-bit tail*, but an *d-bit tail* can be used:

Lemma 1 (Partitioning technique). *Let $a, b \in \mathbb{F}_2^n$ and $z = a \boxplus b$ and $s = a \oplus b$. Let $\mathrm{tr}_i(s)$ denote the i rightmost bits of s. For $i > d$, we have*

$$z[i] = \begin{cases} a[i] \oplus b[i] \oplus a[i-1] & \text{if } \mathrm{tr}_{i-1}(s) = 0\|*^{i-1}, \\ a[i] \oplus b[i] \oplus a[i-2] & \text{if } \mathrm{tr}_{i-1}(s) = 1\|0\|*^{i-2}, \\ \vdots & \\ a[i] \oplus b[i] \oplus a[i-d] & \text{if } \mathrm{tr}_{i-1}(s) = 1^{m-1}\|0\|*^{i-m}, \end{cases}$$

$$z[i] \oplus z[i-1] = \begin{cases} a[i] \oplus b[i] & \text{if } \mathrm{tr}_{i-1}(s) = 1\|*^{i-1}, \\ a[i] \oplus b[i] \oplus a[i-1] \oplus a[i-2] & \text{if } \mathrm{tr}_{i-1}(s) = 0\|0\|*^{i-2}, \\ \vdots & \\ a[i] \oplus b[i] \oplus a[i-1] \oplus a[i-d] & \text{if } \mathrm{tr}_{i-1}(s) = 0\|1^{m-2}\|0\|\ *^{i-m}. \end{cases}$$

When we try to compute $z[i]$ or $z[i] \oplus z[i-1]$, partitioning with an d-bit tail can determine them with a probability of $1 - 2^{-d}$. If we need to guess either a or b, we need d-bit guess to identify each partition.

Since partitioning is applied to individual modular additions, the differential-linear distinguisher is extended until shortly before the last key addition. An advantage of partitioning is that data rejection can save time complexity. The valid keys for a given ciphertext are known immediately, so we can avoid traversing all ciphertexts for each key guess.

In advanced partitioning [BLT20], the extension of the differential-linear distinguisher is chosen dynamically depending on the observed keystream. As a result, they proposed an improved 6-round attack. Unfortunately, they were unable to describe a 7-round attack because extending the differential-linear distinguisher led to a low correlation. In this paper, we use Lemma 1 but do not use advanced partitioning. Therefore, please refer to [BLT20] for further details.

2.4 Puncturing for Linear Key Recovery Attacks

Walsh spectrum puncturing is a recent technique [FT24], presented in the context of linear cryptanalysis. The general idea is that the Boolean function f from a key recovery attack may be substituted for a different map g which approximates it. This map does not need to be a Boolean function, and can take arbitrary real values. The substitution is payed through increased data complexity.

Definition 2. *A* pseudoboolean function *is a map $f : \mathbb{F}_2^n \longrightarrow \mathbb{R}$. All Boolean functions $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ can be turned into pseudoboolean functions by taking $(-1)^f$. The* inner product *and* 2-norm *are defined as:*

$$\langle f, g \rangle = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} f(x)g(x), \quad \|f\|_2 = \sqrt{\langle f, f \rangle} = \sqrt{\frac{1}{2^n} \sum_{x \in \mathbb{F}_2^l} f(x)^2}.$$

Given $f : \mathbb{F}_2^l \longrightarrow \mathbb{R}$*, its* Walsh spectrum *is* $\widehat{f} : \mathbb{F}_2^n \longrightarrow \mathbb{R}$*, where*

$$\widehat{f}(u) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^l} (-1)^{\langle u,x \rangle} f(x).$$

A vectorial Boolean function *is a map* $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$*. Given* $v \in \mathbb{F}_2^m$*, the Boolean function* $f_v(x) = \langle v, f(x) \rangle$ *is called a* component *of* f*. The Walsh spectrum of* f *is a map* $\widehat{f} : \mathbb{F}_2^n \times \mathbb{F}_2^m \longrightarrow \mathbb{R}$ *defined as* $\widehat{f}(u, v) = \widehat{f_v}(u)$*.*

We note that the inner product and the norm can also be computed on the spectrum, as $\langle \widehat{f}, \widehat{g} \rangle = \frac{1}{2^n} \langle f, g \rangle$ and $\|\widehat{f}\|_2 = \frac{1}{\sqrt{2^l}} \|f\|_2$, and that $\widehat{\widehat{f}} = \frac{1}{2^n} f$.

The main result of [FT24] is the following:

Proposition 1 (Corollaries 3 and 5 in [FT24]). *Let* $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$ *be the key recovery map for a linear attack, and let* $g : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$ *be another function. Then* f *can be swapped for* g *in the attack by increasing the data complexity by* $\frac{1}{\rho^2}$*, where* $\rho = \frac{|\langle f,g \rangle|}{\|f\|_2 \cdot \|g\|_2}$ *is called* correlation coefficient*.*

One way to construct g *is to remove coefficients from the Walsh spectrum of* f *and set them to zero. That is, given a set of indices* $\mathcal{P} \subseteq \mathbb{F}_2^\ell$*, we build* g *with* $\widehat{g}(u) = \widehat{f}(u)$ *if* $(u) \notin \mathcal{P}$*, and* $\widehat{g}(u) = 0$ *otherwise. The data complexity increases by* $\frac{1}{1-\epsilon}$*, where* $\epsilon = \sum_{(u) \in \mathcal{P}} \widehat{f}(u)^2$ *is called* puncturing coefficient*. The proportion of* f *which remains,* $\rho^2 = 1 - \epsilon$*, is called puncturing correlation, and* $\rho^2 = 1 - \epsilon = \sum_{(u) \in \mathbb{F}_2^\ell \setminus \mathcal{P}} \widehat{f}(u)^2$*.*

3 Bit Puncturing for ARX

This section discusses the application of bit puncturing to ARX constructions from the perspective of the addition operation. Subsection 3.1 discusses how puncturing works for differential-linear attacks. Subsection 3.2 explains some alternative interpretations of bit puncturing not included in [FT24]. Subsection 3.3 applies bit puncturing to a single modular addition. Finally, Subsection 3.4 compares bit puncturing to existing techniques.

3.1 Puncturing for Differential-linear Attacks

Algorithmically, the key recovery for a differential-linear attack can be interpreted as that of a linear attack where the roles of plaintext and ciphertext are represented by both the ciphertexts in each pair. For clarity, we consider that we puncture half the key recovery map (the function corresponding to one of the ciphertexts). The following result is adapted from Corollaries 3 and 5 in [FT24]:

Proposition 2. *Let* $f : \mathbb{F}_2^r \times \mathbb{F}_2^s \longrightarrow \mathbb{F}_2$ *be the key recovery map for one ciphertext in a DL attack, and let* $g : \mathbb{F}_2^r \times \mathbb{F}_2^s \longrightarrow \mathbb{R}$*. Then* f *can be swapped for* g *by increasing the data complexity by a factor of* $\frac{1}{\rho^4}$*, where* $\rho = \frac{|\langle f,g \rangle|}{\|f\|_2 \cdot \|g\|_2}$*.*

Given a set of indices $\mathcal{P} \subseteq \mathbb{F}_2^r \times \mathbb{F}_2^s$, let g be a punctured version of f, so that $\widehat{g}(u,v) = \widehat{f}(u,v)$ if $(u,v) \notin \mathcal{P}$, and $\widehat{g}(u,v) = 0$ otherwise. Then the data complexity increases by $\frac{1}{(1-\epsilon)^2}$, where $\epsilon = \sum_{(u,v)\in\mathcal{P}} \widehat{f}(u,v)^2$.

We now briefly discuss the assumptions which are required for this result to hold. In the proof of Corollary 3 in [FT24], it is assumed that the key recovery maps f and g behaves under the attack data in the same way it would from a randomly sampled input. In the case of differential-linear attacks, we note a potential issue. The key recovery maps are of the form $f(t_1,k)\cdot f(t_2,k)$, that is, they are the product of two identical functions under partially different inputs. Furthermore, both functions are given the same key. In [FT24], the key is assumed to be XORed directly to the plaintext and ciphertext, so the inputs to f are still fully randomised under the assumption that the ciphertexts are random. However, this is not the case in this paper's application scenario, as the key is added to the ciphertext through modular additions. As a result, it is possible to construct simple cases where not enough randomisation occurs (for example, if f is just the modular addition of the ciphertext and the key, both components are not independent and the product does not behave as expected). As a result, we introduce the following formal assumption:

Assumption 1 *Let $f : \mathbb{F}_2^r \times \mathbb{F}_2^s \to \mathbb{R}$ be the key recovery map or its approximation. We consider that the key guess k is fixed, and that (T_1, T_2) are random vectors obtained by truncating ciphertext pairs from the attack. We assume that*

- *The average of f under T_1 and T_2 is the same under all fixed k, and equal to the expected value of f, $\widehat{f}(0)$.*
- *The variance of f under T_1 and T_2 is the same under all fixed k, and equal to the variance of f, $\|f\|_2^2 - \widehat{f}(0)^2$.*
- *The values of f under T_1 and T_2 for a fixed k are statistically independent.*

Proposition 2 is also supported by the 6-round experiments (see Appendix C.1).

3.2 Understanding Bit Puncturing

In [FT24], several puncturing strategies are introduced, one of which is called *bit puncturing*. The idea is to remove certain bits from the input domain of f in a way which will improve the complexity of the attack. We now provide a formal definition, as well as make a couple of observations which were not in [FT24].

Definition 3. *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{R}$ be a pseudoboolean function, and let $\mathcal{B} \subseteq \{0,\dots,n-1\}$ be a subset of indices. The* bit puncturing *of f according to $\mathcal{B}$ is a function $g : \mathbb{F}_2^n \longrightarrow \mathbb{R}$ whose Walsh spectrum $\widehat{g}$ is:*

$$\widehat{g}(u) = \begin{cases} \widehat{f}(u) \text{ if } u_i = 0 \text{ for all } i \in \mathcal{B} \\ 0 \quad \text{otherwise} \end{cases}$$

We note that $g(x) = g(x \oplus y)$ if $y \in \mathrm{span}\{\mathrm{e}_i : i \in \mathcal{B}\}$. Indeed, g is the average of the values of f over all the inputs obtained by flipping bits in $\mathcal{B}$.

Proposition 3. *Let g be the bit puncturing of $f : \mathbb{F}_2^n \longrightarrow \mathbb{R}$ according to $\mathcal{B} \subseteq \mathbb{F}_2^n$.*

$$g(x) = \frac{1}{2^{|\mathcal{B}|}} \sum_{t \in \mathbb{F}_2^{|\mathcal{B}|}} f\left(x \oplus \bigoplus_{i \in \mathcal{B}} t_i \cdot \mathrm{e}_i\right).$$

Proof. We assume $\mathcal{B} = \{i\}$, as the result can be iterated. Let $h(x) = \frac{1}{2}(f(x) \oplus f(x \oplus \mathrm{e}_i))$, and we will prove that $\widehat{h} = \widehat{g}$.

$$\widehat{h}(u) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle u, x \rangle} \left(\frac{1}{2}(f(x) + f(x \oplus \mathrm{e}_i))\right)$$

If $u_i = 1$, then the term of the sum corresponding to x will be cancelled by the term corresponding to $x \oplus \mathrm{e}_i$, which is the same but with its sign flipped. Therefore, $\widehat{h}(u) = 0 = \widehat{g}(u)$. If $u_i = 0$, on the other hand, both terms have the same sign, and they add up to $(-1)^{\langle u, x \rangle} f(x) + (-1)^{\langle u, x \oplus \mathrm{e}_i \rangle} f(x \oplus \mathrm{e}_i)$. We conclude that $\widehat{h}(u) = \widehat{f}(u) = \widehat{g}(u)$. In summary, $\widehat{h} = \widehat{g}$. □

3.3 Puncturing a Single Modular Addition

We now focus on the modular addition operator, and show a bit puncturing strategy which can reduce the number of active input bits while keeping a high correlation coefficient. Although it is difficult to apply directly to key recovery attacks as discussed in the following section, it provides a purely theory-based way of measuring the impact of ignoring the rightmost bits of a modular addition when computing a parity check on the output. The proofs rely on this result:

Lemma 2 ([Sch13]). *Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the modular addition $z = f(x, y) = x \boxplus y$, and consider masks $u, v, w \in \mathbb{F}_2^n$. The coefficient $\widehat{f}(u, v, w) = \widehat{f_w}(u, v) = \frac{1}{2^{2n}} \sum_{x,y \in \mathbb{F}_2^n} (-1)^{\langle u, x \rangle \oplus \langle v, y \rangle \oplus \langle w, x \boxplus y \rangle}$ is obtained as follows:*

1. *Compute $\gamma = u \oplus v \oplus w$.*
2. *Deduce $\sigma \in \mathbb{F}_2^n$ recursively: $\sigma_{n-1} = 0$, $\sigma_{i-1} = \sigma_i \oplus \gamma_i$ for $i < n$.*
3. *If there is at least one i between 0 and $n-1$ so that $\sigma_i = 0$ and the equalities $u_i = v_i = w_i$ do not hold, then $\widehat{f}(u, v, w) = 0$.*
4. *Otherwise $\widehat{f}(u, v, w) = (-1)^{\langle u \oplus w, v \oplus w \rangle} 2^{-\mathrm{wt}(\sigma)}$.*

For clarity, we will also consider $\sigma_{-1} = \sigma_0 \oplus \gamma_0$. Given $u \in \mathbb{F}_2^n$, $u \neq 0$, we denote $\overrightarrow{u}$ the position of the rightmost nonzero bit of u, $\overrightarrow{u} = \min\{i : u_i \neq 0\}$. Similarly, $\overleftarrow{u}$ is the position of the leftmost nonzero bit of u, $\overleftarrow{u} = \max\{i : u_i \neq 0\}$.

The following result corresponds to the following situation: we assume that the target output masks w satisfy $\overrightarrow{w} \geq l$, where l is some predetermined bit position. We consider that we puncture the bits of the inputs corresponding to $r \geq \overrightarrow{u}$ and $r \geq \overrightarrow{v}$. Then, the resulting puncturing correlation is mainly determined by the length of the *tail* $d = l - r$, that is, the number of buffer bits to the right of the nonzero part of w.

Proposition 4. *Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the modular addition operation $z = f(x, y) = x \boxplus y$. Let $n \geq l \geq r$ be two natural numbers, and $d = l - r$. Consider an output mask $w \in \mathbb{F}_2^n$ satisfying $l \geq \overrightarrow{w}$. Let $g_w : \mathbb{F}_2^n \longrightarrow \mathbb{R}^n$ be a bit puncturing of f_w defined as follows:*

$$\widehat{g_w}(u, v) = \begin{cases} \widehat{f}(u, v, w) & \text{if } \overleftarrow{w} \geq \overleftarrow{u}, \overleftarrow{v} \text{ and } \overrightarrow{u}, \overrightarrow{v} \geq r \\ 0 & \text{otherwise} \end{cases}.$$

Then $1 - \epsilon \geq 1 - \left(2^{-d} - 2^{-l-r}\right)$.

Proof. We show a sketch of the proof and refer to Appendix A.1 for the complete proof. We consider a set $\mathcal{T} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n$ containing all indices (u, v) so that $\widehat{f}(u, v, w) \neq 0$, and a set of $\mathcal{S} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n$ containing the indices which are kept when building $\widehat{g_w}$. The aim is to compute $1 - \epsilon = \sum_{(u,v)\in\mathcal{S}} \widehat{f}(u, v, w)^2$. We note that $\sum_{(u,v)\in\mathcal{T}} \widehat{f}(u, v, w)^2 = 1$, so it suffices to compare the sum over $\mathcal{S}$ to the sum over $\mathcal{T}$. This is done by separating the sums into segments, each one of which corresponding to a fixed header $(u^0, v^0) \in \mathbb{F}_2^{n-l} \times \mathbb{F}_2^{n-l}$, where u^0 and v^0 represent the $n-l$ leftmost bits of u and v. The nonzero coefficients of $\widehat{f_w}$ starting with these headers, which conform a set denoted $\mathcal{T}(u^0, v^0)$ can be enumerated after proving that the rightmost l bits of σ must necessarily be of the form $(1, \ldots, 1, 0, \ldots, 0)$. It is noted that these indices also belong to $\mathcal{S}(u^0, v^0)$ depend on the which position in which the shift from 1 to 0 occurs, as they are kept if the switch happens to the left of the r-th bit, and removed if it happens to the right. After counting the associated coefficients and computing the correlation, it is shown that the quotient between the partial sums is at least $1 - \left(2^{-d} - 2^{-l-r}\right)$. After combining all the partial sums, the stated bound is obtained. □

We note that, although we assume that $\overrightarrow{w} \geq l$, the bound is tightest when $\overrightarrow{w} = l$. However, the result is stated so that the bound can be applied to all masks in a family at the same time. We note that even when $\overrightarrow{w} = l$ the equality doesn't necessarily hold, as happens for example when $w = (0, \ldots, 0, 1, 1, 0, \ldots, 0)$.

When interpreting the function g_w as the average over all possible values of the punctured bits, we note that the only relevant information about the rightmost bits is the carry into position r. This means g_w is the weighted average of two Boolean functions, corresponding to two modular additions over $\mathbb{F}_2^{n-r}$. The first one is denoted f^0, and covers the case where the initial carry is 0 (so it's the modular addition). The second is denoted f^1 and assumes the initial carry is 1. The weights are given by the probabilities that the carry from the $r-1$ to the r-th bit is zero or one. As a result, $g_w = \left(\frac{1}{2} + 2^{-r}\right) f_w^0 + \left(\frac{1}{2} - 2^{-r}\right) f_w^1$. In particular, this means we can retrieve all g_w (2^{n-r} real-valued functions on $\mathbb{F}_2^n$) from just two vectorial boolean functions ($2(n-r)$ Boolean functions on $\mathbb{F}_2^n$).

Other puncturing schemes are possible, which are included in Appendix B.

3.4 Comparing Puncturing to PNBs and Partitioning

We briefly compare bit puncturing to the two dominant existing techniques.

PNBs. When applying the PNBs technique to attacks on ARX constructions, it is assumed that the non-significant bits take a fixed value (usually zero). In contrast, in the case of puncturing, all possible values of the punctured bits are considered, and the average is returned. This means that bit puncturing can extract an optimal amount of information from the remaining bits. In addition, PNBs rely on experiments to determine the backward correlation, while puncturing provides a closed formula which is easier to manipulate when designing attacks. The theory of [FT24] can also be applied to PNBs, and the case in which the least significant bits are set to zero is covered in Appendix B.

