Doctoral Dissertation

A Study on Differential Cryptanalysis of Salsa 20 and ChaCha Stream Ciphers

Graduate School of Engineering
Osaka University

Nasratullah Ghafoori

September 2024

Supervisor: Professor Atsuko Miyaji

Abstract

We use digital platforms, in communications, financial transactions, education, entertainment, healthcare, and transportation. The use of digital platforms have significantly simplified our daily lives. However, security has consistently emerged as the primary concern when integrating digital platforms into our daily activities. To address security concerns, cryptography has always been essential for ensuring data confidentiality and integrity, both in transit and at rest. As cryptographic primitives become more widely used, it is crucial to thoroughly understand their security properties to ensure data integrity and confidentiality.

This thesis studies the symmetric cryptography. We focus on the security assessment of stream ciphers. Stream ciphers are symmetric cryptographic functions that operate on individual bits or bytes of data, unlike block ciphers, which process data in fixed blocks. Their speed and bitwise encryption make them applicable for specific applications such as secure communications, stream media, disk encryption, voice-over IP, instant messaging, and more. Salsa 20 and ChaCha are deployed in Operating Systems, TLS 1.3, programming libraries, networks, and others. The existing research domain mounted differential and differential-linear attacks on Salsa 8, ChaCha 7, and ChaCha 7.25. The most effective key-recovery attack on Salsa 8 has been proposed with time complexities of $2^{245.5}$, $2^{244.9}$, and $2^{243.6}$, respectively. Furthermore, the best key-recovery attack on ChaCha 7.25 was presented with a complexity of $2^{255.62}$. The distinguisher for ChaCha 6 has been proposed with complexities of 2^{116} and 2^{51}. However, the existing research studies have not proposed the boomerang attack on the ChaCha permutation function.

This dissertation examines the robustness of Salsa 20 and ChaCha stream ciphers and it makes three key contributions, each presented in a dedicated chapter. First, it investigates the differential cryptanalysis of Salsa 20 and ChaCha, focusing on key-recovery attacks on their reduced-round versions. Next, it explores the higher-order differential-linear cryptanalysis of ChaCha. Finally, the dissertation examines the boomerang attack on the ChaCha permutation function, providing a detailed analysis of its security implications.

The first research aimed to evaluate the security of Salsa 20 and ChaCha through an in-depth analysis of probabilistic neutral bits within a differential attack framework.

This resulted in a key-recovery attack with a time complexity of $2^{241.62}$ and a data complexity of $2^{31.5}$ on Salsa 8. Additionally, it resulted in a key-recovery attack on ChaCha 7.25 with a time complexity of $2^{254.011}$ and a data complexity of $2^{51.81}$. As a result, we reduced the attack complexity with a factor of $2^{1.98}$ and $2^{1.6}$ on Salsa 8 and ChaCha 7.25, respectively.

The second research delves into higher-order differential-linear cryptanalysis and studies the security of ChaCha stream cipher. Additionally, we propose a higher-order differential-linear distinguishing attack on ChaCha 5, ChaCha 5.5, and ChaCha 6, demonstrating time complexities of $2^{33.21}$, $2^{63.21}$, and $2^{87.21}$, respectively. The existing research domain has not introduced distinguishing attacks. Moreover, we presented a distinguisher of Chacha 6 with a complexity of $2^{47.21}$ and substantially enhanced the ChaCha 6 distinguisher by $2^{3.79}$.

The third study investigated the permutation function of the ChaCha stream cipher using boomerang cryptanalysis, a variant of differential cryptanalysis. This attack merges two distinct differential properties from separate sections of a cipher into a new differential that applies to the entire cipher, occurring with a probability of p^2q^2. This requires that both differential properties be satisfied twice. Additionally, we demonstrated that the likelihood could increase to p^2 for certain attack positions in ChaCha, reducing the attack complexity. Moreover, for the first time, we proposed an algorithm for executing boomerang attacks on the ChaCha stream cipher was also introduced. To demonstrate the effectiveness of boomerang cryptanalysis, we targeted the permutation functions of ChaCha 6 and ChaCha 7. Our findings indicate that a boomerang attack requires $2^{4.04}$ and $2^{5.99}$ adaptively chosen plaintexts and ciphertexts to distinguish the permutation functions of ChaCha 6 and ChaCha 7 from a random permutation, respectively.

This dissertation evaluates the security of Salsa 20 and ChaCha stream ciphers through three cryptanalytic approaches. The first research presents key-recovery attacks on Salsa 8 and ChaCha 7.25 with complexities of $2^{241.62}$ and $2^{254.011}$, respectively. The second research introduces higher-order differential-linear attacks on ChaCha 5, ChaCha 5.5, and ChaCha 6, with complexities of $2^{33.21}$, $2^{63.21}$, and $2^{87.21}$, respectively. It proposed an improved distinguisher for ChaCha 6. The third study applies boomerang cryptanalysis to the permutation functions of ChaCha 6 and 7, requiring $2^{4.04}$and $2^{5.99}$ adaptively chosen plaintexts and ciphertexts to distinguish them from random permutations.

Acknowledgments

In Japan, during my Ph.D. journey, many have offered incredible support. While thanking everyone is impossible, I want to thank those who profoundly impacted me. First, immense thanks to my supervisor, Professor Atsuko Miyaji. Her deep knowledge, inspiring guidance, and enthusiasm powered my research and made her an exceptional supervisor. This thesis wouldn't exist without her. I sincerely thank Dr Yosuke Todo for reviewing my dissertation and providing invaluable comments. His unwavering enthusiasm and insightful ideas constantly motivated and enriched me throughout this journey. I deeply appreciate his generous time and attention, ensuring my concerns were heard. I am also profoundly grateful to Professor Tetsuya Takine and Lecturer Yuya Tarutani for their advice and support. His comments on values have been particularly beneficial to my work. Additionally, I sincerely thank the members of my examination committee, Professor Yuichi Tanaka, Professor Kyo Inoue, and Professor Akihiro Maruta, for their time, effort, and valuable feedback. Their collective expertise and guidance have been instrumental in refining my research and bringing this dissertation to fruition. Further, thanks go to the Japanese government MEXT scholarship and Osaka University. Their unwavering support throughout my journey was invaluable. Their financial assistance and Osaka University's stimulating research environment enabled me to pursue my academic goals and complete this thesis. I would like to thank Professor Hisato Shima, who has always been inspiring and supportive. I am grateful for his guidance during my application for the MEXT scholarship. He has consistently been there for me whenever I needed his support. Ultimately, I extend my heartfelt appreciation to my beloved family. Their constant encouragement and advice have been invaluable during my life and educational journey. They walked beside me every step, their love and encouragement sustaining me in even the toughest times. This Ph.D. dissertation is dedicated to my great father and beloved mother who held my hand and lit my path: My beloved mother, whose unwavering love and belief in me were an endless source of strength and inspiration. Thank you for teaching me resilience and the courage to chase my dreams. My father's legacy of wisdom and perseverance guided me through moments of doubt and challenge. You instilled in me a thirst for knowledge and a commitment to excellence. May this work reflect the love and support you have generously given me. I am always proud of having you beside

me. I would like to thank my dearest brother, Ahmad Siar Ghafoori, and Enayatullah Ghafoori, who always encouraged me and unconditionally supported me throughout my education journey. I want to express my deepest gratitude to my beloved wife, whose unwavering love and support have been my strength during this time. Her patience, encouragement, and belief in me have made this accomplishment possible. To my precious daughter, Yusra Ghafoori, who has been a source of healing and energy during my Ph.D. journey and while drafting this thesis. I hope you will make your father proud and one day complete your Ph.D.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Cryptography is the practice of securing information by transforming it into an unreadable format. It allows only authorized parties to read and modify it. This transformation involves mathematical algorithms that convert readable information called plaintext into an unreadable form called ciphertext. Symmetric cryptography, also known as secret-key cryptography. It employs the same key for both encryption and decryption. A secret key ensures that only those with the key can decrypt the information. Symmetric ciphers secure data across various applications such as mobile phones, ATMs, online shopping, online payment systems, credit cards, social media platforms, video games, etc. Symmetric ciphers ensure confidentiality in digital communications across different platforms.

Symmetric ciphers are integral to Internet communications, where they secure data transmission over insecure networks. For instance, protocols like SSL/TLS use a combination of symmetric and asymmetric cryptography to protect information exchanged between web browsers and servers. Web users, website administrators, and email service providers rely on these cryptographic protocols to maintain secure and trustworthy online interactions. Cryptosystems are pivotal in securing online banking and digital payment systems in financial transactions.