Partitioning. Partitioning, as a form of plaintext rejection, also falls under the scope of [FT24]. In addition, from Lemma 1, we know that the data complexity increase is $1/(1-2^{-d})$ for a d-bit tail. With the same tail length, the data complexity increase for puncturing is $1/(1-2^{-d}+2^{-r-l})$, which is slightly smaller depending on the position of the target bits. The main difference is that while (advanced) partitioning can only handle one modular addition in each key-recovery procedure, bit-puncturing can handle the entire complex ARX structure by processing it as an analysis of the Walsh spectrum.

4 Practical Method to Construct Punctured Spectrum

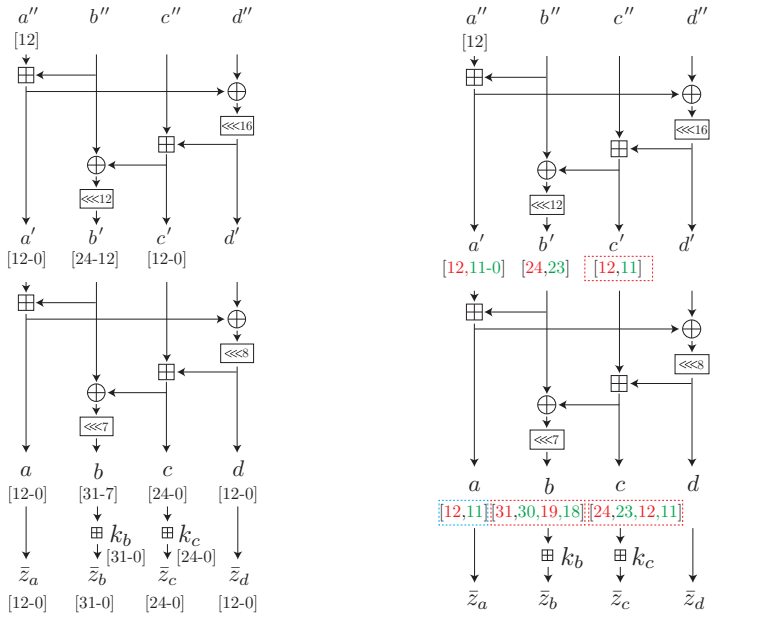
We discussed the theoretical aspect of bit puncturing for ARX constructions in the previous section. Proposition 4 analyzes the bit puncturing for a single modular addition. However, in practice, the key recovery map involves more than a modular addition. The authors of [FT24] show that it is possible to puncture f_2 in a composition $f = f_2 \circ f_1$ and the resulting approximation has the same correlation. As they target mainly SPN ciphers, puncturing f_2 removes active S-boxes from f_1 and significantly reduces the number of active key bits. Unfortunately, this is not as useful in the cryptanalysis of ARX or ChaCha.

For example, see Fig. 2, which features a quarter-round of ChaCha. Considering the exact key-recovery map to compute $a''[12]$, it is

$$(-1)^{a''[12]} = f(z_a[12-0], z_b[31-0], z_c[24-0], z_d[12-0], k_b[31-0], k_c[24-0]).$$

Even if we puncture the modular addition immediately after a'', it does not make any keybits inactive. To successfully apply puncturing, we must consider the whole map, or at least the final key addition. Then, Proposition 4 does not apply, and we need another method to resolve this problem. We also note that a result equivalent to the one of [FT24] for puncturing f generally does not hold.

As a practical solution, we introduce *trail enumeration puncturing*. Each Walsh coefficient is the correlation of a linear approximation, and as such can be computed as the sum of the (signed) correlations of all the linear trails in the approximation's linear hull. Moreover, for ARX ciphers, the correlation distribution among all trails is very uneven. More specifically, among all the linear trails contributing to a coefficient with high value, a few dominant trails represent



(a) Exact map. (b) Example of punctured map.

Fig. 2: Example of the key-recovery map.

most of the correlation, and the rest have negligible impact. This means that these correlations can be estimated accurately using just a few trails.

We start from the target bit, $a''[12]$ in the example, and consider all the linear trails which extend it. This trail extension is performed step by step, for example, one modular addition at a time. In each step, we consider all trail extensions for each approximation in the current set, ignorning those with small correlation. Once we have extended all approximations from the previous step, we combine the trail correlations for any repeating output masks to obtain the set of approximations for the next step. This is repeated until we reach the final round and we have a partial Walsh spectrum of the key-recovery map, which will be used as the punctured Walsh spectrum. The puncturing correlation is estimated as the sum of the squared coefficients of the partial Walsh spectrum.

Example 1. Figure 2b shows an example of trail enumeration, where the red bits are always active, the green bits are optionally active, and the rest are inactive.

Our goal is to evaluate $a''[12]$ from $(\bar{z}_a, \bar{z}_b, \bar{z}_c, \bar{z}_d)$ and the partially guessed k_b and k_c. For this purpose, we enumerate linear trails from $a''[12]$.

$$a''[12] = (a' - ((b' \ggg 12) \oplus c'))\,[12].$$

Given linear masks for a', b', and c', we compute its correlation by using Lemma 2, where the linear mask for c' has to be equal to that for $(b' \ggg 12)$. As depicted

in Figure 2b, we restrict the masks of c' to [12] and [11]. In other words, we ignore all masks where there are active bits in $c'[10-0]$ or $b'[22-11]$. As a result, the following 26 nonzero correlation linear masks for (a', b', c') are considered:

00001800	01000000	00001000	$+2^{-1.00}$	00001000	01800000	00001800	$+2^{-1.00}$
00001400	01000000	00001000	$+2^{-2.00}$	00001C00	01800000	00001800	$+2^{-2.00}$
00001200	01000000	00001000	$+2^{-3.00}$	00001A00	01800000	00001800	$+2^{-3.00}$
00001100	01000000	00001000	$+2^{-4.00}$	00001900	01800000	00001800	$+2^{-4.00}$
00001080	01000000	00001000	$+2^{-5.00}$	00001880	01800000	00001800	$+2^{-5.00}$
00001040	01000000	00001000	$+2^{-6.00}$	00001840	01800000	00001800	$+2^{-6.00}$
00001020	01000000	00001000	$+2^{-7.00}$	00001820	01800000	00001800	$+2^{-7.00}$
00001010	01000000	00001000	$+2^{-8.00}$	00001810	01800000	00001800	$+2^{-8.00}$
00001008	01000000	00001000	$+2^{-9.00}$	00001808	01800000	00001800	$+2^{-9.00}$
00001004	01000000	00001000	$+2^{-10.00}$	00001804	01800000	00001800	$+2^{-10.00}$
00001002	01000000	00001000	$+2^{-11.00}$	00001802	01800000	00001800	$+2^{-11.00}$
00001001	01000000	00001000	$+2^{-12.00}$	00001801	01800000	00001800	$+2^{-12.00}$
00001000	01000000	00001000	$+2^{-12.00}$	00001800	01800000	00001800	$+2^{-12.00}$

Then, we obtain an approximation which estimates $(-1)^{a''[12]}$ from $a'[12-0]$, $b'[24, 23]$, and $c'[12, 11]$, and the puncturing correlation is estimated as $2^{-0.58}$, which is the sum of the 26 squared correlations.

We next use Lemma 2 for each of the above 26 linear masks and obtain linear masks for (a, b, c, d, c'). If different masks of (a', b', c') transition to the same mask of (a, b, c, d, c'), their correlations are added as part of the same linear hull. As a result, we have the following 8 linear masks for (a, b, c, d, c').

00001000	C0080000	01801000	00000000	00001800	$+2^{-12.00}$
00001000	C00C0000	01801800	00000000	00001800	$+2^{-1.58}$
00001800	800C0000	01001800	00000000	00001000	$+2^{-1.58}$
00001800	80080000	01001000	00000000	00001000	$+2^{-12.00}$
00001800	C00C0000	01801800	00000000	00001800	$+2^{-12.00}$
00001800	C0080000	01801000	00000000	00001800	$+2^{-1.58}$
00001000	800C0000	01001800	00000000	00001000	$+2^{-12.00}$
00001000	80080000	01001000	00000000	00001000	$+2^{-1.58}$

The sum of their squared correlations is about $2^{-1.17}$.

We further extend them by evaluating the last key addition, where we use $\bar{z}_b[31, 30, 19, 18]$, $\bar{z}_c[24, 23, 12, 11]$, $\bar{z}'_c[12, 11]$, $k_b[31, 30, 19, 18]$, and $k_c[24, 23, 12, 11]$, where please refer to Sect. 2.1 for $\bar{z}'_c$. Since some bits are always active, the punctured key-recovery map can be written as

$$(-1)^{a''[12]} \approx (-1)^{\bar{z}_a[12]\oplus\bar{z}_b[31]\oplus\bar{z}_b[19]\oplus\bar{z}_c[24]\oplus\bar{z}_c[12]\oplus\bar{z}'_c[12]\oplus k_b[31]\oplus k_b[19]\oplus k_c[24]} \times g_1(\bar{z}_a[11], \bar{z}_b[30, 18], \bar{z}_c[23, 11], \bar{z}'_c[11], k_b[30, 18], k_c[23, 11]).$$

In the end, we have 768 coefficients, and the puncturing correlation is estimated as $2^{-6.17}$. We construct the truth table of the punctured key-recovery map by applying the Fast Walsh Transform (FWT) on its Walsh spectrum. Since the size of the Walsh spectrum is 2^{10}, the cost of the FWT algorithm is 10×2^{10}

Table 2: Example of the trail enumeration

State	Active	# coeffs.	ρ^2
(a'', b'', c'', d'')	$a''[12]$	1	1
(a', b', c', d')	$a'[12,11\text{-}0]$, $b'[24, 23]$, $c'[12, 11]$	26	$2^{-0.58}$
(a, b, c, d, c')	$a[12, 11]$, $b[31, 30, 19, 18]$, $c[24, 23, 12, 11]$, $c'[12, 11]$	8	$2^{-1.17}$
$(\bar{Z}, k_b, k_c)$	$a[12, 11] \leftarrow \{\bar{z}_a[12, 11]\}$, $b[31,30, 19,18] \leftarrow \{\bar{z}_b[31,30, 19,18], k_b[31,30, 19,18]\}$, $c[24,23, 12,11] \leftarrow \{\bar{z}_c[24,23, 12,11], k_c[24,23, 12,11]\}$, $c'[12, 11] \leftarrow \{\bar{z}'_c[12, 11], k_c[12,11]\}$	768	$2^{-6.17}$

additions. For the full differential-linear attack, the final puncturing correlation is estimated as $2^{-6.17\times 2} = 2^{-12.34}$.

In this paper, we use a representation like Table 2 to summarize the behavior of active coefficients at each state and the sum of the squared correlations. Here, red-colored bits are always active, and green-colored bits are arbitrary.

As experimental verification, we generated 2^{15} random pairs for 1000 keys and evaluated the correlation. We expect a distribution with average $2^{-12.34}$ and variance $2^{-12.34-15} = 2^{-27.34}$. We observed average $2^{-12.21}$ and variance $2^{-27.09}$.

Example 2. We can improve the accuracy of the approximation without expanding the key guess. For example, by using 2 bits instead of 1 bit for $\bar{z}_a, \bar{z}_b, \bar{z}_c, \bar{z}'_c$. Then, we have 27712 coefficients, and the punctured map is

$$\begin{aligned}(-1)^{a''[12]} \approx &(-1)^{\bar{z}_a[12]\oplus\bar{z}_b[31]\oplus\bar{z}_b[19]\oplus\bar{z}_c[24]\oplus\bar{z}_c[12]\oplus\bar{z}'_c[12]\oplus k_b[31]\oplus k_b[19]\oplus k_c[24]}\times \\ &g_2(\bar{z}_a[11, 10], \bar{z}_b[30, 29, 18, 17], \bar{z}_c[23, 22, 11, 10], \bar{z}'_c[11, 10], \\ &k_b[30, 18], k_c[23, 11]).\end{aligned} \tag{2}$$

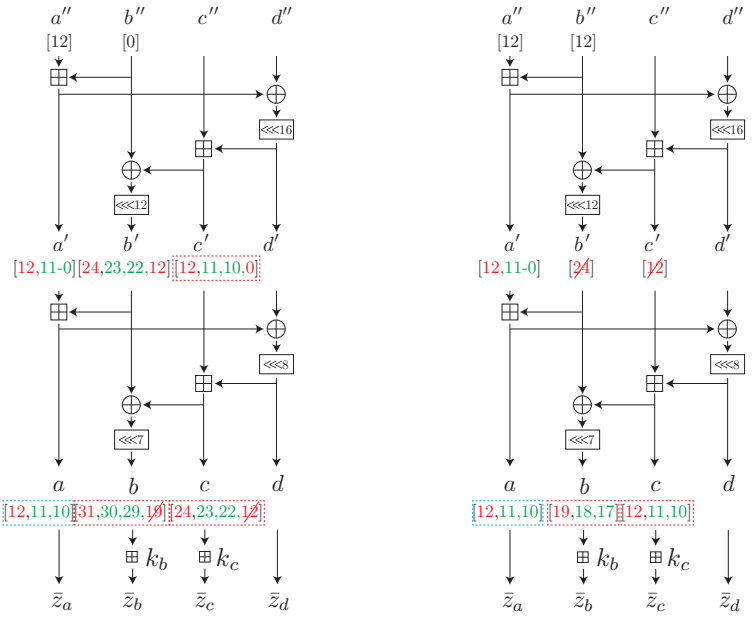
The correlation is improved to $2^{-4.37}$, or $2^{-8.74}$ for the full key recovery in differential-linear attack. We used 2^{12} pairs for 1000 keys in the experiment, and observed $2^{-8.58}$ for the average and $2^{-18.14}$ for the variance.

Example 3. Increasing the amount of intermediate masks is another way to improve the accuracy of the approximation. We construct g_3 considering

$$b[31, 30, 29, 19, 18, 17], c[24, 23, 22, 12, 11, 10], c'[12, 11, 10],$$

which has the same domain as g_2. We have 30080 coefficients, and the puncturing correlation increases to $2^{-4.28}$, or $2^{-8.56}$ in the full key recovery. With the same experiment setup, the average and variance were $2^{-8.38}$ and $2^{-17.95}$, respectively. As predicted, the correlation slightly increases from Example 2.

The examples above show that trail enumeration works very well for the key-recovery map of differential-linear attacks against ChaCha. We emphasize that this method relies on assumptions about the Walsh spectrum and associated linear hulls, and its usability depends on the application.



(a) Example of cancellation 1. (b) Example of cancellation 2.

Fig. 3: Examples of key-recovery map with cancellation.

4.1 On the Impact of Cancellations

Cancellations may be leveraged when targeting several bits simultaneously.

Example 4. In Figure 3a, we target $a''[12] \oplus b''[0]$. In this case, $b[19]$ and $c[12]$ cancel out and we can exclude $k_b[19, 18]$ from the key guess. The key recovery is

$$(-1)^{a''[12]\oplus b''[0]} \approx (-1)^{\bar{z}_a[12]\oplus\bar{z}_b[31]\oplus\bar{z}_c[24]\oplus\bar{z}'_c[12]\oplus\bar{z}'_c[0]\oplus k_b[31]\oplus k_c[24]\oplus k_c[12]\oplus k_c[0]} \times g_4(\bar{z}_a[11,10], \bar{z}_b[30,29], \bar{z}_c[23,22], \bar{z}'_c[11,10], k_b[30], k_c[23,11]).$$

The size of the required guess is 3 bits instead of 4. The puncturing correlation is $2^{-3.94}$, and $2^{-7.88}$ in the full attack. We tested this estimation with 2^{12} pairs for 1000 keys, and observed average $2^{-7.78}$ and variance $2^{-17.94}$.

Example 5. Figure 3b shows another example, where $k_b[31, 30]$ and $k_c[24, 23]$ are excluded from the key-recovery map. The estimated puncturing correlation is $2^{-3.33}$, or $2^{-6.66}$ for the differential-linear key recovery. With the same experiment as Example 4, we observed average $2^{-6.61}$ and variance $2^{-16.15}$.

4.2 Independent Bits Assumption

Assuming independence can also convenient when constructing the punctured key-recovery map. Unlike with cancellation, we handle several target bits inde-

pendently and construct the punctured map for each bit separately. Then, we use the product of the punctured maps to obtain the final key-recovery map.

Example 6. The independence assumption is reasonable when the domain of each key-recovery map is disjoint. Supposing that the target is $a''[24] \oplus a''[12]$, we use g_3 for $a''[12]$ and construct the punctured map for $a''[24]$ similarly as

$$(-1)^{a''[24]} \approx (-1)^{\bar{z}_a[24]\oplus\bar{z}_b[11]\oplus\bar{z}_b[31]\oplus\bar{z}_c[4]\oplus\bar{z}_c[24]\oplus\bar{z}'_c[24]\oplus k_b[11]\oplus k_b[31]\oplus k_c[4]} \times g(\bar{z}_a[23,22], \bar{z}_b[10,9,30,29], \bar{z}_c[3,2,23,22], \bar{z}'_c[23,22], k_b[10,30], k_c[3,23]).$$

The puncturing correlations for $a''[12]$ and $a''[24]$ are $2^{-4.28}$ and $2^{-4.27}$, respectively. The final correlation is estimated as $2^{(-4.28-4.27)\times 2} = 2^{-17.10}$. We experimentally verified our estimation using 2^{20} pairs for 1000 keys. The average correlation was $2^{-15.76}$ and the variance was $2^{-32.11}$.

Example 7. When the target bits are closer to each other, the assumption is less reliable. We consider $a''[12] \oplus a''[11]$ as the target, and puncture $a''[12]$ and $a''[11]$ independently. The theoretical correlation is $2^{(-4.28-4.28)\times 2} = 2^{-17.14}$, but experimentally the average was $2^{-15.74}$ and the variance was $2^{-32.97}$.

Example 7 shows that independent evaluation does not necessarily cause a significant error even if the target bits are consecutive. However, the correlation is suboptimal. Considering $a''[12] \oplus a''[11]$ as a whole, we construct a punctured map whose correlation is $2^{-8.57}$, which is superior to independent analysis.

Remark 1. We sometimes have the same bit in c, c', and c'' in the target bits, e.g., the target bits are $c[10]$ and $c'[10]$. For convenience, we handle these bits independently. We experimentally verified this assumption.

Remark 2. We notice that the experimental average correlation is as expected, but the variance is slightly higher than expected. We estimate the variance, assuming the correlation is the same for all keys. However, in reality, the correlation is key-dependent. Such a phenomenon was already observed for PNBs; the variance is extremely high and it never follows the normal distribution. Therefore, the authors of [AFK+08] used the median instead of the average (guaranteeing 50% success probability). We indeed observe the gap between the theoretical and experimental variances in bit puncturing, but it is enough lower than in the case of PNBs. During the 6-round attack experiment, the distribution looks normal, i.e., the average is almost equal to the median. We discuss this issue in detail in Appendix C.1.