Symmetric ciphers, known for their efficiency, are employed to encrypt large volumes of data quickly. Asymmetric encryption is used for secure key exchange. This combination of cryptographic methods helps prevent fraud and unauthorized access. Thereby protecting the financial institution and the customer. Banks, e-commerce platforms, and payment gateways are prime users of these cryptographic solutions, ensuring secure and seamless financial transactions. Symmetric ciphers are often employed to encrypt large datasets due to their high efficiency, while asymmetric cryptography secures communication channels. This dual approach ensures that even if an encrypted message is intercepted, it cannot be deciphered without the appropriate

keys.

Symmetric ciphers use the same key for encryption and decryption and are widely favored for their speed and efficiency. They excel at encrypting large volumes of data, making them ideal for file encryption, database security, and network protection applications. For example, tools like BitLocker and FileVault use symmetric ciphers to secure entire drives. It ensures that the data stored on these drives is protected from unauthorized access. Similarly, network security protocols such as VPNs utilize stream cipher to safeguard data transmitted over potentially insecure networks. It provides a secure communication channel for remote workers and corporate networks.

They are used in telecommunications to secure voice and video calls, ensuring that conversations remain private. Stream ciphers are also employed in wireless networks to encrypt data transmission. Moreover, stream ciphers' lightweight and efficient nature makes them ideal for securing data transmitted by Internet of Things (IoT) devices, which often have limited computational resources. Telecommunication companies, IoT manufacturers, and software developers rely on stream ciphers for real-time encryption. This ensures secure and continuous data streams in video streaming, online gaming, and smart device communication applications.

Stream ciphers are fundamental to modern cryptography, providing essential security mechanisms for various applications. These cryptographic tools are indispensable in today's digital world, from securing internet communication and financial transactions to protecting government and healthcare data. Ongoing development, analysis, and implementation of these cryptographic methods will be vital. It will allow us to tackle the evolving challenges of data security and privacy, ensuring that sensitive information stays secure in our increasingly interconnected digital world.

Many ciphers are designed based on Addition, Rotation, and exclusive OR (ARX) operations. The ARX consists of three basic operations: modular addition 2^n, bit rotations, and the Exclusive OR operations and ARX ciphers outperform in software [Ber08].

The addition is the core component of the ARX design principle. It provides nonlinearity and diffusion. Due to the bitwise nature of ARX, the rotation is used to speed up and balance the diffusion property. The basic operation in ARX runs constantly, providing resistance against timing attacks. However, it is susceptible to differential and differential-linear attack. When encryption speed is the main requirement, ARX ciphers can significantly boost performance. For example, VMWare View uses PCoIP (PC-over-IP) to send computer displays across a network. ARX-based ciphers, including Salsa 20 [Ber08], and ChaCha [B+08] were introduced in 2005 and 2008, respectively. In versions 4.5 and later, VMWare View uses Salsa 12. The Salsa 12 reaches speeds of around 20 Mbit/s, compared to AES, which is typically limited to around 7 Mbit/s [MP13]. While the ARX ciphers outperform in speed, their security

is poorly understood and needs further investigation. Salsa 20 and ChaCha are the widely deployed stream ciphers, and we briefly present the deployments below: Salsa 20 is deployed in the following applications: Linux and Chromium OS. It is also applied in password managers such as KeepassX, Freepass, McPass, and CurveProtect. It's used in different programming libraries of Java, Go, JavaScript, Python3, Ruby, Rust, SWIFT, and C. It is also used in VPN and tunneling software such as MLVPN, FASTD, PipeSocks, and Salsapipe.

Likewise, Salsa 20, ChaCha is deployed in a wide range of hardware and software. For instance, ChaCha is part of the TLS 1.3 cipher suit. In addition, it is part of WireGuard, Netcode, Noise, and QUIC protocols. It's applied in operating systems like Linux, Andriod, and Redox. It's used in Chrome, Firefox, and Safari. It's also part of many programming languages, such as PHP, NodeJS, JavaScript, and Java. Given the wide deployment of Salsa 20 and ChaCha stream ciphers, it's crucial to understand their security.

1.2 Existing Studies

The existing research studies mainly focus on the security of Salsa 20 and ChaCha stream ciphers using differential and differential-linear cryptanalysis methods. The attacks primarily consist of key-recovery attacks and distinguishers. Differential cryptanalysis examines how an input difference propagates through various rounds of a cipher. The success of the attack is measured by a bias, which indicates the non-random behavior of the cipher.

This method is one of the most widely used cryptanalysis approaches. Linear cryptanalysis, on the other hand, studies the approximation of modular addition via the XOR operation with a certain probability. Differential-linear cryptanalysis combines both differential and linear cryptanalysis techniques. It is regarded as a powerful approach for attacking the higher rounds of a cipher. Boomerang cryptanalysis combines shorter differentials with higher-probability differentials, which are often difficult to find in the higher rounds of a cipher.

Table 1.1: The Existing Key-Recovery Attacks on Salsa 20

Attack	Target	Time	Data	Reference
Differential	7	2^{151}	2^{26}	[AFK+08]
Differential	7	2^{148}	2^{24}	[SZFW12]
Differential-linear	7	2^{137}	2^{61}	[CM16]
Differential	8	2^{251}	2^{31}	[AFK+08]
Differential	8	2^{250}	2^{27}	[SZFW12]
Differential	8	$2^{245.5}$	$2^{22.4}$	[Mai16]
Differential-linear	8	$2^{244.9}$	2^{96}	[CM16]
Differential	8	$2^{243.6}$	$2^{30.4}$	[DS17]
Differential	8	$\mathbf{2^{241.62}}$	$\mathbf{2^{31.5}}$	Chapter 4

Table 1.2: The Existing Key-Recovery Attacks on ChaCha

Attack	Target	Time	Data	Reference
Differential	6	2^{139}	2^{30}	[AFK+08]
Differential-linear	6	$2^{127.5}$	$2^{37.5}$	[CM16]
Differential-linear	6	$2^{77.4}$	2^{58}	[BBC+22]
Differential	6	2^{136}	2^{28}	[SZFW12]
Differential-linear	6	$2^{102.2}$	2^{56}	[CN20]
Differential	7	2^{248}	2^{27}	[AFK+08]
Differential-linear	7	$2^{237.7}$	2^{96}	[CM16]
Differential-linear	7	$2^{230.86}$	$2^{48.8}$	[BBC+22]
Differential-linear	7	$2^{221.95}$	$2^{48.83}$	[DGSS22]
Differential	7	$2^{231.63}$	$2^{49.58}$	[MIM22]
Differential	7	$2^{210.3}$	$2^{103.3}$	[WLHL23]
Differential-linear	7	$2^{206.8}$	$2^{110.81}$	[BGG+23]
Differential	7.25	$2^{255.62}$	$2^{48.36}$	[MIM22]
Differential	7.25	$2^{254.011}$	$2^{51.81}$	Chapter 4

Table 1.3: The Existing Distinguishers of ChaCha

Attack	Target	Time	Data	Reference
Differential-linear	4	2^{6}	2^{6}	[CM16]
Differential-linear	5	2^{16}	2^{16}	[CM16]
Differential-linear	5	$2^{31.21}$	$2^{31.21}$	Chapter 5
Differential-linear	5.5	$2^{39.21}$	$2^{39.21}$	Chapter 5
Differential-linear	6	2^{116}	2^{116}	[CM16]
Differential-linear	6	2^{51}	2^{51}	[CSN21]
Differential-linear	6	$2^{47.21}$	$2^{47.21}$	Chapter 5
Differential-linear	7	2^{224}	2^{224}	[CSN21]
Differential-linear	7	2^{214}	2^{214}	[CPV+23a]
Differential	7	2^{207}	2^{207}	[DS23]
Differential-linear	7	$2^{166.89}$	$2^{166.89}$	[BGG+23]

All existing cryptanalysis methods applied to Salsa 20 and ChaCha stream ciphers have used differential and differential-linear methodologies. The general idea behind these attacks is to introduce differences in the initial state and then evaluate these differences at a target round to determine whether the cipher behaves like a random permutation. Any detected non-randomness can be exploited either to recover the key or to distinguish the cipher from a truly random function. While the current research domain covers both distinguishers and key-recovery attacks, it does not fully explore distinguishing attacks, which aim to differentiate the keystream of a symmetric cipher from that of a random source.

Table 1.4: The Distinguishing Attacks on ChaCha

Attack	Round	Time	Data	Reference
Differential-linear	5	$2^{33.21}$	$2^{33.21}$	Chapter 5
Differential-linear	5.5	$2^{63.21}$	$2^{63.21}$	Chapter 5
Differential-linear	6	$2^{87.21}$	$2^{87.21}$	Chapter 5

1.3 Motivations

To mount an effective attack, it is crucial to identify the attacking points which consist of the initial and final positions. It primarily focuses on determining the initial position first, followed by the exploration of the corresponding final position to identify potential attack pairs. However, this approach may not always yield the optimal attack pair due to the significant impact of the final position on the overall attack complexity. Therefore, a comprehensive analysis and careful selection of the final position are necessary.

This research evaluates whether identifying the final position first and subsequently determining the optimal initial position enhances the attack complexity. We intend to employ the differential attack methodology to assess the effectiveness of this approach. Arka [CM16] introduced the differential-linear cryptanalysis on Salsa 20 and ChaCha and later expanded by Coutinho [CPV+23a].

Building on this foundation, the differential-linear attacks reported in Tables 1.1, 1.2, and 1.3 predominantly utilize single-bit differences to construct and execute their attacks. Although most of the attacks on the ChaCha stream cipher presented in Tables 1.2 and 1.3 use the differential-linear attack, these studies mainly focused on enhancing the attack on ChaCha by improving linear approximation. This dissertation will evaluate the application of two main methods. First, we aim to enhance the differential-linear attack by considering the differential part of the attack.

Furthermore, we will assess the efficacy of the higher-order differential-linear attack on reduced rounds of ChaCha. We adopted Wagner's boomerang attack methodology as outlined in [Wag99]. The boomerang attack is a robust cryptanalytic technique that offers a unique approach to evaluating cryptographic algorithms' security, particularly ChaCha's permutation using the chosen plaintext and cipher text assumption. The boomerang attack leverages differential cryptanalysis in an advanced manner, allowing the examination of complex cryptographic structures that might not be susceptible to traditional differential attacks.

Integrating the boomerang attack into our suite of analysis methods provides a more comprehensive evaluation of the ChaCha security, employing the boomerang attack in our analysis of the ChaCha permutation function is essential for achieving a thorough and robust security evaluation. Salsa 20 and ChaCha stream ciphers have undergone extensive analysis against differential and differential-linear cryptanalysis. Researchers have demonstrated attacks on up to Salsa 8 and ChaCha 7.5.

1.4 Contributions

This dissertation studies the security of Salsa 20 and ChaCha stream ciphers. This work has further improved the existing research and introduced new techniques to attack Salsa 20 and ChaCha. The contribution of this dissertation is covered in three independent chapters, and we summarize it as follows:

- Nasratullah Ghafoori, Atsuko Miyaji, Ryoma Ito and Shotaro Miyashita . PNB based differential cryptanalysis of Salsa 20 and Chacha IEICE TRANSACTIONS on Information and Systems 106, no,9 1407-1422 2023 (Chapter 4)

- Nasratullah Ghafoori, Atsuko Miyaji. Higher-Order Differential-Linear Cryptanalysis of ChaCha Stream Cipher IEEE Access 2024.(Chapter 5)

- Nasratullah Ghafoori, Atsuko Miyaji. "The Boomerang Attack on ChaCha Stream Cipher Permutation. In 2024 6th International Conference on Computer Communication and the Internet (ICCCI), pp, 18-23 2024.(Chapter 6)

Chapter 4: Differential Cryptanalysis of Salsa 20 and ChaCha Based on PNB Analysis: This chapter explores the differential cryptanalysis of Salsa 20 and ChaCha stream ciphers. Our approach involves utilizing differential cryptanalysis and the analysis of PNBs on reduced rounds of Salsa 20. Initially, we thoroughly assess the neutrality measure of all keybits concerning the output differentials $\mathcal{OD}$. Subsequently, we identify the $\mathcal{OD}$ bit position with the highest neutrality measure and seek out the corresponding input differential $\mathcal{ID}$ with the most favorable differential bias. Taking into consideration the various factors, we propose an attack on Salsa 8, with a time complexity of $2^{241.62}$ and a data complexity of $2^{31.5}$.

Additionally, we introduce an attack targeting ChaCha7.25 rounds, with a time complexity estimated at $2^{254.011}$ and a data complexity of approximately $2^{51.81}$. Our work improves upon this differential attack on Salsa 8. We compare the result of our attack with the recent attack by [DS17] in Table 1.5.

Table 1.5: The Salsa 8 attack comparison

Time / Data	Reference
$2^{241.62}$/ $2^{31.5}$	Chapter 4
$2^{243.7}$/ $2^{30.4}$	[DS17]

We have used Table 1.6 to present the comparison of our attack with the recent attack introduced by Miyashita.

Table 1.6: The ChaCha 7.25 attack comparison

Time / Data	Reference
$2^{254.011}$/ $2^{51.81}$	Chapter 4
$2^{255.62}$/ $2^{48.36}$	[MIM22]

Chapter 5: Higher-Order Differential-linear Cryptanalysis ChaCha: This chapter examines the differential-linear and higher-order differential-linear cryptanalysis of the ChaCha. We have enhanced the differential-linear attack on ChaCha and conducted a study utilizing higher-order differential-linear cryptanalysis. Our research investigates the higher-order differentials and their application to ChaCha. The study further explores the effect of higher-order differential cryptanalysis on reduced rounds of ChaCha and evaluates the cipher's resistance to these attacks. Based on our recent journal publication [1], we computed the distinguisher of ChaCha 5, ChaCha 5.5,

[1]The journal paper was published in IEEE Access and its foundation of this chapter 5.

and ChaCha 6 with the complexity of $2^{31.21}$, $2^{39.21}$, and $2^{47.21}$, respectively. In addition to the journal results, chapter 5 introduces distinguishing attacks for ChaCha 5, ChaCha 5.5, and ChaCha 6 that incorporate the linear approximation of the final modular addition used in key generation. In addition, we studied the higher-order differential-linear attack on ChaCha 5, ChaCha 5.5, and ChaCha 6 and introduced a distinguishing attack with $2^{33.21}$, $2^{63.21}$ and $2^{87.21}$ complexities, respectively.

Moreover, we report larger biases that were previously undocumented for internal rounds beyond 3.5 rounds. Additionally, our research reported new linear approximations of specific bits from the 4th to the 6th rounds.

Table 1.7: The ChaCha 6 distinguisher comparison

Time / Data	Reference
$2^{47.21}$/ $2^{47.21}$	Chapter 4
2^{51}/ 2^{51}	[CM16]

Chapter 6: Boomerang Cryptanalysis of ChaCha Permutation: The ChaCha stream cipher has been extensively analyzed for its resistance against traditional forms of differential and differential-linear attacks, but its susceptibility to different variants of the cryptanalysis method has remained ambiguous over the past decade. In this study, we conduct the first-ever boomerang cryptanalysis of ChaCha ' s permutation function.

Our analysis demonstrates that in certain attack positions within ChaCha, the probability may escalate to p^2, thereby offering an additional enhancement to the attack complexity of ChaCha. Additionally, we present an algorithm designed for executing boomerang attacks on the ChaCha stream cipher. To demonstrate the potency of boomerang cryptanalysis on the ChaCha permutation function, we apply it to target ChaCha 6 and ChaCha 7. Our findings reveal that a boomerang attack requires a total of approximately $2^{4.04}$ and $2^{5.99}$ adaptively chosen plaintext and ciphertext combinations to effectively distinguish ChaCha 6 and ChaCha 7 from a random permutation, respectively.

Figure 1.1 shows the dissertation structure. The diagram illustrates the structure of this dissertation. It indicates the sequential progression of topics. The direction of the arrows signifies the order in which each chapter is addressed, with each topic building upon the previous one.