4.3 On Puncturing the Composition of Functions

Trail enumeration starts from each target bit and extends the linear trails step by step. Since the correlation of each trail is highly biased for ARX, trail enumeration is practical. However, the longer we extend each trail, the lower each coefficient becomes, and many coefficients are required for a reasonable puncturing correlation, making the trail enumeration impractical. The following proposition is helpful when the number of trails becomes unmanageable.

Proposition 5. *Let $f_1 : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $f_2 : \mathbb{F}_2^m \to \mathbb{F}_2$ be two Boolean functions, and let $f = f_2 \circ f_1$ be their composition. Let $g_2 : \mathbb{F}_2^m \to \mathbb{R}$ be an approximation of f_2 by puncturing Wlash spectrum coefficients. Furthermore, let $g_{1,u} : \mathbb{F}_2^n \to \mathbb{R}$ be an approximation of $\langle u, f_1 \rangle$ by puncturing Walsh spectrum coefficients, where the puncture set is the same for any u. We define the spectrum of $g : \mathbb{F}_2^n \longrightarrow \mathbb{R}$ as the matrix-vector product $\widehat{g} = \widehat{g_2} \times \widehat{g_1}$. Then, we have*

$$\rho^2 \approx \sum_{u \in \mathbb{F}_2^n} (\widehat{g}(u))^2 = 2^n \times \sum_{v \in \mathbb{F}_2^m} \sum_{w \in \mathbb{F}_2^m} \widehat{g_2}(v)\widehat{g_2}(w)\langle \widehat{g_{1,v}}, \widehat{g_{1,w}} \rangle. \tag{3}$$

See Appendix A.2 for the proof. We can apply this result to ChaCha and similar ciphers by considering that f_1 covers the final key addition, while f_2 covers the rest. We obtain g_2 using trail enumeration and g_1 with a bit puncturing of the modular subtraction. With this result, and with careful analysis of $\langle \widehat{g_{1,v}}, \widehat{g_{1,w}} \rangle$, we can compute $\sum_{u \in \mathbb{F}_2^n} (\widehat{g}(u))^2$ and approximate the puncturing correlation.

5 Improved Attacks on 6-Round ChaCha

5.1 3.5-Round DL Distinguisher

We use the same distinguisher as [BLT20,CN20]. Four rounds are divided into three sub ciphers, E_1 covering one round, E_m covering 2.5 rounds, and E_2 covering 0.5 rounds. For E_1, we use the following differential characteristic $\Delta^0 \xrightarrow{1R} \Delta^1$:

$$(\Delta^0_{12+i}[6]) \xrightarrow{1R} (\Delta^1_i[2], \Delta^1_{4+i}[29, 17, 9, 5], \Delta^1_{8+i}[30, 22, 10], \Delta^1_{12+i}[30, 10]).$$

The probability that pairs with input difference Δ^0 satisfy this characteristic is 2^{-5} on average. For the middle layer we have $\Delta^1 \xrightarrow{2.5R} \Gamma^{3.5}$:

$$(\Delta^1_i[2], \Delta^1_{4+i}[29, 17, 9, 5], \Delta^1_{8+i}[30, 22, 10], \Delta^1_{12+i}[30, 10]) \xrightarrow{2.5R} (\Gamma^{3.5}_{(i+1) \bmod 4}[0])$$

for $i \in \{0, 1, 2, 3\}$ with an autocorrelation of $2^{-8.3}$. In this section, we use $i = 0$.

The authors of [BLT20] proposed the technique, where right pairs, satisfying $\Delta^0 \xrightarrow{1R} \Delta^1$, are efficiently collected. Since only the first column is active in this 1st round differential, we first fix a pair for the 1st column and activate the last three columns. In such a way, $2^5 \times 2^{8.3 \times 2}$ pairs are enough to distinguish, that is lower than $2^{13.3 \times 2}$. If we use this trick, the required number of pairs for the main differential-linear attack must be lower than 2^{96}.

5.2 Linear Decomposition of the Key-Recovery Map

Beierle et al. extended the DL distinguisher to just before the last key addition for using advanced partitioning and analyzed each target bit independently [BLT20]. The mask propagates from round 3.5 to 4.5 with correlation 1. We then partially extend it only when the correlation is 1 (see Figure 4). The output parity of the 3.5-round DL distinguisher is the sum of the blue bits.

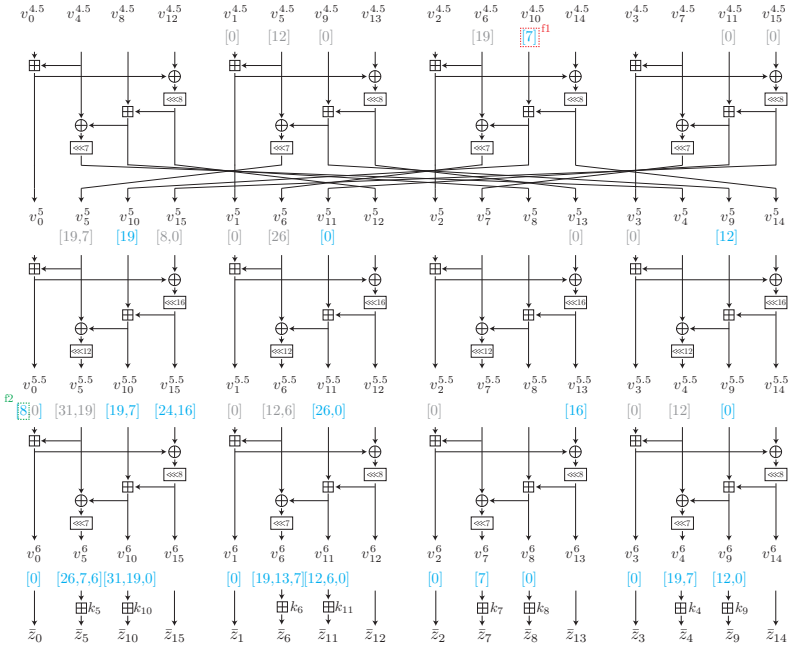


Fig. 4: Linear decomposition of the 6-round key-recovery map.

We decompose the key recovery as a sum of functions $f_i(\bar{Z}, K)$. First,

$$f_1 = (-1)^{v_{10}^{4.5}[7]}, \qquad f_2 = (-1)^{v_0^{5.5}[8]},$$

which involve more than one modular subtraction. The others can be evaluated by subtracting the secret key from the keystream or its equivalent:

$$\begin{aligned}
f_3 &= (-1)^{v_4^6[19]}, & f_4 &= (-1)^{v_4^6[7]}, & f_5 &= (-1)^{v_5^6[26]}, & f_6 &= (-1)^{v_5^6[7] \oplus v_5^6[6]},\\
f_7 &= (-1)^{v_6^6[19]}, & f_8 &= (-1)^{v_6^6[13]}, & f_9 &= (-1)^{v_6^6[7]}, & f_{10} &= (-1)^{v_7^6[7]},\\
f_{11} &= (-1)^{v_9^6[12]}, & f_{12} &= (-1)^{v_{10}^6[31]}, & f_{13} &= (-1)^{v_{10}^6[19]}, & f_{14} &= (-1)^{v_{11}^6[12]},\\
f_{15} &= (-1)^{v_{11}^6[6]}, & f_{16} &= (-1)^{v_9^5[12]}, & f_{17} &= (-1)^{v_{10}^5[19]}, & f_{18} &= (-1)^{v_{10}^{5.5}[19]},\\
f_{19} &= (-1)^{v_{10}^{5.5}[7]}, & f_{20} &= (-1)^{v_{11}^{5.5}[26]}.
\end{aligned}$$

Note that f_6 handles $v_5^6[7]$ and $v_5^6[6]$ together because they are consecutive. When the active bit is the lsb, e.g., $v_{10}^6[0]$, we need to involve the corresponding bit of $\bar{Z}$ but do not need to guess the corresponding key. We use the partitioning technique from f_3 to f_{20} because it is more efficient when we consider the key recovery algorithmic complexity. Puncturing is applied to f_1 and f_2.

Table 3: Trail enumeration for $f_2 = (-1)^{v_0^{5.5}[8]}$

round	active	#coeffs	ρ^2
6	v_0^6[8,7-3], v_5^6[15,14-10], v_{10}^6[8,7-3]	94	$2^{-0.045}$
	$\bar{z}_0$[8,7-3], v_5^6[15,14-10] $\leftarrow \{\bar{z}_5$[15,14-10], k_5[15,14]$\}$,		
end1	v_{10}^6[8,7-3] $\leftarrow \{\bar{z}_{10}$[8,7-3], k_{10}[8]$\}$	1860	$2^{-2.74}$
end2	v_{10}^6[8,7-3] $\leftarrow \{\bar{z}_{10}$[8,7-3], k_{10}[8,7]$\}$	3720	$2^{-1.74}$
end3	v_{10}^6[8,7-3] $\leftarrow \{\bar{z}_{10}$[8,7-3], k_{10}[8,7,6]$\}$	8568	$2^{-1.38}$
end4	v_{10}^6[8,7-3] $\leftarrow \{\bar{z}_{10}$[8,7-3], k_{10}[8,7-5]$\}$	18416	$2^{-1.23}$

5.3 Punctured Map for Key Recovery

Punctured Map for f_2. We first discuss the exact map computing $v_0^{5.5}[8]$. It uses the known branches $\bar{z}_0$, $\bar{z}_5$, and $\bar{z}_{10}$, and the key branches k_5 and k_{10}.

$$(-1)^{v_0^{5.5}[8]} = f_2(\bar{z}_0[8\text{-}0], \bar{z}_5[15\text{-}0], \bar{z}_{10}[8\text{-}0], k_5[15\text{-}0], k_{10}[8\text{-}0]).$$

The domain size of f_2 is $9+16+9+16+9 = 59$ bits. It is practically impossible to construct the truth table of the Boolean function and compute its spectrum. Therefore, we use trail enumeration puncturing to construct a punctured Walsh spectrum. Table 3 summarizes the trails, where we use trail enumeration until before the key addition and consider 4 cases depending on the active key bits. Note that we do not need to guess $k_5[15]$ and $k_{10}[8]$ because they only change the sign of a correlation. In the end, if we guess $k_5[14]$ only, the puncturing correlation is $2^{-2.74}$ (end1 in the table). If we guess $k_5[14]$ and $k_{10}[7]$, the puncturing correlation is improved to $2^{-1.74}$ (end2). In other words, a 1-bit guess reduces the data complexity by $1/4$ in the final key recovery. If we also guess $k_{10}[6]$, the puncturing correlation is improved to $2^{-1.38}$, reducing the data complexity by $2^{-0.71}$ (end3). If we also guess $k_{10}[5]$, the puncturing correlation is $2^{-1.23}$ (end4).

Which key bits are guessed is decided case by case. We guess $k_{10}[6,5]$ because they help evaluate f_1 and f_{19}. We do not guess $k_5[13]$ because it is not used in the other functions. Finally, we use the punctured map with correlation $2^{-1.23}$:

$$f_2 \approx (-1)^{\bar{z}_0[8]\oplus\bar{z}_5[15]\oplus k_5[15]\oplus\bar{z}_{10}[8]\oplus k_{10}[8]} \times g_2(\bar{z}_0[7\text{-}3], \bar{z}_5[14\text{-}10], \bar{z}_{10}[7\text{-}3], k_5[14], k_{10}[7\text{-}5]),$$

The dimension of g_2 is at most $5+5+5+1+3 = 19$, so the cost of finding its (real-valued) truth table is 19×2^{19}, which is practical.

Punctured Map for f_1 Similarly to $v_0^{5.5}[8]$, we estimate the Walsh spectrum through trail enumeration. Table 4 summarizes the trails of f_1, and we have the following punctured map with puncturing correlation $2^{-2.31}$:

$$f_1 \approx (-1)^{\bar{z}''_{10}[7]\oplus\bar{z}'_{14}[23]\oplus\bar{z}_3[7]\oplus\bar{z}_4[14]\oplus\bar{z}_9[7]\oplus k_4[14]\oplus k_9[7]\oplus k_{10}[7]} \times g_1(\bar{z}''_{10}[6\text{-}2], \bar{z}'_{14}[22\text{-}18], \bar{z}_3[6\text{-}2], \bar{z}_4[13\text{-}9], \bar{z}_9[6\text{-}2], k_4[13], k_9[6], k_{10}[6,5]).$$

Table 4: Trail enumeration for $f_1 = (-1)^{v_{10}^{4.5}[7]}$

round	active	#coeffs	ρ^2
5	v_{10}^{5}[7,6-0], v_{14}^{5}[7,6-2]	158	$2^{-0.029}$
5.5	v_{10}^{5}[7,6-0], $v_{3}^{5.5}$[7,6-2], $v_{14}^{5.5}$[23,22-18]	158	$2^{-0.029}$
6	v_{10}^{5}[7,6-0], $v_{14}^{5.5}$[23,22-18], v_{3}^{6}[7,6-2], v_{4}^{6}[14,13-9], v_{9}^{6}[7,6-2]	24248	$2^{-0.15}$
end	$\bar{z}'_{14}$[23,22-18], $\bar{z}_3$[7,6-2], v_{10}^{5}[7,6-0] $\leftarrow \{\bar{z}''_{10}$[7,6-2], k_{10}[7,6,5]$\}$, v_{4}^{6}[14,13-9] $\leftarrow \{\bar{z}_4$[14,13-9], k_4[14,13]$\}$, v_{9}^{6}[7,6-2] $\leftarrow \{\bar{z}_9$[7,6-2], k_9[7,6]$\}$	3699840	$2^{-2.31}$

The dimension of g_1 is at most $5+5+5+5+5+1+1+2 = 29$, so the cost of computing truth table is 29×2^{29}, which is also practical.

5.4 Simple Attack

The correlation of the differential-linear distinguisher is $2^{-8.3}$. When we exploit N' pairs, we suppose the empirical correlation follows $\mathcal{N}(2^{-8.3}, 1/N')$ for the right guess, and $\mathcal{N}(0, 1/N')$ for the wrong guesses. Let ρ^2 be the product of all puncturing correlations from f_1 to f_{20}. According to Proposition 2, the required data complexity, denoted by N, is $N = N' \times \rho^{-4}$.

We use the partitioning technique from f_3 to f_{20}. We guess the 2-bit key for every decomposition. Since some key guesses are overlapped, e.g., f_{13} and f_{17}, guessing the following 30 bits is enough.

$$k_4[18,17,6,5], k_5[25,24,6,5], k_6[18,17,12,11,6,5], k_7[6,5],$$
$$k_9[11,10], k_{10}[30,29,18,17,6,5], k_{11}[25,24,11,10,5,4].$$

The partitioning technique allows us to compute the parity with probability $(3/4)^{18} \approx 2^{-7.47}$. To evaluate the punctured map for f_1 and f_2, we additionally guess $k_5[14]$, $k_{10}[7]$, $k_4[13]$, and $k_9[6]$ with puncturing correlation $2^{-1.23} \times 2^{-2.31} = 2^{-3.54}$. In total, we guess 34 key bits and $\rho^2 = 2^{-7.47-3.54} = 2^{-11.01}$. If $N' = 2^{23}$, the success probability with 34-bit advantage is 99.59%. As a result, we need

$$N = N' \times 2^{11.01 \times 2} = 2^{45.02}$$

pairs. In the partitioning technique, the available pairs are immediately determined depending on the key guess. Thus, the time complexity is estimated as

$$2N \times 2^{-7.47 \times 2} \times 2^{34} = 2^{65.08}.$$

We need to repeat this procedure 2^5 times to satisfy the differential characteristic in the 1st round. Therefore, the data complexity is $2^{45.02+5+1} = 2^{51.02}$ keystream blocks and the time complexity is $2^{65.08+5} = 2^{70.08}$.

5.5 Improved Attack Using a Distillation Table

The distillation table is a well-known technique to accelerate linear attack key recovery [Mat94]. The information from the keystream pairs is stored in a table and guess the key against the distillation table. When the size of this distillation table is lower than the number of pairs, we can reduce the time complexity. To the best of our knowledge, this technique has not been applied to ChaCha.

The keystream block is large (512 bits), which in principle makes the distillation table huge. However, we can combine the existing attack procedure with the distillation table one. In the first phase, we first evaluate some functions in the decomposition and reduce the number of involved keystream bits after the evaluation. Once the size is small enough, we go to the distillation phase.

Phase 1. Two punctured functions for f_1 and f_2 involve many bits of $\bar{Z}$ compared to the amount of the key guess. Since they are not suitable to the distillation table, we evaluate them in the first phase. The partitioning evaluation for $f_{19} = (-1)^{v_{10}^{5,5}[7]}$ shares the same guess $k_{10}[6,5]$. The key $k_9[11,10]$ is used in $f_{11} = (-1)^{v_9^6[12]}$ and $f_{16} = (-1)^{v_9^5[12]}$. The key $k_{10}[18,17]$ is used in $f_{13} = (-1)^{v_{10}^6[19]}$, $f_{17} = (-1)^{v_{10}^5[19]}$, and $f_{18} = (-1)^{v_{10}^{5,5}[19]}$. These six functions are evaluated in the first phase because the required size of the distillation table cannot be reduced even if they share the same guess. To optimize the attack, we further evaluate four functions, e.g., f_3, f_4, f_5, and f_6 in the first phase.

In summary, we guess 18 key bits and partitioning can compute the parity for $(3/4)^{10} \approx 2^{-4.15}$ pairs. The time complexity of this phase is estimated as

$$2N \times 2^{-4.15\times 2} \times 2^{18} \approx 2^{55.72}.$$

Phase 2. We still need to evaluate the following eight functions: f_7, f_8, f_9, f_{10}, f_{12}, f_{14}, f_{15}, and f_{20}. When we use partitioning with a 2-bit tail, each function involves 2 bits of $\bar{Z}$ and 2 bits of $\bar{Z}'$. Therefore, a distillation table with a dimension of $(2+2)\times 8 + 18 = 50$ is enough. From the output of phase 1, we construct the distillation table with complexity $N \times 2^{-4.15\times 2} \times 2^{18} \approx 2^{54.72}$. Then, we evaluate each f_i step-by-step. For example, we first evaluate f_7 with a complexity of $2^{50} \times 2^2 = 2^{52}$ and reduce the size of the distillation table to $(2+2)\times 7 + 20 = 48$. We next evaluate f_8 with cost of $2^{48} \times 2^2 = 2^{50}$ and reduce the size of the table to $(2+2)\times 6 + 22 = 46$. As a result, the time complexity of the second phase is estimated as

$$2^{54.72} + 2^{52} + 2^{50} + 2^{48} + 2^{46} + 2^{44} + 2^{42} + 2^{40} + 2^{38} \approx 2^{54.98}.$$

The time complexity of both phases is $2^{55.72+54.98} \approx 2^{56.40}$. We need to repeat this procedure 2^5 times, so the data complexity is $2^{51.02}$ and the time complexity is $2^{56.40+5} = 2^{61.40}$.

Data-time Trade-off. We can reduce the time complexity by using more data. Specifically, we use partitioning with a 1-bit tail instead of a 2-bit tail for f_3, f_4, f_5, and f_6. Then, the number of required pairs increases to

$$N = 2^{23} \times (3/4)^{14\times 2} \times (1/2)^{4\times 2} \times 2^{3.5377\times 2} \approx 2^{49.70}.$$

On the other hand, the time complexity for the 1st phase is

$$2 \times 2^{49.70} \times (3/4)^{6\times 2} \times (1/2)^{4\times 2} \times 2^{14} \approx 2^{51.72}.$$

The distillation table is has size $(2+2)\times 8 + 14 = 46$, so the second phase costs

$$2^{50.7160} + 2^{48} + 2^{46} + 2^{44} + 2^{42} + 2^{40} + 2^{38} + 2^{36} + 2^{34} \approx 2^{50.98}.$$

The time complexity of both phases is $2^{51.72} + 2^{50.98} \approx 2^{52.40}$. We need to repeat this procedure 2^5 times. Therefore, the data complexity is $2^{55.70}$ and the time complexity is $2^{57.40}$.

Toward Full Key Recovery. Our attack is a partial key recovery and does not recover the full 256-bit (or 128-bit) key. However, a full recovery should not be difficult because we have already recovered part of the key bits. For example, if we extend the key guess for each decomposition by 1 bit, we can recover those bits at a lower cost than the initial key recovery stage. We can also change the parameter i for the 3.5-round differential-linear distinguisher using additional data. Since some key guesses are overlapped even after changing i, the time complexity would be faster than the initial partial key recovery. We do not discuss how to recover the full key, but we expect a small additional cost.

5.6 Experimental Verification

To verify the complexity experimentally, we ran several experiments. We first verified the puncturing correlation for the product of the 20 functions, and the experiment almost matched our theoretical estimation, $2^{-11.01}$. Then, we verified the puncturing correlation for the differential-linear key recovery. The required data complexity is too large to observe the total correlation including the DL distinguisher. Therefore, we guess more key bits for f_3 to f_{20} to increase the puncturing correlation, thus reducing the required data to a practical range. In this case, the total correlation and distribution also follow the theoretical estimation. A detailed experimental report is shown in Appendix C.1.

6 Improved Attacks on 7-Round ChaCha

6.1 5-Round DL Distinguisher

Bellini et al. proposed a new differential-linear distinguisher that is optimized for 7-round attacks [BGG+23]. Similarly to [BLT20], five rounds are divided into

three sub ciphers, E_1 covering one round, E_m covering 2 rounds, and E_2 covering 2 rounds. Unlike the 3.5-round distinguisher, the the input difference has weight 2:

$$\Delta^0 = (\Delta^0_{15}[29], \Delta^0_{15}[19]) \xrightarrow{1R} \Delta^1 = (\Delta^1_3[25,5], \Delta^1_7[28,12], \Delta^1_{11}[25,21], \Delta^1_{15}[21,13]).$$

The probability that pairs with input difference Δ^0 satisfy this characteristic is 2^{-7} on average. For E_m, we have the following differential-linear distinguisher with autocorrelation $2^{-30.15}$:

$$\Delta^1 \xrightarrow{2R} \Gamma^3 = (\Gamma^3_2[4,3,0], \Gamma^3_7[20,4,0], \Gamma^3_8[20,19], \Gamma^3_{13}[4]).$$

This DL distinguisher is extended with a linear trail with 2^{-2} correlation.

$$\Delta^1 \xrightarrow{4R} \Gamma^5 = (\Gamma^5_2[0], \Gamma^5_6[19,7], \Gamma^5_{10}[12], \Gamma^5_{14}[0]).$$

As a result, Bellini et al. obtained the 4-round (from the 2nd round to the 5th round) differential-linear distinguisher with autocorrelation $2^{-34.15}$. Xu et al. analyzed the differential-linear hull of the 4-round distinguisher [XXTQ24], and the estimated autocorrelation was increased to $2^{-32.2}$.

We consider two cases. The first case appends a differential characteristic to the 4-round differential-linear distinguisher. We obtain the 5-round differential-linear distinguisher with autocorrelation $2^{-32.2-7} = 2^{-39.2}$. When we use this distinguisher, 2^{128} IVs are available. The second case excludes the differential characteristics like existing works [BLT20,XXTQ24]. To collect the right pairs satisfying $\Delta^0 \xrightarrow{1R} \Delta^1$ efficiently, at most 2^{96} pairs are available.

6.2 Linear Decomposition of the Key-Recovery Map

Like the 6-round attack, we decompose the key recovery map as a sum. Figure 5 shows the correlation 1 propagation, and the XOR of the blue bits is equal to $v^5_2[0] \oplus v^5_6[19] \oplus v^5_6[7] \oplus v^5_{10}[12] \oplus v^5_{14}[0]$. We aim to construct punctured for each blue bit. We handle some together to cancel out some active bits and reduce the key guess. As a result, we have 24 functions:

$$f_1 = (-1)^{v^5_{10}[12] \oplus v^7_7[19] \oplus v^7_{11}[12]}, \quad f_2 = (-1)^{v^{5.5}_{11}[19] \oplus v^6_{11}[19] \oplus v^7_4[26] \oplus v^7_8[19]},$$
$$f_3 = (-1)^{v^{5.5}_{11}[7]}, \qquad f_4 = (-1)^{v^6_3[16]}, \qquad f_5 = (-1)^{v^{6.5}_2[24]},$$

which involve more than one modular subtraction, and

$$f_6 = (-1)^{v^7_5[7]}, \quad f_7 = (-1)^{v^7_6[25]}, \quad f_8 = (-1)^{v^7_6[19]}, \quad f_9 = (-1)^{v^7_6[13]},$$
$$f_{10} = (-1)^{v^7_6[7]}, \quad f_{11} = (-1)^{v^7_7[26]}, \quad f_{12} = (-1)^{v^7_7[7] \oplus v^7_7[6]}, \quad f_{13} = (-1)^{v^7_{10}[18]},$$
$$f_{14} = (-1)^{v^7_{10}[12]}, \quad f_{15} = (-1)^{v^7_{10}[6]}, \quad f_{16} = (-1)^{v^7_{11}[31]}, \quad f_{17} = (-1)^{v^7_{11}[19]},$$
$$f_{18} = (-1)^{v^6_8[12]}, \quad f_{19} = (-1)^{v^6_{11}[31]}, \quad f_{20} = (-1)^{v^{6.5}_8[7]}, \quad f_{21} = (-1)^{v^{6.5}_{10}[26]},$$
$$f_{22} = (-1)^{v^{6.5}_{10}[6]}, \quad f_{23} = (-1)^{v^{6.5}_{11}[19]}, \quad f_{24} = (-1)^{v^{6.5}_{11}[7]},$$

which involve a single modular subtraction.

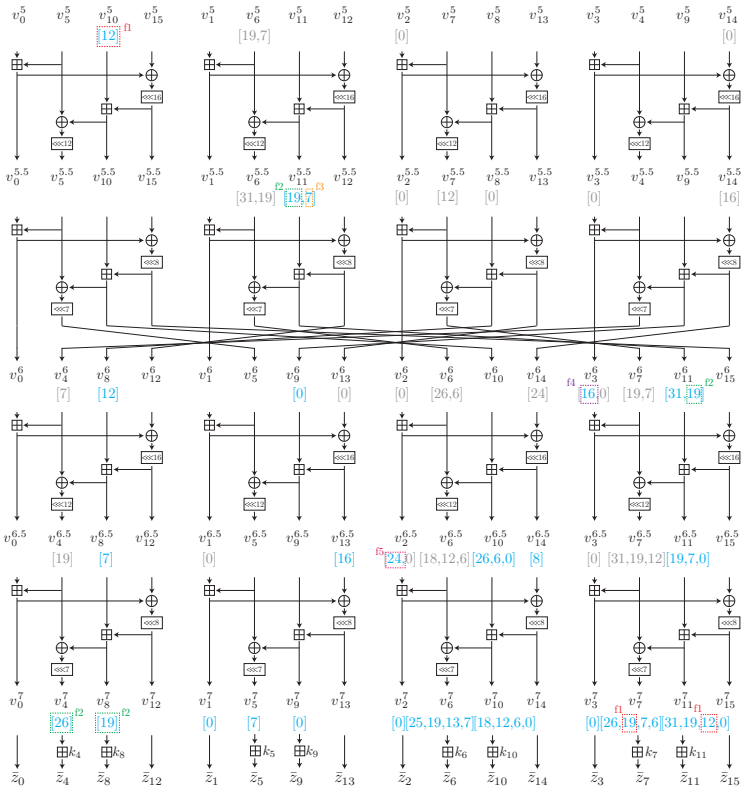


Fig. 5: Key recovery map for the 7-round attack.

6.3 Puncturing Each Function of the Decomposition

We use trail enumeration to obtain the punctured Walsh spectrum. Table 5 summarizes our punctured map.

For f_1, we evaluate $v_7^7[19]$ and $v_{11}^7[12]$ at the same time as $v_{10}^5[12]$ to cancel active bits. These cancellations reduce the size of key guesses to evaluate the punctured function. Table 6 in Appendix D shows the detail of the trail enumeration. Before the last key addition, we have a spectrum with 2088000 non-zero coefficients, and the puncturing correlation is $2^{-2.75}$. Many coefficients take very low values. We further puncture coefficients whose absolute value is less than 2^{-18} to accelerate the program. The number of remaining coefficients is 738352, and its puncturing correlation is still $2^{-2.75}$.

Further extension involving the last key addition is difficult in practice because of the huge memory and time complexities. We use Proposition 5 instead. It allows us to obtain the puncturing correlation without computing the table of the punctured Walsh spectrum in full. The puncturing correlation is $2^{-7.14}$, and the punctured Walsh spectrum fits in a table of size 2^{55+14}. In other words, we

Table 5: Punctured functions for the 7-round attack.

Target	Active	#coeffs	ρ^2
f_1	$\bar{z}_0$[12,11-9], $\bar{z}_3$[20,19-17, 12,11-9], $\bar{z}'_{15}$[28,27-25, 4,3-1], v_4^7[31,30-28, 19,18-16] $\leftarrow \{\bar{z}_4$[31,30-26, 19,18-14$], k_4$[31,30, 19,18]$\}$, v_7^7[27,26-24] $\leftarrow \{\bar{z}_7$[27,26-22$], k_7$[27,26,22]$\}$, v_8^7[24,23-21, 12,11-9] $\leftarrow \{\bar{z}_8$[24,23-19, 12,11-7$], k_8$[24,23, 12,11,10]$\}$, v_{11}^7[20,19-17] $\leftarrow \{\bar{z}_{11}$[20,19-15$], k_{11}$[20,19-17,15]$\}$, $v_8^{6.5}$[12,11-9] $\leftarrow \{\bar{z}'_8$[12,11-7$], k'_8$[12,11,10]$\}$, v_{10}^6[12,11-9] $\leftarrow \{\bar{z}''_{10}$[12,11-7$], k''_{10}$[12,11]$\}$	$\leq 2^{55+14}$	$2^{-7.14}$
f_2	$\bar{z}_0$[19,18-14], $\bar{z}'_{12}$[3,2-0,31,30], v_{11}^6[18-14] $\leftarrow \{\bar{z}''_{11}$[18-13$], k''_{11}$[18,17,15,14]$\}$	1740	$2^{-2.39}$
f_3	$\bar{z}_0$[7,6-2], $\bar{z}'_{12}$[23,22-18], v_8^7[7,6-2] $\leftarrow \{\bar{z}_8$[7,6-2$], k_8$[7,6]$\}$, v_4^7[14,13-9] $\leftarrow \{\bar{z}_4$[14,13-9$], k_4$[14,13]$\}$, v_{11}^6[7,6-2] $\leftarrow \{\bar{z}''_{11}$[7,6-2$], k''_{11}$[7,6]$\}$	1570720	$2^{-2.57}$
f_4	$\bar{z}_3$[16,15-11], v_7^7[23,22-18, 3,2-0] $\leftarrow \{\bar{z}_7$[23,22-18, 3,2-0$], k_7$[23,22, 3,2]$\}$, v_{11}^7[28,27-25, 16,15-11] $\leftarrow \{\bar{z}_{11}$[28,27-23, 16,15-11$], k_{11}$[28,27, 16,15,14]$\}$, $v_{11}^{6.5}$[16,15-13] $\leftarrow \{\bar{z}'_{11}$[16,15-11$], k_{11}$[16,15,14]$\}$	$\leq 2^{28+7}$	$2^{-3.07}$
f_5	$\bar{z}_2$[24,23-19], v_6^7[31,30-26] $\leftarrow \{\bar{z}_6$[31,30-26$], k_6$[31,30]$\}$, v_{10}^7[24,23-19] $\leftarrow \{\bar{z}_{10}$[24,23-19$], k_{10}$[24,23]$\}$	2144	$2^{-1.74}$

need 2^{55+14} memory and $(55+14) \times 2^{55+14}$ time to construct the corresponding pseudoboolean function.

For f_2, active key bits can be removed through cancellation. A puncturing correlation of $2^{-2.39}$ is obtained guessing $k_{11}[18, 17, 15, 14]$. Table 7 in Appendix D summarizes the trail enumeration.

For the rest, there is no room for cancellation (see Tables 8, 9, and 10 in Appendix D for f_3, f_4, and f_5, respectively).

We choose which key bits to puncture using a greedy method. We first evaluate the puncturing correlation when all key bits are punctured. We then check the correlation when adding each key bit. If the correlation improves by at least $2^{0.5}$, we guess the key bit. We repeat this procedure until no more key bits cause such an improvement. As a result, the puncturing correlation for f_1 to f_5 is

$$2^{-7.14-2.39-2.57-3.07-1.74} \approx 2^{-16.91}.$$

For f_6 to f_{24}, we use partitioning with a 3-bit tail. Then, these parities are determined with probability $(7/8)^{19} \approx 2^{-3.66}$. Between the 1st and 2nd phases, we guess 60 bits in total.

6.4 Improved Attack Using Distillation Table

We first discuss the data complexity. The autocorrelation of the differential-linear distinguisher is $2^{-39.2}$. When $N' = 2^{85.5}$, the success probability with 60-bit advantage is 99.79%. As a result, we estimate N as

$$N = N' \times 2^{(16.91+3.66)\times 2} \approx 2^{126.65}$$

pairs. Thus, the data complexity is $2^{127.65}$.

Phase 1. Let N be the number of pairs. We first guess the following 20 bits involved in the punctured maps of f_1 to f_5, which are

$$\begin{aligned} &k_4[30, 18, 13], k_6[30], k_7[26, 22, 2], \\ &k_8[23, 11, 10, 6], k_{10}[23, 11], k_{11}[27, 19, 18, 17, 15, 14, 6]. \end{aligned} \tag{4}$$

Thus, the time complexity of this phase is $2N \times 2^{20} \approx 2^{147.65}$.

Phase 2. We still need to evaluate the functions, $f_6, \ldots, f_{24}$. For them, we use partitioning with a 3-bit tail, where the following 48 bits are used.

$$\begin{aligned} &k_5[6, 5, 4], k_6[24, 23, 22, 18, 17, 16, 12, 11, 10, 6, 5, 4], k_7[25, 24, 23, 6, 5, 4], \\ &k_8[11, 10, 9, 6, 5, 4], k_{10}[25, 24, 23, 17, 16, 15, 11, 10, 9, 5, 4, 3], \\ &k_{11}[30, 29, 28, 18, 17, 16, 6, 5, 4]. \end{aligned} \tag{5}$$

Note that some bits overlap with Equation (4), so we guess 60 bits in total.

We construct the distillation table by traversing $2^{146.65}$ pairs. The required dimension of the distillation table is $19 \times 3 \times 2 + 20 = 134$. We then evaluate each f_i step by step, e.g., we first guess the bit $k_{11}[16]$ to evaluate f_{17} and f_{23}. Then, the dimension of the distillation table decreases $17 \times 3 \times 2 + 21 = 123$. Because $N \times 2^{20} \gg 2^{134+1}$, the complexity of this step-by-step evaluation is negligible.

Time Complexity. The time complexity is

$$2^{147.65} + 2^{146.65} \approx 2^{148.23}.$$

Note that this attack partially, i.e., 60 bits, recovers the key. We need an extra analysis toward the full recovery, e.g., we can mount the attack procedure by describing the following subsection with the knowledge of the partially recovered key. Similar to the 6-round attack, we omit the detailed procedure because it should not be more difficult than the initial partial key recovery.

Experimental Verification. We have five component functions containing more than one modular addition. Two functions, f_1 and f_4, have a dimension beyond what we can practically manipulate on current hardware. However, we can verify the rest experimentally. The puncturing correlation for f_2, f_3, and f_5 is $2^{-6.70}$. There are 19 decomposed functions for which we apply partitioning with a 3-bit tail. Thus, the total puncturing correlation is $2^{-6.70} \times (7/8)^{19} \approx 2^{-10.36}$. We experimentally verified that the experiment almost matched our theoretical estimation. For more details, see Appendix C.2.

6.5 Reducing the Data Complexity

Since existing attacks have lower data complexity, we consider a trade-off where the data complexity is competitive with previous works. We use the trick in [BLT20] to improve the autocorrelation of the DL distinguisher. Then, at most 2^{96} pairs are available for the main attack procedure.

In the former attack, the distillation table catalogues 114 keystream bits. Since $2^{114} > 2^{96}$, such a distillation table is now inefficient. Therefore, we pick the four functions f_{17}, f_{18}, f_{20}, and f_{23}, and evaluate them in the 1st phase instead of the 2nd phase. For the attack to succeed with at most 2^{96} pairs, we need to guess more key bits to improve the puncturing correlation. Thus, we change the threshold of the greedy selection method from $2^{0.5}$ to $2^{0.15}$.