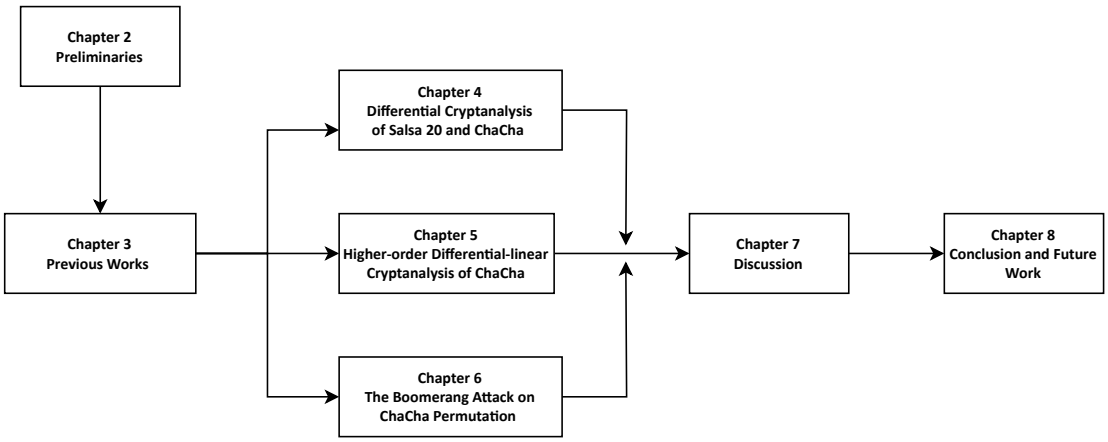

Figure 1.1: Thesis Overview

1.5 Organization

The main focus of this dissertation is to study and analyze the security of Salsa 20 and ChaCha stream ciphers and to present their potential vulnerabilities. We organized the dissertation as follows: Chapter 2 explains the structure of the Salsa 20 and ChaCha and introduces the general framework for differential attacks, differential-linear attacks, higher-order differential attacks, and boomerang attacks.

Chapter 3 comprehensively studies existing attacks on Salsa 20 and ChaCha, detailing their methodologies and outcomes.

Chapter 4 presents our attacks on Salsa 20 and ChaCha, based on a detailed analysis of Probabilistic Neutral Bits (PNBs).

Chapter 5 discusses our differential-linear and higher-order differential-linear distinguishers on ChaCha, and additionally, how we extended these to distinguishing attacks.

Chapter 6 introduces the first application of the boomerang attack on the ChaCha permutation function, demonstrating how we evaluated its security in both chosen plaintext and ciphertext scenarios.

Chapter 2

Preliminaries

This chapter provides the foundational concepts for understanding the key terminologies and symbols used throughout this work. We introduce the principles of symmetric key cryptography and define the Salsa 20 and ChaCha ciphers, which are central to this study. We also outline the cryptographic approaches employed in the research and present several important lemmas underpinning the analysis.

2.1 Cryptography Primitives

Cryptographic primitives include basic mathematical functions or algorithms essential for constructing various cryptographic frameworks and structures that ensure confidentiality, integrity, authentication, and non-repudiation. These are divided into three main subclasses: public key cryptography, symmetric key cryptography, and hash functions.

Symmetric Key Cryptography

Symmetric key cryptography functions depend on a shared secret key. It mainly encrypts data between the sender and receiver using the same shared key, providing data confidentiality. The security of symmetric key cryptography relies on the confidentiality of the encryption key, which must be kept secret and available only to authorized parties. Symmetric key cryptography is further classified into stream ciphers and block ciphers.

Stream cipher

Stream ciphers are cryptographic functions that take keybits, initial vectors, and sometimes constants to produce random keystreams. The XOR operation encrypts and decrypts data using the keystream generated by stream cipher functions.

Definition 2.1 ([DK15]). *Let L represent a set of keys and M denote a set of plaintexts, where the elements of M are referred to as characters. A stream cipher $E{:}K^{\times}M^{*} \rightarrow C^{*}$ can be defined in this context as follows:*

$$E^{*}(k, m) := c := c_1 c_2 c_3 \cdots$$

where E^{*} is a function that accepts a key k and a plaintext m, resulting in the generation of a ciphertext c.

Block Cipher

Block ciphers typically encrypt 64 or 128 bits under a key k of a certain size. Encryption keys are considered to be randomly chosen. The block cipher map and input of n size to an output of n under a random key k. Encryption is reversible, and the key can be decrypted using the same key k.

Definition 2.2 ([Alk16]). *A block cipher is an invertible mapping that operates on inputs and outputs of block size n bits, using a key of size k bits. This mapping is defined as $\mathcal{MK} \rightarrow C$, where $\mathcal{M} \in E_2^n$ represents the message space, $C \in E_2^n$ denotes the ciphertext space, and $\mathcal{K} \in E_2^k$ indicates the key space. The inverse mapping is given by the decryption function $E^{-1} : \mathcal{CK} \rightarrow M$.*

2.2 Specification of Salsa 20

The family of Salsa 20 stream cipher [Ber08] was proposed by Daniel J. Bernstein in 2005. The original version of Salsa 20 has 20 rounds with 256 keybit security against key recovery attacks. Salsa 12 and Salsa 8 were proposed as reduced rounds of Salsa 20, where speed is more important than security. In addition, the Salsa 20 allows the 128−keybits security as an option. However, it is not recommended. In [Ber08] Bernstein reported that Salsa 20 encrypts 3.93 Core-2 cycles/byte, Salsa 12 encrypts data in 2.80 Core-2 cycles/byte, and Salsa 8 encrypts data in 1.88 Core-2 cycles/byte which makes it one of the fastest symmetric ciphers.

The Salsa 20 algorithm consists of a lengthy sequence of three basic operations:

- Modular addition of two 32-bit words denoted as $a + b \bmod 2^{32}$.
- The XOR of two 32-bit words denoted as $a \oplus b$.
- Constant distance 32-bit rotations denoted as $a \lll b$, which rotates the word a by b bits to the left. The b is always constant.

Table 2.1: Symbols and Notations

Notation	Description
X	A 4×4 matrix composed of 16 words.
$X^{(0)}$	The ChaCha initial state matrix.
$X'^{(0)}$	The corresponding matrix with a single-bit difference at the $x_{i,j}$ position.
$X^{(R)}$	The matrix after R rounds of ChaCha.
$X^{(r)}$	The matrix after r rounds of ChaCha, where $R > r$.
$x_i^{(R)}$	The i-th word of state matrix $X^{(R)}$.
$\Theta(x, y)$	The carry function for the sum $x + y$.
$\mathcal{ID}, \mathcal{OD}$	Input difference, output difference
$\Delta X_i^{(n)} j$	The difference in the j-th bit of the i-th word after n rounds.
$Pr(E)$	The likelihood of event E occurring.
$x_{i,j}^{(R)}$	The j-th bit of i-th word of matrix $X^{(R)}$.
$x + y$	The word wise modular addition of x and y.
$x - y$	The word wise subtraction of x and y.
$x \oplus y$	The bit-wise XOR operation between the words x and y.
$x \lll n$	The left rotation of the word x by n bits.
Δx	The XOR difference between the words x and x'.
ε_d	Differential bias.
ε_a	Inverse bias.
ε_L	Linear bias.
$\varepsilon_d \cdot \varepsilon_L^2$	Differential-Linear bias.
γ_i	The neutrality measure of the i-th key bit.
ChaCha n	The n-th round of the ChaCha stream cipher.

In the first round, columns operate independently of each other. Similarly, in the following round, rows operate independently. For instance, the words (i, j) remain unaffected by an input difference until the third round. As a result, each word of Salsa 20 affects each other from the 4-th round. The [Ber08] stated that the rotation distances of $7, 11, 13$, and 18 in Salsa 20 are selected to effectively propagate changes across various bit positions within a few rounds.

The designer introduced the 20-round Salsa 20 stream cipher to the ECRYPT Stream Cipher Project, eSTREAM[1], as a contender for stream ciphers intended for software applications demanding high throughput and hardware applications with limited resources. In September 2008, the eSTREAM portfolio was finalized, with the 12-round version of Salsa 20, known as Salsa 12, being chosen as one of the finalists for the software category in the portfolio. The security of the ARX structure depends on modular addition, which creates non-linearity.

The Salsa 20 stream cipher generates a keystream block comprising 16 words, each 32 bits in size, through the following three steps:

[1]https://www.ecrypt.eu.org/stream/

Step 1. The first state matrix $X^{(0)}$, which is a 4×4 matrix, is established using a 256-bit confidential key $k = (k_0, k_1, \ldots, k_7)$, a 64-bit nonce $v = (v_0, v_1)$, a 64-bit block counter $t = (t_0, t_1)$, and four 32-bit constants $c = (c_0, c_1, c_2, c_3)$, where the constants are defined as $c_0 =$ 0x61707865, $c_1 =$ 0x3320646e, $c_2 =$ 0x79622d32, and $c_3 =$ 0x6b206574.