Table 11 in Appendix E summarizes the punctured map. As a result, we need to guess 51 secret key bits. Partitioning is used on $f_{17} = (-1)^{v_{11}^{7}[19]}$ and $f_{23} = (-1)^{v_{11}^{6.5}[19]}$ with a 7-bit tail, $f_{18} = (-1)^{v_{8}^{6}[12]}$ with a 9-bit tail, and $f_{20} = (-1)^{v_{8}^{6.5}[7]}$ with a 4-bit tail. Then, these parities are determined with probability $\left(\frac{15}{16}\right) \times \left(\frac{511}{512}\right) \times \left(\frac{127}{128}\right)^2 \approx 2^{-0.12}$. The puncturing correlation in the 1st phase is about $2^{-8.54}$. The 2nd phase uses a 3-bit tail for 15 functions. Thus, the correlation is $(7/8)^{15} \approx 2^{-2.89}$. Between the 1st and 2nd phases, we guess 83 bits.

The autocorrelation of the distinguisher is $2^{-32.2}$. If $N' = 2^{72}$, the success probability with 83-bit advantage is 99.97%. Thus,

$$N = N' \times 2^{(8.54+2.89)\times 2} \approx 2^{94.85}$$

pairs. Thus, the data complexity is $2^{95.85}$, which is less than 2^{96}. The time complexity is

$$2N \times 2^{-0.12\times 2} \times 2^{51} + N \times 2^{-0.12\times 2} \times 2^{51} \approx 2^{147.20},$$

as the cost of the second phase is negligible. Due to the trick from [BLT20], we repeat this procedure 2^7 times, and the final data and time complexities are $2^{102.85}$ and $2^{154.20}$, respectively.

7 Improved Attack on 7.5-Round ChaCha

Recently, Dey proposed the first 7.5-round attack, with success probability 79% [Dey24] and time complexity $2^{255.2}$, which is very close to exhaustive search. We propose the first 7.5-round attack which is meaningfully better than exhaustive search.

The attack strategy is almost the same as the 7-round attack. We use the 5-round DL distinguisher with autocorrelation $2^{-39.2}$ and construct punctured functions for each component of the key-recovery map. The main difficulty is the data complexity. Using trail enumeration, the trails become very long and thus have small correlation. The number of coefficients required to obtain a high puncturing correlation increases exponentially. To increase the puncturing correlation and make the trail enumeration practical, we guess the full 32-bit k_4

and 20 bits k_9[19-0]. Such a guess allows us to compute v_3^7[19-0], v_4^7[19-0], and the carry toward v_3^7[20]. Therefore, we do not need to extend the trail further once it reaches these bits.

We refer to Appendix F for further details. Briefly, the puncturing correlation is $2^{-20.39}$ and $2^{-1.40}$ in Phases 1 and 2, respectively, or $\rho^2 \approx 2^{-21.79}$ in total. We guess 122 key bits in Phase 1 and 42 additional bits in Phase 2. The data complexity is $2^{127.08}$ and the time complexity is $2^{250.17}$.

8 Conclusion

Since 2008, PNBs have been the leading cryptanalysis technique for ChaCha, and most of the literature relies on novel ideas within the framework of PNBs. Due to the complicated analysis of ARX, it was difficult to demonstrate theoretical analysis techniques which beat the experimental analysis of PNBs. In this paper, we have proposed a theoretical analysis using bit puncturing against ChaCha. Bit puncturing does not rely on blackbox experiments like PNBs, and derives the theoretically optimal key recovery. This is the first successful theoretical analysis to significantly outperform PNBs. As a result, we improved the 6 and 7-round attacks and proposed the first 7.5-round attack with a significant advantage over exhaustive search.

There are a multitude of questions which remain open to future work. First, a better method for evaluating the puncturing correlation would be very helpful from both the theoretical and the practical perspectives. Our puncturing technique is based on trail enumeration. It relies on running a large number of computations, like the search for differential or linear trails. As the number of target rounds grows, it requires increasingly more memory and time, and the correlation which can be obtained diminishes. A better understanding of the composition of punctured maps may lead to simpler, more effective and more efficient algorithms. Second, we used a greedy method to determine which key bits to guess with and without a distillation table, based on heuristic criteria. The optimal methodology is still open. Furthermore, the possibility of applying key recovery techniques based on the FFT [CSQ07] to ChaCha remains an open problem.

References

AFK+08. Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of salsa, chacha, and rumba. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 470–488. Springer, 2008.

BBC+22. Christof Beierle, Marek Broll, Federico Canale, Nicolas David, Antonio Flórez-Gutiérrez, Gregor Leander, María Naya-Plasencia, and Yosuke Todo. Improved differential-linear attacks with applications to ARX ciphers. *J. Cryptol.*, 35(4):29, 2022.

BC14. Eli Biham and Yaniv Carmeli. An improvement of linear cryptanalysis with addition operations with applications to FEAL-8X. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 59–76. Springer, 2014.

Ber08. Daniel J Bernstein. ChaCha, a variant of Salsa20, 2008.

BGG+23. Emanuele Bellini, David Gérault, Juan Grados, Rusydi H. Makarim, and Thomas Peyrin. Boosting differential-linear cryptanalysis of chacha7 with MILP. *IACR Trans. Symmetric Cryptol.*, 2023(2):189–223, 2023.

BLT20. Christof Beierle, Gregor Leander, and Yosuke Todo. Improved differential-linear attacks with applications to ARX ciphers. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020.

CM16. Arka Rai Choudhuri and Subhamoy Maitra. Significantly improved multi-bit differentials for reduced round salsa and chacha. *IACR Trans. Symmetric Cryptol.*, 2016(2):261–287, 2016.

CN20. Murilo Coutinho and T. C. Souza Neto. New multi-bit differentials to improve attacks against chacha. *IACR Cryptol. ePrint Arch.*, page 350, 2020.

CSQ07. Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the time complexity of matsui's linear cryptanalysis. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology - ICISC 2007, 10th International Conference, Seoul, Korea, November 29-30, 2007, Proceedings*, volume 4817 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2007.

Dey24. Sabyasachi Dey. Advancing the idea of probabilistic neutral bits: First key recovery attack on 7.5 round chacha. *IEEE Trans. Inf. Theory*, 70(8):6091–6106, 2024.

DGSS22. Sabyasachi Dey, Hirendra Kumar Garai, Santanu Sarkar, and Nitin Kumar Sharma. Revamped differential-linear cryptanalysis on reduced round chacha. In Orr Dunkelman and Stefan Dziembowski, editors, *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*, volume 13277 of *Lecture Notes in Computer Science*, pages 86–114. Springer, 2022.

DGSS23. Sabyasachi Dey, Hirendra Kumar Garai, Santanu Sarkar, and Nitin Kumar Sharma. Enhanced differential-linear attacks on reduced round chacha. *IEEE Trans. Inf. Theory*, 69(8):5318–5336, 2023.

FT24. Antonio Flórez-Gutiérrez and Yosuke Todo. Improving linear key recovery attacks using walsh spectrum puncturing. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 187–216. Springer, 2024.

Leu16. Gaëtan Leurent. Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In Marc Fischlin and Jean-Sébastien Coron, editors, *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, volume 9665 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2016.

LH94. Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 1994.

Mat94. Mitsuru Matsui. The first experimental cryptanalysis of the data encryption standard. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.

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

WLHL23. Shichang Wang, Meicheng Liu, Shiqi Hou, and Dongdai Lin. Moving a step of chacha in syncopated rhythm. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 273–304. Springer, 2023.

XXTQ24. Zhichao Xu, Hong Xu, Lin Tan, and Wenfeng Qi. Differential-linear cryptanalysis of reduced round chacha. *IACR Trans. Symmetric Cryptol.*, 2024(2):166–189, 2024.

A Detailed Proofs

A.1 Proof of Proposition 4

Proposition 4 *Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the modular addition operation $z = f(x, y) = x \boxplus y$. Let $n \geq l \geq r$ be two natural numbers, and $d = l - r$. Consider an output mask $w \in \mathbb{F}_2^n$ satisfying $l \geq \overrightarrow{w}$. Let $g_w : \mathbb{F}_2^n \longrightarrow \mathbb{R}^n$ be a bit puncturing of f_w defined as follows:*

$$\widehat{g_w}(u, v) = \begin{cases} \widehat{f}(u, v, w) & \text{if } \overleftarrow{w} \geq \overleftarrow{u}, \overleftarrow{v} \text{ and } \overrightarrow{u}, \overrightarrow{v} \geq r \\ 0 & \text{otherwise} \end{cases}.$$

Then $1 - \epsilon \geq 1 - \left(2^{-d} - 2^{-l-r}\right)$.

Proof. Let $\mathcal{S} = \{(u, v) \in \mathbb{F}_2^{2n} : \overrightarrow{u}, \overrightarrow{v} \geq r\}$ be the set of input masks whose associated coefficients are copied from $\widehat{f_w}$ to $\widehat{g_w}$. Our aim is to compute the sum $1 - \epsilon = \sum_{(u,v)\in\mathcal{S}} \widehat{f}(u, v, w)^2$. Our approach is to divide this sum into several smaller sums corresponding to different headers for the masks u and v, which will be denoted by $u^0, v^0 \in \mathbb{F}_2^{n-l}$. Let

$$\mathcal{T}(u^0, v^0) = \{(u, v) \in \mathbb{F}_2^{2n} : u_{l+i} = u_i^0, v_{l+i} = v_i^0 \text{ for all } 0 \leq i < n - l\},$$
$$\mathcal{S}(u^0, v^0) = \mathcal{T}(u^0, v^0) \cap \mathcal{S}.$$

Then, it is clear that

$$\sum_{(u,v)\in\mathcal{S}} \widehat{f}(u, v, w)^2 = \sum_{u^0,v^0\in\mathbb{F}_2^{n-l}} \sum_{(u,v)\in\mathcal{S}(u^0,v^0)} \widehat{f}(u, v, w)^2. \tag{6}$$

We also note that

$$\sum_{u^0,v^0\in\mathbb{F}_2^{n-l}} \sum_{(u,v)\in\mathcal{T}(u^0,v^0)} \widehat{f}(u, v, w)^2 = \sum_{u,v\in\mathbb{F}_2^n} \widehat{f}(u, v, w)^2 = 1. \tag{7}$$

We focus on each of the partial sums. In particular, we consider fixed $u^0, v^0 \in \mathbb{F}_2^{n-l}$, and we assume they are compatible (that is, there are nonzero $\widehat{f}(u, v, w)$ with $(u, v) \in \mathcal{T}(u^0, v^0)$). In particular, this implies that $\overleftarrow{u^0}, \overleftarrow{v^0} \leq \overleftarrow{w} - l$. Indeed, without loss of generality, if $\overleftarrow{u} > \overleftarrow{w}$ and $\overleftarrow{u} \geq \overleftarrow{v}$, then $\sigma_{\overleftarrow{u}} = 0$ and $u_{\overleftarrow{u}} = w_{\overleftarrow{u}}$, which contradicts the assumption.

We want to compare the sum over $\mathcal{S}(u^0, v^0)$ to the sum over $\mathcal{T}(u^0, v^0)$. We will enumerate all the nonzero terms of the latter, and see which ones are also in the former. We note that σ_{n-1} to σ_{l-1} are determined by w, u^0 and v^0. Let $\sigma^0 = (\sigma_{n-1}, \ldots, \sigma_l)$, and $c_0 = 2^{-\mathrm{wt}(\sigma^0)}$.

Let $(u, v) \in \mathcal{T}(u^0, v^0)$ with $\widehat{f}(u, v, w) \neq 0$. We consider the form of σ. We note that given $i < l$, if $\sigma_i = 0$, then $u_i = v_i = w_i = 0$, which means that $\gamma_i = 0$ and in consequence $\sigma_{i-1} = 0$. Reasoning by induction, it is clear that $\sigma_j = 0$ and $u_j = u_j = 0$ for all $j < i$. This means that, from σ_{l-1} onwards, σ must take the

form $(1, \ldots, 1, 0, \ldots 0)$. In particular, it holds that $\widehat{f}(u, v, w) = \pm 2^{\min\{\overrightarrow{\sigma}-l,0\}} c_0$, where the minimum is considered to accommodate the case $\sigma_{l-1} = 0$. We assume $\sigma_{l-1} = 1$, and the other case will be treated separately. We must count how many nonzero terms exist in the sum for a given value of $\overrightarrow{\sigma}$. We note that for each $\sigma_i = 1$, we have four possible choices of u_i and v_i, two of which lead to $\sigma_{i-1} = 0$ and two of which lead to $\sigma_{i-1} = 1$. From this observation, we can deduce that the number of coefficients with value $\pm 2^{\overrightarrow{\sigma}-l} c_0$ is $2^{l-\overrightarrow{\sigma}}$. We must also consider the separate case that $\sigma_{-1} = 1$, which corresponds to 2^l terms with $\widehat{f}(u, v, w) = \pm 2^{-l} c_0$. In summary,

$$\sum_{(u,v)\in\mathcal{T}(u^0,v^0)} \widehat{f}(u,v,w)^2 = \sum_{i=1}^{l} 2^i \cdot 2^{-2i} \cdot c_0^2 + 2^l \cdot 2^{-2l} \cdot c_0^2$$
$$= c_0^2 \left(\sum_{i=1}^{l} 2^{-i} + 2^{-l} \right) = c_0^2 (1 - 2^{-l} + 2^{-l}) = c_0^2.$$

We next consider which of these terms also appear in $\mathcal{S}(u^0, v^0)$, once again separating according to $\overrightarrow{\sigma}$. If $\overrightarrow{\sigma} \geq r$, then $u_j = v_j = 0$ for all $j < r$, and $(u, v) \in \mathcal{S}(u^0, v^0)$. On the other hand, if $\overrightarrow{\sigma} < r$ and $\sigma_{-1} = 0$, necessarily $\gamma_{\overrightarrow{\sigma}} = u_{\overrightarrow{\sigma}} \oplus v_{\overrightarrow{\sigma}} = 1$, and $(u, v) \notin \mathcal{S}(u^0, v^0)$. Finally, if $\sigma_{-1} = 1$, we find that there are two choices for u_{l-i} and v_{l-i} for $i = 1$ until $i = l - r = d$, and from then on $u_i = v_i = 0$. As a result, there are $2^{l-r} = 2^d$ such terms in the sum.

$$\sum_{(u,v)\in\mathcal{S}(u^0,v^0)} \widehat{f}(u,v,w)^2 = \sum_{i=1}^{l-r} 2^i \cdot 2^{-2i} \cdot c_0^2 + 2^{l-r} \cdot 2^{-2l} \cdot c_0^2 =$$
$$= c_0^2 \left(\sum_{i=1}^{d} 2^{-i} + 2^{-l-r} \right) = c_0^2 (1 - 2^{-d} + 2^{-l-r}).$$

We still have to consider the case $\sigma_{l-1} = 0$. This case is actually pretty simple, as $\sigma_i = 0$ for all $i < l$, and therefore $u_i = v_i = w_i = 0$, and there is a single nonzero coefficient in $\mathcal{T}(u^0, v^0)$ which is also in $\mathcal{S}(u^0, v^0)$. We thus have

$$\sum_{(u,v)\in\mathcal{T}(u^0,v^0)} \widehat{f}(u,v,w)^2 = \sum_{(u,v)\in\mathcal{S}(u^0,v^0)} \widehat{f}(u,v,w)^2 = c_0^2.$$

We return to the full sum. We note that for all $u^0, v^0 \in \mathbb{F}_2^{n-l}$, we have

$$\sum_{(u,v)\in\mathcal{S}(u^0,v^0)} \widehat{f}(u,v,w)^2 \geq (1 - 2^{-d} + 2^{-l-r}) \sum_{(u,v)\in\mathcal{T}(u^0,v^0)} \widehat{f}(u,v,w)^2$$

We conclude by substituting in Equation 6 and considering Equation 7. □

We note that the bound is if the case $\sigma_{l-1} = 0$ never occurs, which is true for some w. Finding a bound which accounts for the form of w in more detail and is thus always tight remains an open problem.

A.2 Proof of Proposition 5

Proposition 5 *Let $f_1 : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $f_2 : \mathbb{F}_2^m \to \mathbb{F}_2$ be two Boolean functions, and let $f = f_2 \circ f_1$ be their composition. Let $g_2 : \mathbb{F}_2^m \to \mathbb{R}$ be an approximation of f_2 by puncturing Wlash spectrum coefficients. Furthermore, let $g_{1,u} : \mathbb{F}_2^n \to \mathbb{R}$ be an approximation of $\langle u, f_1 \rangle$ by puncturing Walsh spectrum coefficients, where the puncture set is the same for any u. We define g as the matrix-vector product as $\widehat{g} = \widehat{g_1} \times \widehat{g_2}$. Then, we have*

$$\rho^2 \approx \sum_{u \in \mathbb{F}_2^n} (\widehat{g}(u))^2 = 2^n \times \sum_{v \in \mathbb{F}_2^m} \sum_{w \in \mathbb{F}_2^m} \widehat{g_2}(v)\widehat{g_2}(w)\langle \widehat{g_{1,v}}, \widehat{g_{1,w}} \rangle. \tag{8}$$

Proof. We suppose that we have the punctured Walsh spectrum $\widehat{g_2}$. We now extend $\widehat{g_2}$ further to include the function f_1.

$$\begin{aligned}
\sum_{u \in \mathbb{F}_2^n} (\widehat{g}(u))^2 &= \sum_{u \in \mathbb{F}_2^n} \widehat{g}(u)\widehat{g}(u) \\
&= \sum_{u \in \mathbb{F}_2^n} \left(\sum_{v \in \mathbb{F}_2^m} \widehat{g_2}(v)\widehat{g_{1,v}}(u) \right) \left(\sum_{w \in \mathbb{F}_2^m} \widehat{g_2}(w)\widehat{g_{1,w}}(u) \right) \\
&= \sum_{v \in \mathbb{F}_2^m} \sum_{w \in \mathbb{F}_2^m} \widehat{g_2}(v)\widehat{g_2}(w) \sum_{u \in \mathbb{F}_2^n} \widehat{g_{1,v}}(u)\widehat{g_{1,w}}(u) \\
&= 2^n \times \sum_{v \in \mathbb{F}_2^m} \sum_{w \in \mathbb{F}_2^m} \widehat{g_2}(v)\widehat{g_2}(w)\langle \widehat{g_{1,v}}, \widehat{g_{1,w}} \rangle
\end{aligned}$$

B Extended Puncturing for the Modular Addition

First, we consider the following scenario, which corresponds to situations in which we only need to restrict the nonzero mask bits for one of the input branches. A slightly better puncturing correlation is obtained:

Proposition 6. *Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the modular addition operation $z = f(x, y) = x \boxplus y$. Let $n \geq l \geq r$ be two natural numbers, and $d = l - r$. Consider an output mask $w \in \mathbb{F}_2^n$ satisfying $l \geq \overrightarrow{w}$. Let $g_w : \mathbb{F}_2^n \longrightarrow \mathbb{R}^n$ be a bit puncturing of f_w defined as follows:*

$$\widehat{g}(u, v, w) = \begin{cases} \widehat{f}(u, v, w) & \text{if } \overleftarrow{u}, \overleftarrow{v} \leq \overleftarrow{w} \text{ and } \overrightarrow{v} \geq r \\ 0 & \text{otherwise} \end{cases}.$$

Then $1 - \epsilon \geq 1 - \frac{2}{3}\left(2^{-d} - 2^{-l-r}\right)$.

Proof. The proof follows the same scheme as for Proposition 4, but with $\mathcal{S} = \{(u, v) \in \mathbb{F}_2^{2n} : \overrightarrow{v} \geq r\}$. The sets $\mathcal{S}(u^0, v^0)$ are larger. Let $(u, v) \in \mathcal{T}(u^0, v^0)$. If $\overrightarrow{\sigma} \geq r$, we know that $(u, v) \in \mathcal{S}(u^0, v^0)$. If $\overrightarrow{\sigma} < r$ and $\sigma_{-1} = 0$, then $u_{\overrightarrow{\sigma}} \oplus v_{\overrightarrow{\sigma}} = 1$. In the case $v_{\overrightarrow{\sigma}} = 1$ we have $(u, v) \notin \mathcal{S}(u^0, v^0)$. However, if $u_{\overrightarrow{\sigma}} = 1$, there are 2^d

such terms which are in $\mathcal{S}(u^0, v^0)$. For $\sigma_{-1} = 1$, we keep the same 2^d terms as in Proposition 4. In summary, the partial sums now look like this:

$$\begin{aligned}\sum_{(u,v)\in\mathcal{S}(u^0,v^0)} \widehat{f}(u,v,w)^2 &= c_0^2 \left(\sum_{i=1}^{d} 2^i \cdot 2^{-2i} + \sum_{i=d+1}^{l} 2^d \cdot 2^{-2i} + 2^d \cdot 2^{-2l} \right) = \\ &= c_0^2 \left(1 - 2^{-d} + 2^d \cdot \frac{2^{-2d-2} - 2^{-2l-2}}{1 - 2^{-2}} + 2^{-l-r} \right) \\ &= c_0^2 \left(1 - \frac{2}{3} \left(2^{-d} - 2^{-l-r} \right) \right).\end{aligned}$$

The proof concludes in the same way as Proposition 4. □

Finally, we consider another possibility, which corresponds to ignoring the last r bits of the input branches and simply computing the modular addition over a smaller number of bits, that is, taking $g = f^0$. This also corresponds to setting the last r bits to zero (as would be done for PNBs). Although this situation is no longer puncturing, we can still compute the correlation of f and g, and obtain a result which is slightly worse than puncturing.

Proposition 7. *Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the modular addition operation $z = f(x, y) = x \boxplus y$. Let $r \leq l \leq n$ be two natural numbers, and $d = l - r$. Let $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a function obtained by setting the r rightmost bits of x and y to zero and computing the modular addition. Let $w \in \mathbb{F}_2^n$ be an output mask with $\overrightarrow{w} \geq l$. Then g_w approximates f_w and $\rho^2 = \langle f_w, g_w \rangle^2 \geq 1 - 2^{-d} - 2^{-l} + 2^{-2d} + 2^{-d-l} + 2^{-2l}$.*

Proof. Once again the proof is similar to Proposition 4. We note that the spectrum of g can be obtained from the spectrum of the modular addition on 2^{n-r}, and that $\|g_w\|_2 = 1$ because it's a Boolean function. It thus suffices to compute $\langle \widehat{f_w}, \widehat{g_w} \rangle = \sum_{u,v\in\mathbb{F}_2^n} \widehat{f}(u,v,w)\widehat{g}(u,v,w)$. We define the headers u^0, v^0, c_0 and the sets $\mathcal{T}(u^0, v^0)$ in the same way as in Proposition 4. It is clear that

$$\langle f_w, g_w \rangle = \sum_{u^0,v^0\in\mathbb{F}_2^{n-l}} \sum_{(u,v)\in\mathcal{T}(u^0,v^0)} \widehat{f}(u,v,w)\widehat{g}(u,v,w).$$

We already know the structure of $\mathcal{T}(u^0, v^0)$ from the proof of Proposition 4. We separate the different values of $\overrightarrow{\sigma}$:

- If $\overrightarrow{\sigma} \geq r$, then $\widehat{g}(u,v,w) = \widehat{f}(u,v,w) = \pm 2^{-l+\overrightarrow{\sigma}} c_0$.
- If $\overrightarrow{\sigma} < r$ and $\sigma_{-1} = 0$, then $\widehat{g}(u,v,w) = 0$.
- If $\sigma_{-1} = 1$, there are 2^d terms for which $\widehat{g}(u,v,w) = \pm 2^{-d} c_0 = \pm 2^r \widehat{f}(u,v,w)$. For the rest, $\widehat{g}(u,v,w) = 0$.

This enables us to compute $\langle f_w, g_w \rangle$:

$$\langle f_w, g_w \rangle \geq \sum_{u^0, v^0 \in \mathbb{F}_2^{n-l}} c_0^2 \left(\sum_{i=1}^{d} 2^{-i} \cdot 2^{-2i} - 2^d \cdot 2^{-l} \cdot 2^{-d} \right) =$$
$$= \sum_{u^0, v^0 \in \mathbb{F}_2^{n-l}} c_0^2 (1 - 2^{-d} - 2^{-l}) \geq 1 - 2^{-d} - 2^{-l}.$$

By squaring this bound we conclude the proof. □

C Experimental Verification

C.1 Several Experiments for 6-Round Attack

To verify the correctness of our complexity estimation, we did a partial implementation of the attack.

Verification of the Puncturing Correlation ρ^2. Let f and g be the accurate and approximated key recovery maps (for just one ciphertext), respectively. In the case of Walsh spectrum puncturing, we know that $\rho^2 = \langle f, g \rangle$. In theory, we should observe $\rho^2 \approx 2^{-11.01}$, and we calculate $\langle f, g \rangle$ experimentally. Specifically, we prepared samples S of 2^{15} inputs, and computed

$$\begin{cases} \frac{1}{|S|} \sum_{Z \in S} f(Z, K) \times g(Z, K_R) & \text{with the right guess } K_R \\ \frac{1}{|S|} \sum_{Z \in S} f(Z, K) \times g(Z, K_W) & \text{with the wrong guess } K_W \end{cases}$$

We repeated this experiment 1000 times with different keys K.

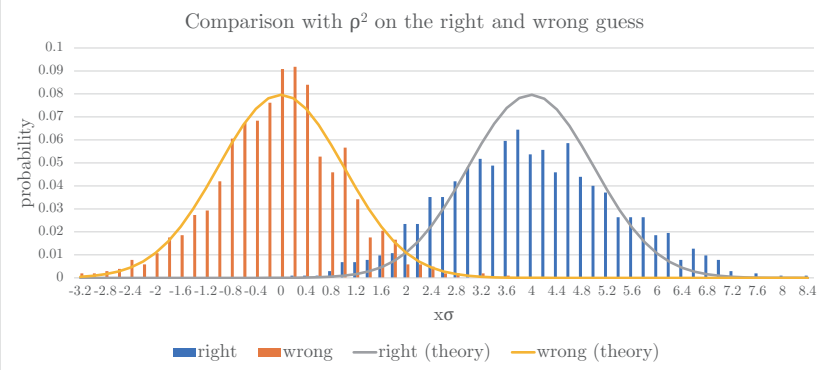


Fig. 6: Comparison of ρ^2 between the expected and experimental correlations for right and wrong keys.

We expect that the average correlation is $2^{-11.01}$ for the right key and 0 for the wrong key. The expected variance is $||g||_2^2/2^{15} \approx 2^{-26.01}$, and the standard deviation is $\sigma = 2^{-13.01}$. Figure 6 compares our expected and experimentally observed correlations. The variance of the right key is slightly higher than the theoretical estimation, and we discuss this issue later.

Verification of Puncturing Correlation, ρ^4. The differential-linear key recovery attack needs to analyze two texts. Assuming both texts are statistically independent, the puncturing correlation is estimated as ρ^4. Indeed, it is natural that we assume both keystreams are statistically independent: otherwise, we could directly observe a non-negligible bias between keystream words. On the other hand, both parts of the key recovery map are fed the same key, so if there is not enough mixing, the outputs of both parts may not be independent. We should thus verify this assumption experimentally. Similarly to the experiment for ρ^2, we randomly sampled set $(Z, Z') \in S$ with $|S| = 2^{25}$ and computed

$$\begin{cases} \frac{1}{|S|} \sum_{(Z,Z')\in S} f(Z,K) \times f(Z',K) \times g(Z,K_R) \times g(Z',K_R) & \text{with } K_R \\ \frac{1}{|S|} \sum_{(Z,Z')\in S} f(Z,K) \times f(Z',K) \times g(Z,K_W) \times g(Z',K_W) & \text{with } K_W \end{cases}$$

We repeated this experiment 1000 times with different keys K.

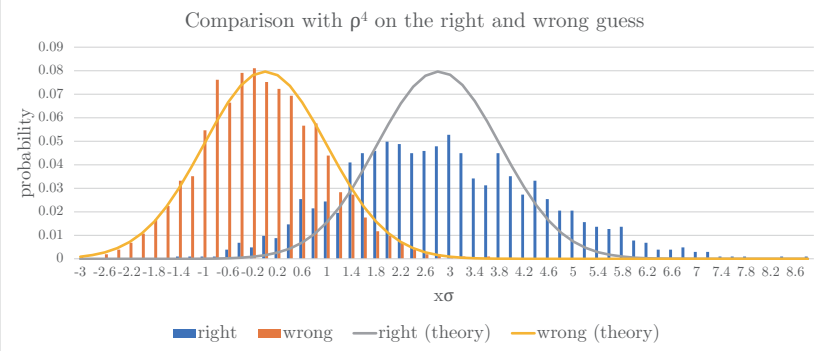


Fig. 7: Comparison of ρ^4 between the expected and experimental correlations for right and wrong keys.

We expect the average correlation to be $2^{-22.02}$ for the right key and 0 for the wrong key. The variance is expected to be $||g||_2^2/2^{25} \approx 2^{-47.02}$, and the standard deviation $\sigma = 2^{-23.51}$. Figure 7 compares the expected and experimental correlations. Interestingly, the empirical average correlation, $2^{-21.92}$, was slightly higher than $2^{-22.02}$. On the other hand, the large variance observed in ρ^2 was clearly amplified in ρ^4. Again, we discuss this issue later.

Verification including the Distinguisher Correlation. We finally verify the right-key and wrong-key distributions for the whole attack, including the DL distinguisher. However, even if we know the right key and guarantee that the differential is satisfied in the first round, we need at least $c^{-2} \times \rho^{-4} \approx 2^{38.6}$ pairs to observe a non-negligible correlation. This is a challenging sample size to collect in practice if we wish to run a significant number of trials. Instead, we use a 5-bit tail instead of a 2-bit tail for f_3 to f_{20}. Then, we have $\rho^4 = (31/32)^{18\times 2} \times 2^{-3.54\times 2} \approx 2^{-8.72}$. Since $c^{-2} \times \rho^{-4} \approx 2^{25.32}$, it is possible to run the experiment many times. We prepared $2^{19}/2^{-8.72}$ pairs satisfying the first round differential and encrypted them to obtain each sample S. We computed

$$\begin{cases} \frac{1}{|S|}\sum_{(Z,Z')\in S} g(Z, K_R) \times g(Z', K_R) & \text{with the right guess } K_R \\ \frac{1}{|S|}\sum_{(Z,Z')\in S} g(Z, K_W) \times g(Z', K_W) & \text{with the wrong guess } K_W \end{cases}$$

We repeated this procedure 1000 times with different keys.

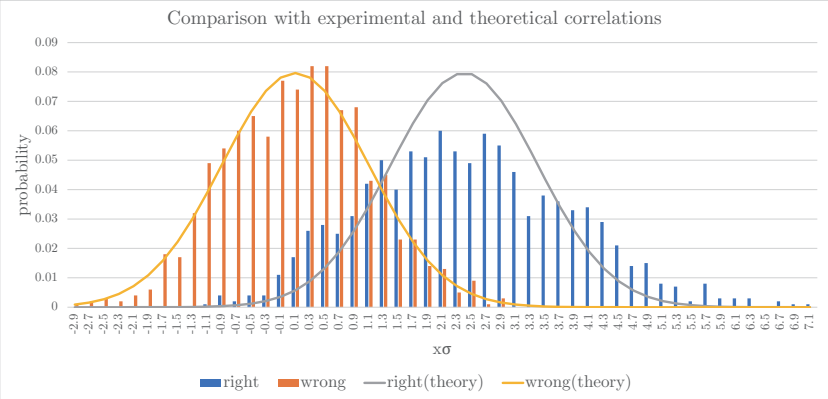
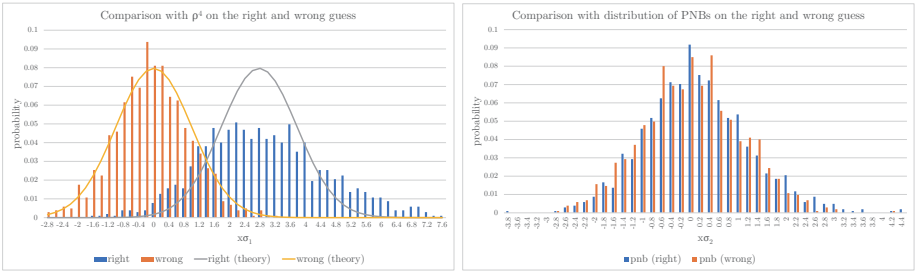


Fig. 8: Correlation comparison between the expected and experimental correlations for right and wrong keys.

The expected average is $2^{-8.3}$ for the right key and 0 for the wrong key. The expected variance is 2^{-19}, and the expected standard deviation is $\sigma = 2^{-7.5}$. Figure 8 compares the expected and experimental distributions.

We observe a similar behaviour to the case of ρ^2 and ρ^4. The right-key variance is higher than our theoretical estimation, but the distributions otherwise matched our estimation.

On the Variance of the Right-Key Distribution. Experimentally, we observed a clearly higher variance than expected in the right-key distribution. We believe this is because of the use of a modular addition instead of XOR to absorb the secret key into the state. The correlation is key-dependent, but our calculations only estimate the average correlation over the punctured key bits. Some



(a) Experiments for ρ^4. (b) Experiments for PNBs.

Fig. 9: The case of the 6-bit guess. Comparison between the right and wrong guesses using 2^{12} pairs for 1000 random keys and theoretical estimation.

keys cause higher or lower correlation than average. Because our theoretical estimation for the variance assumes that all right keys take the same correlation, we observed a gap in the variance.

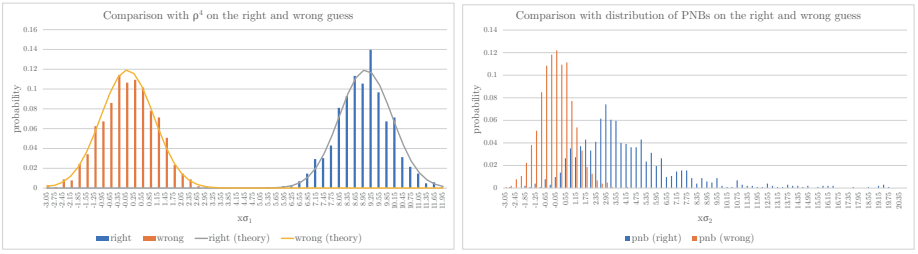
Note that this issue is not unique to puncturing. Rather, the same is true (and arguably worse) for PNBs. The authors of [AFK+08] already mentioned this phenomenon, and they suggested using the median instead of the average as an estimation of the correlation, given the skewed nature of the observed distribution. To discuss this issue, we provide some additional experiments.

First, for the sake of convenience, we changed the punctured f_1 and f_2 to

$$\begin{aligned} f_1 &\approx (-1)^{\bar{z}_0[8]\oplus\bar{z}_5[15]\oplus k_5[15]\oplus\bar{z}_{10}[8]\oplus k_{10}[8]} \times \\ &\quad g_1'(\bar{z}_0[\text{7-3}], \bar{z}_5[\text{14-12}], \bar{z}_{10}[\text{7-5}], k_5[14], k_{10}[\text{7-5}]), \\ f_2 &\approx (-1)^{\bar{z}''_{10}[7]\oplus\bar{z}'_{14}[23]\oplus\bar{z}_3[7]\oplus\bar{z}_4[14]\oplus\bar{z}_9[7]\oplus k_4[14]\oplus k_9[7]\oplus k_{10}[7]} \times \\ &\quad g_2'(\bar{z}''_{10}[\text{6-4}], \bar{z}'_{14}[\text{22-18}], \bar{z}_3[\text{6-2}], \bar{z}_4[\text{13-11}], \bar{z}_9[\text{6-4}], k_4[13], k_9[6], k_{10}[6,5]). \end{aligned}$$

For f_3 to f_{20}, we use partitioning with 5-bit tails. Then, the puncturing correlation is estimated as $\rho^2 \approx 2^{-4.53}$ and $\rho^4 \approx 2^{-9.05}$. To verify ρ^4, we did the same experiment using 2^{12} pairs. Figure 9a shows both distributions. For the right key, the variance is higher than predicted. For comparison, we estimated the so-called backward correlation ε_a using the punctured bits as PNBs, and found that we cannot distinguish the right and wrong keys. Since the observed average and median of the right-key backward correlation were $2^{-8.5019}$ and $2^{-9.4150}$, respectively, 2^{12} pairs are insufficient. This comparison shows the advantage of puncturing over PNBs.

In the case of puncturing, we believe the reason for the higher observed variance is that the correlation depends on the specific value of the punctured key bits. We thus expect that guessing these key bits will result in a lower



(a) Experiments for ρ^4. (b) Experiments for PNBs.