Following this setup, we derive the initial state matrix $X^{(0)}$ as follows:

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

Step 2. The Salsa 20 round function consists of four quarter-round functions. In this process, a vector $(x_a^{(r)}, x_b^{(r)}, x_c^{(r)}, x_d^{(r)})$ from the internal state matrix $X^{(r)}$ is updated through a series of sequential computations:

$$\begin{aligned} x_b^{(r+1)} &= ((x_a^{(r)} + x_d^{(r)}) \lll 7) \oplus x_b^{(r)}, \\ x_c^{(r+1)} &= ((x_b^{(r+1)} + x_a^{(r)}) \lll 9) \oplus x_c^{(r)}, \\ x_d^{(r+1)} &= ((x_c^{(r+1)} + x_b^{(r+1)}) \lll 13) \oplus x_d^{(r)}, \\ x_a^{(r+1)} &= ((x_d^{(r+1)} + x_c^{(r+1)}) \lll 18) \oplus x_a^{(r)}. \end{aligned} \tag{2.1}$$

The symbols '$+$', '$\lll$', and '$\oplus$' denote word-wise modular addition, bit-wise left rotation, and bit-wise XOR, respectively. In odd-numbered rounds, known as column-rounds, the quarter-round function is applied to the following four column vectors: $(x_0^{(r)}, x_4^{(r)}, x_8^{(r)}, x_{12}^{(r)})$, $(x_5^{(r)}, x_9^{(r)}, x_{13}^{(r)}, x_1^{(r)})$, $(x_{10}^{(r)}, x_{14}^{(r)}, x_2^{(r)}, x_6^{(r)})$, and $(x_{15}^{(r)}, x_3^{(r)}, x_7^{(r)}, x_{11}^{(r)})$. In even-numbered rounds, referred to as row-rounds, the quarter-round function is applied to these four row vectors: $(x_0^{(r)}, x_1^{(r)}, x_2^{(r)}, x_3^{(r)})$, $(x_5^{(r)}, x_6^{(r)}, x_7^{(r)}, x_4^{(r)})$, $(x_{10}^{(r)}, x_{11}^{(r)}, x_8^{(r)}, x_9^{(r)})$, and $(x_{15}^{(r)}, x_{12}^{(r)}, x_{13}^{(r)}, x_{14}^{(r)})$.

Step 3. A 512-bit keystream block is generated as $Z = X^{(0)} + X^{(R)}$, where R denotes the last round. The initial version of the Salsa 20 stream cipher, known simply as Salsa 20, uses $R = 20$ rounds. However, the version recognized as one of the finalists in the eSTREAM software portfolio [eP] is Salsa 12, where $R = 12$.

The round function in Salsa 20 is invertible, which implies that a vector $(x_a^{(r+1)}, x_b^{(r+1)}, x_c^{(r+1)}, x_d^{(r+1)})$ within the internal state matrix $x^{(r+1)}$ can be reversed through the following sequential operations:

$$\begin{aligned} x_a^{(r)} &= ((x_d^{(r+1)} + x_c^{(r+1)}) \lll 18) \oplus x_a^{(r+1)}, \\ x_d^{(r)} &= ((x_c^{(r+1)} + x_b^{(r+1)}) \lll 13) \oplus x_d^{(r+1)}, \\ x_c^{(r)} &= ((x_b^{(r+1)} + x_a^{(r)}) \lll 9) \oplus x_c^{(r+1)}, \\ x_b^{(r)} &= ((x_a^{(r)} + x_d^{(r)}) \lll 7) \oplus x_b^{(r+1)}. \end{aligned} \tag{2.2}$$

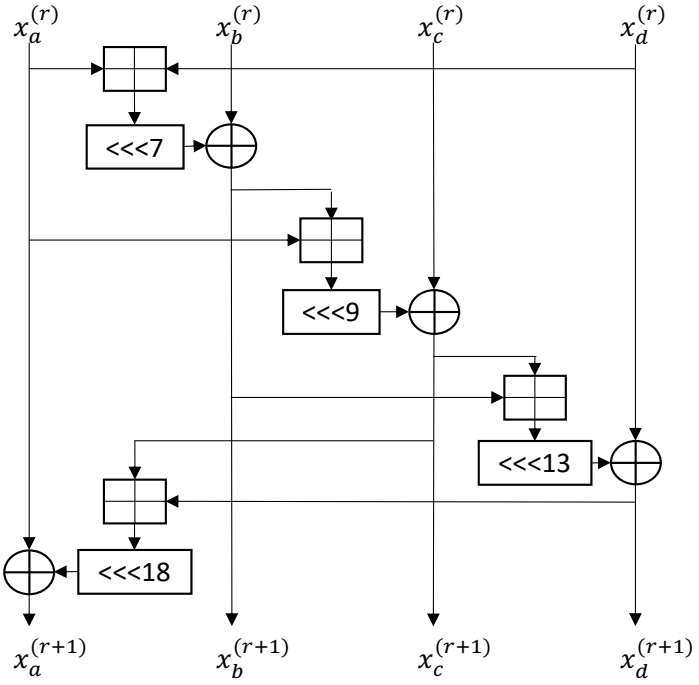

Figure 2.1: Schematic of Salsa 20

2.3 ChaCha Stream Cipher

ChaCha stream cipher [B+08] is a variant of Salsa 20. It is designed using ARX based on the same principles as its predecessor, Salsa 20. ChaCha was designed to improve the diffusion per round and increase resistance against cryptanalysis approaches. The diffusion is not increased by adding additional operations to the structure of the ChaCha stream cipher. It employs 16 additions, 16 XORs, and 16 constant-distance rotations of 32-bit words.

ChaCha [B+08] involves three steps to produce a keystream block containing 16 words, with each word being 32 bits in size. Unlike Salsa 20, the structure of the ChaCha stream cipher ensures that each input word will affect each output word, which improves the diffusion property of Salsa 20. The order of words in the ChaCha stream cipher differs from Salsa 20, as the attacker-controlled words are placed at the bottom of the matrix. To generate the keystream, ChaCha stream ciphers run the following steps.

Step 1. To produce a 512-bit keystream, the ChaCha algorithm initializes the state matrix $X^{(0)}$ of size 4×4 using a 256-bit key $k = (k_0, k_1, \ldots, k_7)$, a 96-bit nonce $v = (v_0, v_1, v_2)$, a 32-bit block counter t_0, and four predefined 32-bit constants $c = (c_0, c_1, c_2, c_3)$, specifically $c_0 =$ 0x61707865, $c_1 =$ 0x3320646e, $c_2 =$ 0x79622d32, and $c_3 =$ 0x6b206574. After setting up, the resulting initial state matrix is as follows:

$$X^{(0)} = \begin{vmatrix} x_0^{(0)} & x_1^{(0)} & x_2^{(0)} & x_3^{(0)} \\ x_4^{(0)} & x_5^{(0)} & x_6^{(0)} & x_7^{(0)} \\ x_8^{(0)} & x_9^{(0)} & x_{10}^{(0)} & x_{11}^{(0)} \\ x_{12}^{(0)} & x_{13}^{(0)} & x_{14}^{(0)} & x_{15}^{(0)} \end{vmatrix} = \begin{vmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & v_0 & v_1 & v_2 \end{vmatrix}.$$

Step 2. The round function of ChaCha includes four concurrent executions of the quarterround function. In this process, a vector $(x_a^{(r)}, x_b^{(r)}, x_c^{(r)}, x_d^{(r)})$ within the intermediate matrix $X^{(r)}$ is altered by performing the following steps in sequence:

$$\begin{aligned} x_{a'}^{(r)} &= x_a^{(r)} + x_b^{(r)} & x_a^{(r+1)} &= x_{a'}^{(r)} + x_{b''}^{(r)} \\ x_{d'}^{(r)} &= x_d^{(r)} \oplus x_{a'}^{(r)} & x_{d'''}^{(r)} &= x_{d''}^{(r)} \oplus x_a^{(r+1)} \\ x_{d''}^{(r)} &= x_{d'}^{(r)} \lll 16 & x_d^{(r+1)} &= x_{d'''}^{(r)} \lll 8 \\ x_{c'}^{(r)} &= x_c^{(r)} + x_{d''}^{(r)} & x_c^{(r+1)} &= x_{c'}^{(r)} + x_d^{(r+1)} \\ x_{b'}^{(r)} &= x_b^{(r)} \oplus x_{c'}^{(r)} & x_{b'''}^{(r)} &= x_{b''}^{(r)} \oplus x_c^{(r+1)} \\ x_{b''}^{(r)} &= x_{b'}^{(r)} \lll 12 & x_b^{(r+1)} &= x_{b'''}^{(r)} \lll 7. \end{aligned} \tag{2.3}$$