Fig. 10: The case of the 13-bit guess. Comparison between the right and wrong guesses using 2^{12} pairs for 1000 random keys and theoretical estimation.

variance. To confirm this intuition, we change the punctured map for f_1 and f_2:

$$f_1 \approx (-1)^{\bar{z}_0[8]\oplus\bar{z}_5[15]\oplus k_5[15]\oplus\bar{z}_{10}[8]\oplus k_{10}[8]} \times \\ g_1'(\bar{z}_0[7\text{-}3], \bar{z}_5[14\text{-}12], \bar{z}_{10}[7\text{-}5], k_5[14, 13, 12], k_{10}[7\text{-}5]),$$

$$f_2 \approx (-1)^{\bar{z}_{10}''[7]\oplus\bar{z}_{14}'[23]\oplus\bar{z}_3[7]\oplus\bar{z}_4[14]\oplus\bar{z}_9[7]\oplus k_4[14]\oplus k_9[7]\oplus k_{10}[7]} \times \\ g_2'(\bar{z}_{10}''[6\text{-}4], \bar{z}_{14}'[22\text{-}18], \bar{z}_3[6\text{-}2], \bar{z}_4[13\text{-}11], \bar{z}_9[6\text{-}4], k_4[13, 12, 11], k_9[6, 5, 4], k_{10}[6, 5, 4]).$$

These additional guesses improve the puncturing correlation to $\rho^2 \approx 2^{-2.83}$ and $\rho^4 \approx 2^{-5.65}$. We did the same experiment using 2^{12} pairs. Figure 10a shows each distribution. As expected, we no longer observe a significant gap in the variance of the right guess.

The average and median of the correlation for PNBs were $2^{-3.9110}$ and $2^{-4.2056}$, respectively. Using 2^{12} pairs is sufficient to distinguish the right and wrong keys, but as shown in Fig.10b, it is unreasonable to expect that the distribution when using PNBs follows the normal distribution.

According to these experiments, with bit puncturing, similarly to PNBs, the correlation is key-dependent and we observe a higher variance than expectated. On the other hand, the distribution follows the normal distribution, unlike with PNBs. Moreover, when we guess many key bits (like the data-complexity optimized 7-round attack or the 7.5-round attack), we expect better behavior from the variance. In conclusion, we think that the variance gap is not a major concern, especially in comparison to PNBs, and decide to use the right-key distribution for simplicity. Although there is a possibility that we overestimate the success probability, we can confidently predict that it will be larger than 50% which is the limit for PNBs.

C.2 Experiments for 7-Round Attack

As shown in Table 5, we have five component functions containing more than one modular addition. Unfortunately, two functions, f_1 and f_4, are of dimension than we can practically manipulate on current hardware. However, we verify the

rest experimentally. The puncturing correlation for f_2, f_3, and f_5 is $2^{-6.70}$. There are 19 decomposed functions for which we apply partitioning with a 3-bit tail. Thus, the total puncturing correlation is $2^{-6.70} \times (7/8)^{19} \approx 2^{-10.36}$. To verify ρ^2, we used 2^{15} randomly sampled inputs Z, and repeated this experiment 1000 times with different keys.

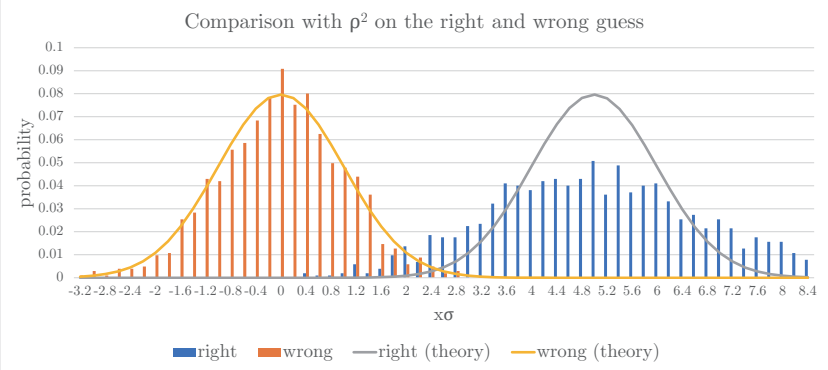


Fig. 11: Comparison between the expected and experimental correlations for the right and wrong keys.

The observed result is very similar to the 6-round attack experiments. The right-key distribution follows the normal distribution, but the variance is higher than estimated, for the already discussed reasons.

D Detail of the Trail Enumerations for the 7-Round Attack

Tables 6, 7, 8, 9, and 10 summarize trail enumeration for f_1, f_2, f_3, f_4, and f_5, respectively.

Table 6: Trail enumeration for f_1

Round	Active	#coeffs	ρ^2
5	v_{10}^{5}[12]	1	1
5.5	$v_{10}^{5.5}$[12,11-0], $v_{15}^{5.5}$[12,11-9]	94	$2^{-0.13}$
6	v_{0}^{6}[12,11-9], v_{10}^{6}[12,11-9], v_{15}^{6}[20,19-17, 12,11-9]	308	$2^{-0.22}$
6.5	$v_{0}^{6.5}$[12,11-0], $v_{3}^{6.5}$[20,19-17, 12,11-9], $v_{4}^{6.5}$[24,23-21], $v_{8}^{6.5}$[12,11-9], $v_{15}^{6.5}$[28,27-25, 4,3-1], v_{10}^{6}[12,11-9]	36776	$2^{-0.44}$
7	v_{0}^{7}[12,11-9], v_{3}^{7}[20,19-17, 12,11-9], v_{4}^{7}[31,30-28, 19,18-16], v_{7}^{7}[27,26-24, ~~19~~], v_{8}^{7}[24,23-21, 12,11-9], v_{11}^{7}[20,19-17, ~~12~~], $v_{15}^{6.5}$[28,27-25, 4,3-1], $v_{8}^{6.5}$[12,11-9], v_{10}^{6}[12,11-9]	2088000 (738352)	$2^{-2.75}$ ($2^{-2.75}$)
end	$\bar{z}_0$[12,11-9], $\bar{z}_3$[20,19-17, 12,11-9], $\bar{z}'_{15}$[28,27-25, 4,3-1], v_{4}^{7}[31,30-28, 19,18-16] $\leftarrow$ {$\bar{z}_4$[31,30-26, 19,18-14], k_4[31,30, 19,18]}, v_{7}^{7}[27,26-24] $\leftarrow$ {$\bar{z}_7$[27,26-22], k_7[27,26,22]}, v_{8}^{7}[24,23-21, 12,11-9] $\leftarrow$ {$\bar{z}_8$[24,23-19, 12,11-7], k_8[24,23, 12,11,10]}, v_{11}^{7}[20,19-17] $\leftarrow$ {$\bar{z}_{11}$[20,19-15], k_{11}[20,19-17,15]}, $v_{8}^{6.5}$[12,11-9] $\leftarrow$ {$\bar{z}'_8$[12,11-7], k'_8[12,11,10]}, v_{10}^{6}[12,11-9] $\leftarrow$ {$\bar{z}''_{10}$[12,11-7], k''_{10}[12,11]}	$\leq 2^{55+14}$	$2^{-7.14}$

Table 7: Trail enumeration for f_2

Round	Active	#coeffs	ρ^2
5.5	$v_{11}^{5.5}$[19]	1	1
6	v_{11}^{6}[~~19~~, 18-14], v_{12}^{6}[19,18-14]	62	$2^{-0.046}$
6.5	$v_{0}^{6.5}$[19,18-14], $v_{12}^{6.5}$[3,2-0,31,30], v_{11}^{6}[18-14]	62	$2^{-0.046}$
7	v_{0}^{7}[19,18-14], v_{4}^{7}[~~26~~], v_{8}^{7}[~~19~~], $v_{12}^{6.5}$[3,2-0,31,30], v_{11}^{6}[18-14]	166	$2^{-2.07}$
end	$\bar{z}_0$[19,18-14], $\bar{z}'_{12}$[3,2-0,31,30], v_{11}^{6}[18-14] $\leftarrow$ {$\bar{z}''_{11}$[18-13], k''_{11}[18,17,15,14]}	1740	$2^{-2.39}$

Table 8: Trail enumeration for f_3

Round	Active	#coeffs	ρ^2
5.5	$v_{11}^{5.5}$[7]	1	1
6	v_{11}^{6}[7,6-2], v_{12}^{6}[7,6-2]	94	$2^{-0.043}$
6.5	$v_{0}^{6.5}$[7,6-2], $v_{12}^{6.5}$[23,22-18], v_{11}^{6}[7,6-2]	94	$2^{-0.043}$
7	v_{0}^{7}[7,6-2], v_{4}^{7}[14,13-9], v_{8}^{7}[7,6-2], $v_{12}^{6.5}$[23,22-18], v_{11}^{6}[7,6-2]	14248	$2^{-0.16}$
end	$\bar{z}_0$[7,6-2], $\bar{z}'_{12}$[23,22-18], v_8^7[7,6-2] $\leftarrow \{\bar{z}_8$[7,6-2], k_8[7,6]$\}$ v_4^7[14,13-9] $\leftarrow \{\bar{z}_4$[14,13-9], k_4[14,13]$\}$ v_{11}^6[7,6-2] $\leftarrow \{\bar{z}''_{11}$[7,6-2], k''_{11}[7,6]$\}$	1570720	$2^{-2.57}$

Table 9: Trail enumeration for f_4

Round	Active	#coeffs	ρ^2
6	v_3^6[16]	1	1
6.5	$v_3^{6.5}$[16,15-4], $v_7^{6.5}$[28,27-25], $v_{11}^{6.5}$[16,15-13]	86	$2^{-0.13}$
7	v_3^7[16,15-11], v_7^7[23,22-18, 3,2-0], v_{11}^7[28,27-25, 16,15-11], $v_{11}^{6.5}$[16,15-13]	1652	$2^{-0.13}$
end	$\bar{z}_3$[16,15-11], v_7^7[23,22-18, 3,2-0] $\leftarrow \{\bar{z}_7$[23,22-18, 3,2-0], k_7[23,22, 3,2]$\}$, v_{11}^7[28,27-25, 16,15-11] $\leftarrow \{\bar{z}_{11}$[28,27-23, 16,15-11], k_{11}[28,27, 16,15,14]$\}$, $v_{11}^{6.5}$[16,15-13] $\leftarrow \{\bar{z}'_{11}$[16,15-11], k_{11}[16,15,14]$\}$	$\leq 2^{28+7}$	$2^{-3.07}$

Table 10: Trail enumeration for f_5

Round	Active	#coeffs	ρ^2
6.5	$v_2^{6.5}$[24]	1	1
7	v_2^7[24,23-19], v_6^7[31,30-26], v_{10}^7[24,23-19]	62	$2^{-0.046}$
end	$\bar{z}_2$[24,23-19], v_6^7[31,30-26] $\leftarrow \{\bar{z}_6$[31,30-26], k_6[31,30]$\}$, v_{10}^7[24,23-19] $\leftarrow \{\bar{z}_{10}$[24,23-19], k_{10}[24,23]$\}$	2144	$2^{-1.74}$

E 7-Round Attack Reducing the Data Complexity

Table 11: Punctured functions for another 7-round attack.

Target	Active	#coeffs	ρ^2
f_1	$\bar{z}_0$[12,11-9], $\bar{z}_3$[20,19-17, 12,11-9], $\bar{z}'_{15}$[28,27-25, 4,3-1], v_4^7[31,30-28, 19,18-16] $\leftarrow$ {$\bar{z}_4$[31,30-26, 19,18-14], k_4[31,30-28, 19,18-16]}, v_7^7[27,26-24] $\leftarrow$ {$\bar{z}_7$[27,26-22], k_7[27,26-24,22]}, v_8^7[24,23-21, 12,11-9] $\leftarrow$ {$\bar{z}_8$[24,23-19, 12,11-7], k_8[24,23-21, 12,11-7]}, v_{11}^7[20,19-17] $\leftarrow$ {$\bar{z}_{11}$[20,19-15], k_{11}[20,19-15]}, $v_8^{6.5}$[12,11-9] $\leftarrow$ {$\bar{z}'_8$[12,11-7], k'_8[12,11-7]}, v_{10}^6[12,11-9] $\leftarrow$ {$\bar{z}''_{10}$[12,11-7], k''_{10}[12,11,10]}	$\leq 2^{55+30}$	$2^{-3.97}$
f_2	$\bar{z}_0$[19,18-14], $\bar{z}'_{12}$[3,2-0,31,30], v_{11}^6[18-14] $\leftarrow$ {$\bar{z}''_{11}$[18-13], k''_{11}[18-13]}	5187	$2^{-2.11}$
f_3	$\bar{z}_0$[7,6-2], $\bar{z}'_{12}$[23,22-18], v_8^7[7,6-2] $\leftarrow$ {$\bar{z}_8$[7,6-2], k_8[7,6-3]}, v_4^7[14,13-9] $\leftarrow$ {$\bar{z}_4$[14,13-9], k_4[14,13-11]}, v_{11}^6[7,6-2] $\leftarrow$ {$\bar{z}''_{11}$[7,6-2], k''_{11}[7,6,5]}	$\leq 2^{25+9}$	$2^{-0.92}$
f_4	$\bar{z}_3$[16,15-11], v_7^7[23,22-18, 3,2-0] $\leftarrow$ {$\bar{z}_7$[23,22-18, 3,2-0], k_7[23,22-20, 3,2-0]}, v_{11}^7[28,27-25, 16,15-11] $\leftarrow$ {$\bar{z}_{11}$[28,27-23, 16,15-11], k_{11}[28,27-25, 16,15-12]}, $v_{11}^{6.5}$[16,15-13] $\leftarrow$ {$\bar{z}'_{11}$[16,15-11], k_{11}[16,15-12]}	$\leq 2^{28+17}$	$2^{-0.85}$
f_5	$\bar{z}_2$[24,23-19], v_6^7[31,30-26] $\leftarrow$ {$\bar{z}_6$[31,30-26], k_6[31,30-28]}, v_{10}^7[24,23-19] $\leftarrow$ {$\bar{z}_{10}$[24,23-19], k_{10}[24,23-21]}	49888	$2^{-0.58}$

F The 7.5-Round Attack

We extend the 7-round attack to a 7.5-round attack. We use the same 5-round DL distinguisher with autocorrelation $2^{-39.2}$. We decompose the key recovery map as a sum and construct punctured functions for each of the components.

F.1 Linear Decomposition of the Key-Recovery Map

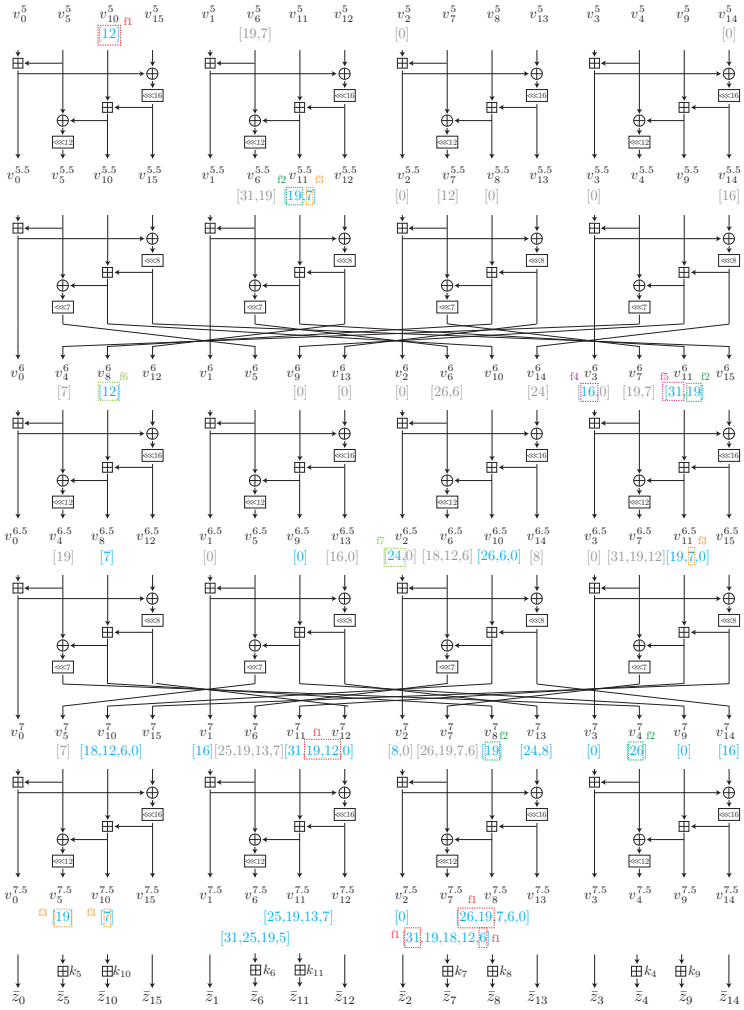


Fig. 12: Key recovery map for the 7.5-round attack.

Figure 12 shows the linear propagation (with correlation 1), where the XOR of the blue bits is equal to $v_2^5[0] \oplus v_6^5[19,7] \oplus v_{10}^5[12] \oplus v_{14}^5[0]$. As mentioned in

Sect 7, we guess the full 32-bit k_4 and 20 bits k_9[19-0]. Thus, v_3^7[19-0], v_4^7[19-0], and the carry toward v_3^7[20] can be evaluated without the trail enumeration.

We have 24 component functions. These nine components involve more than one modular subtraction:

$$f_1 = (-1)^{v_{10}^5[12] \oplus v_{11}^7[19] \oplus v_{11}^7[12] \oplus v_7^{7.5}[31] \oplus v_7^{7.5}[6] \oplus v_8^{7.5}[26] \oplus v_8^{7.5}[19]},$$
$$f_2 = (-1)^{v_{11}^{5.5}[19] \oplus v_{11}^6[19] \oplus v_4^7[26] \oplus v_8^7[19]},$$
$$f_3 = (-1)^{v_{11}^{5.5}[7] \oplus v_{11}^{6.5}[7] \oplus v_5^{7.5}[19] \oplus v_{10}^{7.5}[7]},$$
$$f_4 = (-1)^{v_3^6[16]}, \quad f_5 = (-1)^{v_8^6[12]}, \quad f_6 = (-1)^{v_{11}^6[31]}, \quad f_7 = (-1)^{v_2^{6.5}[24]},$$
$$f_8 = (-1)^{v_1^7[16]}, \quad f_9 = (-1)^{v_2^7[8]}.$$

These 19 components involve a modular subtraction only:

$$f_{10} = (-1)^{v_6^{7.5}[31]}, \quad f_{11} = (-1)^{v_6^{7.5}[25]}, \quad f_{12} = (-1)^{v_6^{7.5}[19]}, \quad f_{13} = (-1)^{v_6^{7.5}[5]},$$
$$f_{14} = (-1)^{v_7^{7.5}[19] \oplus v_7^{7.5}[18]}, \quad f_{15} = (-1)^{v_7^{7.5}[12]}, \quad f_{16} = (-1)^{v_8^{7.5}[7] \oplus v_8^{7.5}[6]}, \quad f_{17} = (-1)^{v_{11}^{7.5}[25]},$$
$$f_{18} = (-1)^{v_{11}^{7.5}[19]}, \quad f_{19} = (-1)^{v_{11}^{7.5}[13]}, \quad f_{20} = (-1)^{v_{11}^{7.5}[7]}, \quad f_{21} = (-1)^{v_8^{6.5}[7]},$$
$$f_{22} = (-1)^{v_{10}^{6.5}[26]}, \quad f_{23} = (-1)^{v_{10}^{6.5}[6]}, \quad f_{24} = (-1)^{v_{11}^{6.5}[19]}, \quad f_{25} = (-1)^{v_{10}^7[18]},$$
$$f_{26} = (-1)^{v_{10}^7[12]}, \quad f_{27} = (-1)^{v_{11}^7[6]}, \quad f_{28} = (-1)^{v_{11}^7[31]}.$$

F.2 Punctured Functions for Each Decomposition

Tables 12, 13, 14, 15, 16, 17, 18, 19, and 20 summarize trail enumeration for f_1 to f_9, respectively.

The product of the puncturing correlations from f_1 to f_9 is $2^{-20.21}$. Four functions, f_{16}, f_{21}, f_{26} and f_{28} share many active key bits with those for f_1 to f_9. Thus, we evaluate these functions in Phase 1, where we use 6, 6, 4, and 5-bit tails for f_{16}, f_{21}, f_{16}, and f_{28}, respectively. The fraction of ciphertexts for which we can evaluate these functions is

$$\left(\frac{63}{64}\right)^2 \times \left(\frac{15}{16}\right) \times \left(\frac{31}{32}\right) \approx 2^{-0.18}.$$

We evaluate the rest of the functions in Phase 2, using 4-bit tails, meaning probability $(15/16)^{15} \approx 2^{-1.40}$. In total, the puncturing correlation is

$$\rho^2 = 2^{-20.21-0.18-1.40} \approx 2^{-21.79}.$$

F.3 Attack Procedure

We first discuss the data complexity. When the autocorrelation is $2^{-39.2}$, $N' = 2^{82.5}$ pairs provide a 7-bit advantage with a success probability of 93.07 %. Then, we use

$$N = N' \times 2^{21.79 \times 2} \approx 2^{126.08}$$

pairs, and the data complexity is $2^{127.08}$.

Table 12: Trail enumeration for f_1

Round	Active	#coeffs	ρ^2
5	v_{10}^{5}[12]	1	1
5.5	$v_{10}^{5.5}$[12,11-0], $v_{15}^{5.5}$[12,11,10]	50	$2^{-0.26}$
6	v_{0}^{6}[12,11,10], v_{10}^{6}[12,11-0], v_{15}^{6}[20,19,18, 12,11,10]	1396	$2^{-0.38}$
6.5	$v_{0}^{6.5}$[12,11-0], $v_{3}^{6.5}$[20,19,18, 12,11,10], $v_{4}^{6.5}$[24,23,22], $v_{8}^{6.5}$[12,11,10], $v_{10}^{6.5}$[12,11,10], $v_{14}^{6.5}$[12,11,10], $v_{15}^{6.5}$[28,27,26, 4,3,2]	15360	$2^{-0.93}$
7	v_{0}^{7}[12,11,10], v_{2}^{7}[12,11,10], v_{3}^{7}[28,27,26, 20,19,18, 12,11,10, 4,3,2], v_{4}^{7}[31,30,29, 19,18,17], v_{7}^{7}[27,26,25, 19], v_{8}^{7}[24,23,22, 12,11,10], v_{11}^{7}[20,19,18, ~~12~~], v_{14}^{7}[20,19,18], v_{15}^{7}[4,3,2, 12,11,10], $v_{8}^{6.5}$[12,11,10], $v_{10}^{6.5}$[12,11,10]	99840	$2^{-3.54}$
7.5	$v_{0}^{7.5}$[12,11-9], $v_{2}^{7.5}$[12,11-9], $v_{3}^{7.5}$[28,27-25], $v_{5}^{7.5}$[24,23-21], $v_{7}^{7.5}$[~~31~~, 24,23-21, 7,6,5], $v_{8}^{7.5}$[27,26,25, ~~19~~, 12,11-9], $v_{9}^{7.5}$[31,30,29, 28,27-25], $v_{10}^{7.5}$[12,11-9], v_{14}^{7}[20,19,18], v_{15}^{7}[12,11,10, 4,3,2], $v_{8}^{6.5}$[12,11,10], $v_{10}^{6.5}$[12,11,10] v_{8}^{7}[24,23,22, 12,11,10]v_{11}^{7}[20,19,18], v_{3}^{7}[20,19,18, 12,11,10, 4,3,2], v_{4}^{7}[19,18,17], $v_{4}^{7.5}$[11,10,9, 8,7-5],	(3289600)	$(2^{-4.23})$
end	$\bar{z}_0$[12,11-9], $\bar{z}_2$[12,11-9], $\bar{z}_3$[28,27-25], $\bar{z}'_{14}$[20,19,18], $\bar{z}'_{15}$[12,11,10, 4,3,2], $v_{5}^{7.5}$[24,23-21] $\leftarrow$ {$\bar{z}_5$[24,23-17], k_5[24,23-21]}, $v_{7}^{7.5}$[24,23-21, 7,6,5] $\leftarrow$ {$\bar{z}_7$[24,23-17, 7,6-0], k_7[24,23,22,19,18, 7,6,5,3-0]} $v_{8}^{7.5}$[27,26,25, 12,11-9] $\leftarrow$ {$\bar{z}_8$[27,26-20, 12,11-5], k_8[27,26,25,23-20, 12,11-5]}, $v_{9}^{7.5}$[31,30,29, 28,27-25] $\leftarrow$ {$\bar{z}_9$[31,30-24], k_9[31,30-26]} $v_{10}^{7.5}$[12,11-9] $\leftarrow$ {$\bar{z}_{10}$[12,11-5], k_{10}[12,11-8]} $v_{8}^{6.5}$[12,11,10] $\leftarrow$ {$\bar{z}''_{8}$[12,11-5], k_8[12,11-5]}, $v_{10}^{6.5}$[12,11,10] $\leftarrow$ {$\bar{z}''_{10}$[12,11-5], k_{10}[12,11-8]} v_{8}^{7}[24,23,22, 12,11,10] $\leftarrow$ {$\bar{z}'_{8}$[24,23-17, 12,11-5], k_8[24,23-20, 12,11-5]} v_{11}^{7}[20,19,18] $\leftarrow$ {$\bar{z}'_{11}$[20,19-13], k_{11}[20,19,18,15-13]}, v_{3}^{7}[20,19,18, 12,11,10, 4,3,2], v_{4}^{7}[19,18,17], $v_{4}^{7.5}$[11,10,9, 8,7-5],	$\leq 2^{112+62}$	$2^{-6.44}$

Phase 1. We first evaluate f_1 to f_9 and f_{16}, f_{21}, f_{16}, and f_{28}. We guess 122 bits of key in total, and the fraction of available pairs is $\left(\frac{63}{64}\right)^2 \times \left(\frac{15}{16}\right) \times \left(\frac{31}{32}\right)^2 \approx 2^{-0.37}$. If N is the number of pairs, the time complexity is

$$2N \times 2^{-0.37} \times 2^{122} \approx 2^{248.71}.$$

Phase 2. We still have 15 functions. Since we use a 4-bit tail for each function, the size of the distillation table is $15 \times 8 + 122 = 120 + 122 = 242$. We first traverse $2^{247.71}$ pairs to construct the distillation table.

Then, we evaluate each function one by one, and at most 4 key bits are guessed for each step. Specifically, we first evaluate f_{20}, which requires an additional 2-bit guess. Then, the complexity is $2^{242+2} = 2^{244}$. Since the first step is dominant, the complexity of the following steps can be regarded as negligible. Thus, the time complexity of the second phase is about

$$2^{247.71} + 2^{244} \approx 2^{247.82}.$$

Table 13: Trail enumeration for f_2

Round	Active	#coeffs	ρ^2
5.5	$v_{11}^{5.5}$[19]	1	1
6	v_{11}^{6}[~~19~~], v_{12}^{6}[19,18-16]	3	$2^{-1.61}$
6.5	$v_{0}^{6.5}$[19,18-16], $v_{12}^{6.5}$[3,2-0]	3	$2^{-1.61}$
7	v_{0}^{7}[19,18-7, 3,2-0], v_{4}^{7}[~~26~~], v_{8}^{7}[~~19~~], v_{12}^{7}[11,10-8]	33	$2^{-3.49}$
7.5	$v_{0}^{7.5}$[19,18-14, 3,2-0], $v_{5}^{7.5}$[31,30-26, 15,14-12], $v_{10}^{7.5}$[19,18-14, 3,2-0], v_{12}^{7}[11,10-8]	14996	$2^{-3.50}$
end	$\bar{z}_{0}^{7.5}$[19,18-14, 3,2-0], $\bar{z}'_{12}$[11,10-8], $v_{5}^{7.5}$[31,30-26, 15,14-12] $\leftarrow \{\bar{z}_5$[31,30-24, 15,14-8$], k_5$[31,30,29, 15,14,13$]\}$, $v_{10}^{7.5}$[19,18-14, 3,2-0] $\leftarrow \{\bar{z}_{10}$[19,18-12, 3,2-0$], k_{10}$[19,18,17, 3,2,1$]\}$,	$\leq 2^{35+8}$	$2^{-5.12}$

Table 14: Trail enumeration for f_3

Round	Active	#coeffs	ρ^2
5.5	$v_{11}^{5.5}$[7]	1	1
6	v_{11}^{6}[7,6-0], v_{12}^{6}[7,6-4]	54	$2^{-0.13}$
6.5	$v_{0}^{6.5}$[7,6-4], $v_{11}^{6.5}$[~~7~~,6-4], $v_{12}^{6.5}$[23,22-20], v_{15}^{6}[7,6-4]	272	$2^{-0.22}$
7	v_{0}^{7}[23,22-20, 7,6-4], v_{3}^{7}[7,6-4], v_{4}^{7}[14,13-11], v_{8}^{7}[7,6-4], v_{12}^{7}[31,30-28], v_{15}^{7}[15,14-12], $v_{11}^{6.5}$[6-4]	6824	$2^{-0.56}$
7.5	$v_{0}^{7.5}$[23,22-19, 7,6-3], $v_{5}^{7.5}$[~~19~~,3,2-0],$v_{10}^{7.5}$[23,22-20, ~~7~~], v_{12}^{7}[31,30-28], v_{15}^{7}[15,14-12], $v_{11}^{6.5}$[6-4], v_{8}^{7}[7,6-4], v_{3}^{7}[7,6-4], v_{4}^{7}[14,13-11],	409888	$2^{-2.36}$
end	$\bar{z}_{0}^{7.5}$[23,22-19, 7,6-3], $\bar{z}'_{12}$[31,30-28], $\bar{z}'_{15}$[15,14-12], $v_{5}^{7.5}$[3,2-0] $\leftarrow \{\bar{z}_5$[3,2-0$], k_5$[3,2,1$]\}$, $v_{10}^{7.5}$[23,22-20] $\leftarrow \{\bar{z}_{10}$[23,22-16$], k_{10}$[23,22,21,18,17$]\}$, $v_{11}^{6.5}$[6-4] $\leftarrow \{\bar{z}''_{11}$[6-0$], k_{11}$[6, 5$]\}$, v_{8}^{7}[7,6-4] $\leftarrow \{\bar{z}'_{8}$[7,6-0$], k_8$[7,6-1$]\}$, v_{3}^{7}[7,6-4], v_{4}^{7}[14,13-11],	$\leq 2^{44+14}$	$2^{-3.48}$

Exhaustive Search and Final Time Complexity. With a 7-bit advantage, we need to guess $2^{256-7} = 2^{249}$ keys. Therefore, the final time complexity is

$$2^{248.71} + 2^{247.82} + 2^{249} \approx 2^{250.17}.$$

Table 15: Trail enumeration for f_4

Round	Active	#coeffs	ρ^2
6	v_3^6[16]	1	1
6.5	$v_3^{6.5}$[16,15-4], $v_7^{6.5}$[28,27-25], $v_{11}^{6.5}$[16,15-13]	86	$2^{-0.13}$
7	v_3^7[16,15-2], v_7^7[23,22-20, 3,2-0], v_{11}^7[28,27-25, 16,15-13], $v_{11}^{6.5}$[16,15-13]	5080	$2^{-0.18}$
7.5	$v_7^{7.5}$[15,14-12, 3,2-0], $v_8^{7.5}$[23,22-20, 3,2-0], v_{11}^7[28,27-25, 16,15-13], $v_{11}^{6.5}$[16,15-13], v_3^7[16,15-2],	5080	$2^{-0.18}$
7.5	$v_7^{7.5}$[15,14-12, 3,2-0] $\leftarrow \{\bar{z}_7$[15,14-8, 3,2-0], k_7[15,14,13, 3,2-0]$\}$, $v_8^{7.5}$[23,22-20, 3,2-0] $\leftarrow \{\bar{z}_8$[23,22-16, 3,2-0], k_8[23,22-20, 3,2,1]$\}$, v_{11}^7[28,27-25, 16,15-13] $\leftarrow \{\bar{z}'_{11}$[28,27-21, 16,15-9], k'_{11}[28,27,26, 16,15-13]$\}$, $v_{11}^{6.5}$[16,15-13] $\leftarrow \{\bar{z}''_{11}$[16,15-9], k_{11}[16,15-13]$\}$, v_3^7[16,15-2],	$\leq 2^{55+18}$	$2^{-1.76}$

Table 16: Trail enumeration for f_5

Round	Active	#coeffs	ρ^2
6	v_{11}^6[31]	1	1
6.5	$v_{11}^{6.5}$[31,30-19], $v_{15}^{6.5}$[31,30-26]	286	$2^{-0.030}$
7	v_3^7[31,30-26], v_{15}^7[7,6-2], $v_{11}^{6.5}$[31,30-19]	286	$2^{-0.030}$
7.5	$v_3^{7.5}$[31,30-26], $v_9^{7.5}$[31,30-26], v_{15}^7[7,6-2], v_{11}^7[31,30-19], $v_4^{7.5}$[11,10-6]	22624	$2^{-0.16}$
end	$\bar{z}_3$[31,30-26], $\bar{z}'_{15}$[7,6-2], $v_9^{7.5}$[31,30-26] $\leftarrow \{\bar{z}_9$[31,30-24], k_9[31,30-26]$\}$, v_{11}^7[31,30-19] $\leftarrow \{\bar{z}''_{11}$[31,30-24], k_{11}[31,30-26]$\}$, $v_4^{7.5}$[11,10-6]	$\leq 2^{29+10}$	$2^{-0.29}$

Table 17: Trail enumeration for f_6

Round	Active	#coeffs	ρ^2
6	v_8^6[12]	1	1
6.5	$v_8^{6.5}$[12,11-7], $v_{12}^{6.5}$[12,11-7]	94	$2^{-0.046}$
7	v_0^7[12,11-7], v_{12}^7[20,19-15], $v_8^{6.5}$[12,11-7]	94	$2^{-0.046}$
7.5	$v_0^{7.5}$[12,11-7], $v_5^{7.5}$[24,23-19], $v_{10}^{7.5}$[12,11-7], v_{12}^7[20,19-15], $v_8^{6.5}$[12,11-7]	14248	$2^{-0.17}$
end	$\bar{z}_0$[12,11-7], $\bar{z}'_{12}$[20,19-15], $v_5^{7.5}$[24,23-19] $\leftarrow \{\bar{z}_5$[24,23-17], k_5[24,23-21]$\}$, $v_{10}^{7.5}$[12,11-7] $\leftarrow \{\bar{z}_{10}$[12,11-5], k_{10}[12,11-8]$\}$, $v_8^{6.5}$[12,11-7] $\leftarrow \{\bar{z}''_8$[12,11-5], k_8[12,11-5]$\}$	$\leq 2^{31+14}$	$2^{-0.76}$

Table 18: Trail enumeration for f_7

Round	Active	#coeffs	ρ^2
6.5	$v_2^{6.5}$[24]	1	1
7	v_2^7[24,23-19], v_6^7[31,30-26], v_{10}^7[24,23-19]	62	$2^{-0.046}$
7.5	$v_2^{7.5}$[24,23-19], $v_6^{7.5}$[11,10-6], $v_7^{7.5}$[4,3-0], $v_8^{7.5}$[24,23-20], v_{11}^7[31,30-26], v_{10}^7[24,23-19],	2592	$2^{-0.087}$
end	$\bar{z}_2$[24,23-19], $v_6^{7.5}$[11,10-6] $\leftarrow \{\bar{z}_6$[11,10-4], k_6[11,10,9]$\}$, $v_7^{7.5}$[4,3-0] $\leftarrow \{\bar{z}_7$[4,3-0], k_7[4,3-0]$\}$, $v_8^{7.5}$[24,23-20] $\leftarrow \{\bar{z}_8$[24,23-17], k_8[24,23-20]$\}$, v_{10}^7[24,23-19] $\leftarrow \{\bar{z}'_{10}$[24,23-17], k_{10}[24,23-21,18,17]$\}$, v_{11}^7[31,30-26] $\leftarrow \{\bar{z}'_{11}$[31,30-24], k_{11}[31,30-26]$\}$,	$\leq 2^{37+20}$	$2^{-0.95}$

Table 19: Trail enumeration for f_8

Round	Active	#coeffs	ρ^2
7	v_1^7[16]	1	1
7.5	$v_1^{7.5}$[16,15-11], $v_6^{7.5}$[28,27-23], $v_{11}^{7.5}$[16,15-11]	62	$2^{-0.046}$
end	$\bar{z}_1$[16,15-11], $v_6^{7.5}$[28,27-23] $\leftarrow \{\bar{z}_6$[28,27-21], k_6[28,27,26]$\}$, $v_{11}^{7.5}$[16,15-11] $\leftarrow \{\bar{z}_{11}$[16,15-9], k_{11}[16,15-13]$\}$	$\leq 2^{19+5}$	$2^{-0.79}$

Table 20: Trail enumeration for f_9

Round	Active	#coeffs	ρ^2
7	v_2^7[8]	1	1
7.5	$v_2^{7.5}$[8,7-3], $v_7^{7.5}$[20,19-15], $v_8^{7.5}$[8,7-3]	94	$2^{-0.045}$
end	$\bar{z}_2$[8,7-3], $v_7^{7.5}$[20,19-15] $\leftarrow \{\bar{z}_7$[20,19-13], k_7[20,19,18,14,13]$\}$, $v_8^{7.5}$[8,7-3] $\leftarrow \{\bar{z}_8$[8,7-1], k_8[8,7-1]$\}$	$\leq 2^{19+11}$	$2^{-0.62}$