The notations "+", "$\oplus$", and "$\lll$" stand for modular addition performed on words, bitwise XOR operation, and bitwise left rotation, respectively. For odd-numbered rounds, which are called columnrounds, the quarterround function is applied to the following four column vectors: $(x_0^{(r)}, x_4^{(r)}, x_8^{(r)}, x_{12}^{(r)})$, $(x_1^{(r)}, x_5^{(r)}, x_9^{(r)}, x_{13}^{(r)})$, $(x_2^{(r)}, x_6^{(r)}, x_{10}^{(r)}, x_{14}^{(r)})$, and $(x_3^{(r)}, x_7^{(r)}, x_{11}^{(r)}, x_{15}^{(r)})$. For even-numbered rounds, which are called diagonalrounds, the quarterround function is applied to the following four diagonal vectors: $(x_0^{(r)}, x_5^{(r)}, x_{10}^{(r)}, x_{15}^{(r)})$, $(x_1^{(r)}, x_6^{(r)}, x_{11}^{(r)}, x_{12}^{(r)})$, $(x_2^{(r)}, x_7^{(r)}, x_8^{(r)}, x_{13}^{(r)})$, and $(x_3^{(r)}$,
$x_4^{(r)}, x_9^{(r)}, x_{14}^{(r)})$.

Step 3. A 512-bit keystream block is derived as $Z = X^{(0)} + X^{(R)}$, with R indicating the last round. The original ChaCha consists of $R = 20$ rounds, and ChaCha20/R refers to the variant with fewer rounds.

The round function is invertible. This means that a vector $(x_a^{(r+1)}, x_b^{(r+1)}, x_c^{(r+1)}, x_d^{(r+1)})$ in the internal state matrix $X^{(r+1)}$ can be reversed by executing the following steps in sequence:

$$\begin{aligned} &x_{b'''}^{(r)} = x_b^{(r+1)} \lll 25, \quad x_{b''}^{(r)} = x_{b'''}^{(r)} \oplus x_c^{(r+1)}, \quad x_{c'}^{(r)} = x_c^{(r+1)} - x_d^{(r+1)}, \\ &x_{d'''}^{(r)} = x_d^{(r+1)} \lll 24, \quad x_{d''}^{(r)} = x_{d'''}^{(r)} \oplus x_a^{(r+1)}, \quad x_{a'}^{(r)} = x_a^{(r+1)} - x_{b''}^{(r)}, \\ &x_{b'}^{(r)} = x_{b''}^{(r)} \lll 20, \quad x_b^{(r)} = x_{b'}^{(r)} \oplus x_{c'}^{(r)}, \quad x_c^{(r)} = x_{c'}^{(r)} - x_{d''}^{(r)}, \\ &x_{d'}^{(r)} = x_{d''}^{(r)} \lll 16, \quad x_d^{(r)} = x_{d'}^{(r)} \oplus x_{a'}^{(r)}, \quad x_a^{(r)} = x_{a'}^{(r)} - x_b^{(r)}. \end{aligned} \tag{2.4}$$

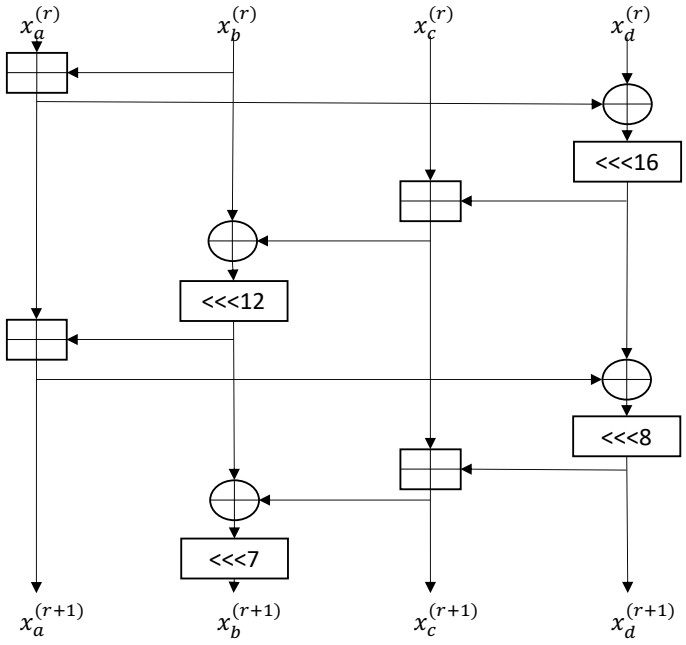

Figure 2.2: Schematic of ChaCha

2.4 Cryptanalysis Approaches

As mentioned in the prior chapter, cryptanalysis is used to evaluate the security of cryptographic primitives. The cryptanalysis will follow one or more attack models. The role of the cryptanalysts' community is to conduct different types of attacks and push the attack to its limits. Some cryptanalysis methods treat the cipher as a black box and attack the cipher. However, some methods target the specific weakness in a cipher. All the existing cryptanalysis approaches aim to understand the security of cryptographic primitives.

In this section, we describe the most prominent cryptanalysis attack. precisely, cryptanalysis is the theoretical and practical study of techniques for analyzing the security of cryptographic primitives. It is an important process for building trust in cryptographic primitives. The cryptanalysis process involves mathematics, computer science, and logic to identify vulnerabilities within a cryptographic scheme. Comprehensive cryptanalysis research helps cryptographers to understand the security of a cipher better.

Cryptanalysis plays a critical role in evaluating the robustness of cryptographic systems. It ensures the integrity and confidentiality of information. Cryptanalysts are ideally looking to recover the secret key; however, there are many types of cryptanalysis techniques, as follows:

- **Complete Break:** This scenario involves an attack that fully recovers the encryption key.

- **Reduced Complexity:** The attack can retrieve the secret key more efficiently than a brute-force search.
- **Distinguishing Attack:** The attacker can tell the difference between the cipher and a random permutation.

To conduct any of the attacks above, we assume that the cryptanalyst might have access to a certain amount of information [KR11], called attack models. The choice of attack model by the attacker depends on cipher structure, time complexity, data complexity, and success probability. The attack models are classified as follows:

- **Chosen Plaintext attack:** The chosen plaintext attack is an attack scenario where the adversary can select certain plaintext and analyze the corresponding ciphertext. If a cipher is susceptible to a known plaintext attack, it is also inherently vulnerable to a chosen plaintext attack. However, the reverse isn't always true [Bir11].
- **Adaptively Chosen plaintext ciphertext attack:** Adaptive chosen-ciphertext scenario mount attack on cryptographic protocols enable an attacker to decrypt a ciphertext and obtain the plaintext through the submission of a series of chosen-ciphertexts to an oracle [MZ06].

To mount a successful attack, the cryptanalysts' community looks for the following three important factors [KR11].

- **Time complexity:** The first and foremost factor required to measure an attack's success is the attack's time complexity. It is the time or number of encryption to attack a cipher. In some cases, it is considered the only factor to measure the success of an attack. For instance, an attack with a time complexity of 2^{40} is considered more effective than an attack with a time complexity of 2^{50}.
- **Data Complexity:** The amount of data is important to measure the effectiveness of an attack. If the attack's data complexity is excessively high, it becomes impractical to execute. Thus, managing data complexity is essential for practical attacks.

When evaluating the effectiveness of a cryptographic attack, the availability and type of data are crucial factors in determining its success. Considering the type and amount of data involved in the attack, cryptanalysts have introduced numerous types of cryptanalysis approaches.

Some attacks achieve a total break, while others involve complexity deduction or introduce distinguisher. The cipher's security depends on the secret key k in symmetric cryptography. The easiest way to access the secret key is by guessing it. If the key

length is k, there are 2^k possible combinations, and the probability of guessing the correct key is 2^{-k}.

Moreover, an exhaustive search is a systematic method to obtain the secret key. Regardless of the security strength of a cipher, an exhaustive search can be used to find the key. To defend against exhaustive searches, the key length is vital in maintaining cipher security [KR11].

When ciphers withstand exhaustive searches, the cryptanalysis community employs various approaches to reduce the complexity of brute-force attacks. For example, suppose a cipher key requires 2^{128} key bits for an exhaustive search. In that case, cryptanalysts introduce approaches to reduce the exhaustive search range to 2^{128-d}, where d represents the deduction resulting from a cryptanalysis approach. Drawing from Shannon's work [Sha48], [Sha49], a cipher must not be distinguishable from a random permutation. If a cryptanalysis approach can efficiently distinguish a cipher from a random permutation in terms of time and data complexity, this is also considered an attack on ciphers.

2.4.1 Differential Cryptanalysis

In 1991, Biham and Shamir presented the differential cryptanalysis [BS91] as an adversary model to attack the Data Encryption Standard (DES) [S+99]. Over the last decades, differential cryptanalysis has been the main tool for analyzing the security of different ciphers.

Differential cryptanalysis is a chosen plain text attack that studies the impact of a difference in plaintext on the difference of ciphertext. The differences can be used to recover the secret key with certain probabilities. We need many plaintext pairs with an exact difference and the resultant ciphertext to mount a differential attack. The exclusive OR operation is used as a difference operator. Over the years, the variation of differential cryptanalysis was introduced.

These variations include truncated differential cryptanalysis [Knu95], higher-order differential cryptanalysis [Knu95], boomerang attacks [Wag99], differential-linear attacks [BS91], and impossible differential cryptanalysis [Knu98].

Differential Probability The differential probability of an encryption function f is computed over the function f which is shown as a pair of (Δ_x, Δ_y) or alternatively (α, β) where Δ_x, or α show the input difference $\mathcal{ID}$ and Δ_y, or β show the output difference $\mathcal{OD}$.

Definition 2.3 ([BS91], [LWD04]). *The differential probability (DP) of a differential relation for the function f, represented as (α, β), is calculated as*

$$DP(\alpha, \beta) = 2^{-n}\#\{X \in \mathbb{F}_2^n | (f(x \oplus \alpha) = f(x) \oplus \beta)\}.$$

Where n shows the size of function f, x indicates the input of function f. The differential probability suggests the number of possible input/output pairs that satisfy the differential relation over the function f.

Definition 2.4 ([LWD04]). *Limpa studied the XOR differential likelihood of addition, indicated as xdp^+, and the additive differential likelihood of XOR, expressed as $adp^\oplus$. The differential probability (DP) of addition modulo 2^n describes the probability that a particular input difference leads to a defined output difference.*

$$DP^+(\alpha, \beta) := P_{x,y}[(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta))].$$

The most common operator for computing the differential probability of a function f is the exclusive OR (XOR) operator $\oplus$. However, we could also consider the additive differential probability of XOR denoted as $adp^\oplus$. The XOR operator can neutralize the effect of key bits applied during the encryption process. For instance, given a pair of plaintext and ciphertext $(c, p), (\hat{c}, \hat{p})$ where $c = f(m) = m \oplus k$, and k is the encryption key, we have $c = m \oplus k$ and $\hat{c} = \hat{m} \oplus k$. By applying the XOR operator, we obtain $c \oplus \hat{c} = m \oplus \hat{m} \oplus k \oplus k = m \oplus \hat{m}$. This operation effectively eliminates the influence of the encryption key k [Alk16].

The differential probability of an encryption function f is determined solely by the non-linear element of a cipher. For instance, in Substitution-Permutation Networks (SPNs), the S-box reflects the differential probability. In contrast, in Add-Rotate-XOR (ARX) ciphers, the modular addition reflects the differential with a certain probability. The scenario differs in the linear element, where the differential probability always occurs with a probability of 1.

Differential Probability of Iterated Ciphers The iterative ciphers encrypt a plaintext by applying the same basic operations in multiple rounds. Each round consists of specific cryptographic operations such as modular additions, exclusive OR, and more. The output ciphertext from each round serves as the input for the subsequent round, with this procedure continuing for a predetermined number of rounds.

Definition 2.5 ([LMM91], [Alk16]). *An iterated cipher with the round function $f = f_{k_i}(X)$ is called a Markov cipher if there exists a difference operation Δ such that $Pr(\Delta y = \beta \mid \Delta x = \alpha, x = y)$ remains unaffected by γ for all input and output difference masks α and β, provided that the round key k_i is selected uniformly at random.*

The differential characteristics are initially searched over one round and then expanded over the target round. This principle is known as differential characteristic (DC).

Definition 2.6 ([LMM91], [Alk16]). *A differential characteristic (DC) refers to a series of intermediate differences passing through various encryption stages at each analyzed round.*

$$\Delta_0 \rightarrow R_1\Delta_1 \rightarrow \cdots \rightarrow \Delta_{r-1} \rightarrow R_r \rightarrow \Delta_r$$

An initial input difference and a set of output differences from each step determine the sequence.

A Single element of the differential characteristic sequence is denoted as $\alpha \xrightarrow{R} \beta$ where α indicates the input difference or $\mathcal{ID}$ and β denotes the output difference or $\mathcal{OD}$ over the number rounds R of a cipher E.

$$DP(DC) = Pr(\Delta_0 \rightarrow R_1 \rightarrow \Delta_1 \rightarrow \cdots \rightarrow \Delta_{r-1} \rightarrow R_r \rightarrow \Delta_r) = \prod_{i=1}^{r} Pr(\Delta_{i-1} \rightarrow R_i \rightarrow \Delta_i).$$

The differential probability of each round is assumed to be independent, and the overall differential probability over multiple rounds is the product of each differential characteristic at each round. When a single-round differential probability $\alpha = \Delta_0 \xrightarrow{r-round} \beta = \Delta_R$ is too small to be considered, cryptanalysts utilize the differential probability over multiple rounds $\alpha \xrightarrow{R-rounds} \beta$ which is the collection of several rounds of differentials. This is also called differential trails.

In such a case, the attacker can define a single input differential $\mathcal{ID}$ at the initial point and an output differential $\mathcal{OD}$ point after specific number rounds [KR11]. The differential probability of such cases is defined as:

$$DP(\alpha, \beta) = Pr(\Delta_r = \beta | \Delta_0 = \alpha) = \sum_{DC \in (\alpha,\beta)} DP(DC).$$

In practice, the cryptanalysts search for the input and output differential position, resulting in the highest possible differential probability, subsequently reducing the attack complexity. This is called Maximum Differential Probability. The Difference Distribution Table (DDT) helps cryptanalysts determine the maximum differential probability. Since S-boxes usually operate with 8 or $4-bit$ word sizes, analyzing their differential difference distribution table (DDT) is straightforward.

However, in the case of ARX algorithms, creating a DDT for this operation with $n-bit$ word sizes needs $2^{3n} \times 4$ bytes of memory [BV14]. This is impractical for the usual 32-bit word size, such as ChaCha and Salsa 20, making it difficult to find a proper differential position to mount an attack. As a result, the cryptanalyst community proposed different differential search methodologies, including [HW19, BVLC16, SHY16].

Cryptographic Importance of Derivatives

As discussed in Section 2.4.1, the basic idea of differential cryptanalysis relies on differential probability. A differential pair (α, β) where α is the input difference Δx of input blocks x and x', and β is the output difference Δy of the output blocks y and y' where $\Delta x = \alpha$ and $\Delta y = \beta$. If we define the difference by the operation $+$, then:

$$P(\Delta y = \beta | \Delta x = \alpha) = P(f(x + \alpha) - f(x) = \beta) = P(\Delta_\alpha f = \beta).$$

Proposition 1 ([Lai94]). *The probability of a differential* (α, β) *is that the first derivative of function* $f(x)$ *at point* α *takes on value* β *when* x *is uniformly random.*

The success of differential cryptanalysis is based on the fact that an output difference β of an input difference at the initial round of cipher can be anticipated with a higher probability. Higher-order derivatives can generalize the basic concept of differential cryptanalysis by having more than one input difference at the initial plaintext to recover the secret key.

For instance, the differential analysis uses an input difference α; however, with the application of higher-order differentials, we can use the α and λ to predict the occurrence of the output difference β [Lai94]. In 2011, Ming [DL11] laid the foundation for higher-order differentials, showing that higher-order differential cryptanalysis is based on higher-order derivatives of Boolean functions and separated the cryptanalysis framework into two phases: *Offline Selection* and *Online Attack*. In the offline selection phase, the attacker selects the derivative points (e.g., IVs and inputs) conducive to obtaining the attack functions in the subsequent online phase. In the online phase, the attacker either tries to recover the secret key or distinguish the cipher from a random permutation.

The higher-order differential cryptanalysis framework extends attacks to higher-order derivatives and enhances the comprehension of these attacks. It motivates cryptanalysts to explore higher-order differential cryptanalysis techniques further in the future. Knudsen [Knu95] introduced higher-order differential cryptanalysis using higher derivatives. This adversary model extends the concept of differential cryptanalysis to attack ciphers. Knudsen demonstrated the use of truncated and higher-order differentials, revealing that certain ciphers secure against traditional differential cryptanalysis may be susceptible to higher-order differential attacks.

Numerous researchers have utilized higher-order differential cryptanalysis to assess the security of various ciphers. Zhu [ZCL10] developed a cryptanalysis tool for evaluating the security of block ciphers based on Boolean algebra and proposed an algorithm to accelerate cryptanalysis. Shi [SZFW12] employed the second-order differential cryptanalysis to the Salsa 20 and ChaCha. Shi described the second-order differential as follows: Let X be the initial state matrix, and X_1, X_2, and X_3 be related

state matrices with a single-bit input difference $[\Delta_{ij}^{(0)}] = 1$ in X_1, a single-bit input difference $[\Delta_{mn}^{(0)}] = 1$ in X_2, and a double-bit input difference $[\Delta_{ij}^{(0)}] = 1$, $[\Delta_{mn}^{(0)}] = 1$, respectively. According to Shi [SZFW12], $(i-m)^2 + (j-m)^2 = 0$ should not hold. The single-bit output difference $[\Delta_{pq}^{(r)}] = 1$ after r internal rounds can be determined as follows:

$$[\Delta_p^{(r)}]_q = [X_p^{(r)}]_q \oplus [X_{1,p}^{(r)}]_q \oplus [X_{2,p}^{(r)}]_q \oplus [X_{p+1}^{(r)}]_q. \tag{2.5}$$

The second-order $\mathcal{ID}$ is dented by:

$$([X_p^{(r)}]_q | X_i^{(0)}]_j, [X_m^{(0)}]_n).$$

The bias ε_d is calculated as

$$\Pr\Big([\Delta_p^{(r)}]q = 1 | [\Delta_i^{(0)}]_j, [\Delta_m^{(0)}]_n\Big) = \frac{1}{2}(1 + \varepsilon_d). \tag{2.6}$$

Truncated Differential Cryptanalsis Knudsen [Knu95] introduced truncated differential cryptanalysis as a variation of differential cryptanalysis. As discussed, traditional cryptanalysis looks for a differential $\alpha \rightarrow \beta$ with a high probability. The attacker aims to extract the secret key information with an input difference $\mathcal{ID}$ and check whether the resultant ciphertext after a specific number of rounds has an output difference $\mathcal{OD}$ β.

The truncated differential constructs differentials over a few bits of input/output differences. It demonstrates that to construct a differential over r rounds, it is unnecessary to know the full n-bit differentials, and the attacker does not fully observe the differences through the full n bits but rather focuses on a specific part of the cipher. Sometimes, a single-bit differential suffices to mount a truncated differential attack on a cipher. As a result, a differential that anticipates a part of an n-bit value is called truncated differential cryptanalysis. red

A differential attack employing truncated differentials exhibits a complexity of $2L$ chosen plaintexts and an execution time approximately $L \times 2^{2n}$, where L denotes the smallest integer satisfying $(W)^L \leq 2^{-2n}$. Here, L does not exceed $2n + 1$, and W represents the fraction of possible output differences. Truncated differential cryptanalysis is employed to attack different ciphers, including SAFER [KB96], IDEA [KR97], and Crypton [KHL+04], among others.

Higher Order Differential Cryptanalysis Lai [Lai94] presented the notion of higher-order derivatives for functions with multiple variables. He examined the possibility of extending first-order differential cryptanalysis by incorporating higher-order derivatives. Motivated by boomerang and differential-linear cryptanalysis, Biham [BDK05] explored various combined attack strategies. These approaches include differential-

bilinear, higher-order differential-linear (HDL), and boomerang attacks. Now, let's revisit the basic definitions.

Definition 2.7 ([Lai94]). *Let $(S,+)$ and $(T,+)$ be two Abelian groups. For a function $f : S \to T$, the derivative of f at a point $a \in S$ is defined as:*

$$\Delta_a f(x) = f(x+a) - f(x).$$

The i-th derivative of the function f at the point $(a_1, a_2, \ldots, a_i)$ is defined as:

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \Delta_{a_i}(\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} f(x)),$$

where $\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} f(x)$ denotes the $(i-1)$-th derivative of f at $(a_1, \ldots, a_{i-1})$. The 0-th derivative of $f(x)$ is simply $f(x)$.
For $i = 2$, this becomes:

$$\begin{aligned}
\Delta^{(2)}_{a1,a2} f(x) &= \Delta_{a2}(\Delta_{a1} f(x)) \\
&= \Delta_{a2}(f(x+a_1) - f(x)) \\
&= (f(x+a_1+a_2) - f(x+a_2) - (f(x+a_1) - f(x)) \\
&= f(x+a_1+a_2) - f(x+a_1) - f(x+a_2) + f(x).
\end{aligned}$$

It then follows that.

$$f(x+a_1+a_2) = \Delta^{(2)}_{a_1,a_2} f(x) + \Delta_{a_1} f(x) + \Delta_{a_2} f(x) + f(x).$$

Proposition 2 ([Lai94]).

$$f(x+a_1+a_2+\ldots+a_n) = \sum_{i=0}^{n} \sum_{1 \le j_1 \cdots < j_i \le n} \Delta^{(i)}_{a_{j_1},\ldots,a_{j_i}} f(x).$$

In [Lai94], Lai added some basic properties of the derivatives as follows.

$$\Delta_a(f+g) = \Delta_a f + \Delta_a g$$

$$\Delta_a(f(x)g(x)) = f(x+a)\Delta_a g(x) + (\Delta_a f(x)) f(x).$$

Proposition 3 ([Lai94]). *Let $deg(f)$ denote the nonlinearity degree of a multivariable polynomial function $f(x)$. Therefore, it follows that $deg(\Delta_a f(x)) \le deg(f(x)) - 1$.*

Furthermore, Lai discussed the deviates of binary functions and proposed the following propositions.

Proposition 4 ([Lai94]). *Let $L[a_1, a_2, \cdots, a_i]$ be the list of all 2^i possible linear com-*

binations of $a_1, a_2, \cdot a_i$ *then,*

$$\Delta^{(i)}_{a_1,\cdots,a_i} f(x) = \sum_{c \in L[a_1,a_2,\cdots a_i]} f(x \oplus c).$$

Proposition 5 ([Lai94]). *For any function* $f : F_2^n \to F_2^m$*, the* n*-th derivative of* f *is constant. If* $f : F_2^n \mapsto F_2^m$ *is invertible, then the* $(n-1)$*-th derivative of* f *is also constant.*

Proposition 6 ([Lai94]). *The derivatives of a Boolean function are independent of the order of differentiation. For any permutation* $p(j)$ *of the index* j*, the derivatives stay the same.*

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \Delta^{(i)}_{a_{p(1)},\ldots,a_{p(i)}} f(x).$$

The complexity of higher order differential could be computed as $2^{r+1} \times 2^{2n+r}$ where r is the nonlinear order of the function.

2.4.2 Linear Cryptanalysis

Matsui introduced Linear Cryptanalysis [Mat93] as an attack model aimed at evaluating the security of the DES cipher. Similar to differential cryptanalysis, it is a chosen plaintext attack where the attacker can choose a set of plaintexts, either predefined or random, along with their corresponding ciphertexts.

This method leverages the statistical properties of linear approximations between the plaintext and ciphertext. By detecting these linear relationships, attackers can extract vital key information. The core idea involves approximating part of the cipher's operations through bitwise manipulation using mod-2, specifically utilizing the XOR operation, symbolized by $\oplus$. This can be represented in the following form:

$$X_{t_1} \oplus X_{t_2} \cdots \oplus X_{t_u} \oplus Y_{v_1} \oplus Y_{v_2} \cdots \oplus Y_{v_z} = 0. \tag{2.7}$$

Here, X_t denotes the t-th bit of the input vector $X = [X_1, X_2, ...]$, and Y_v represents the v-th bit of the output vector $Y = [Y_1, Y_2, ...]$. Equation 2.7 expresses the exclusive-OR of t input bits and v output bits.

In linear cryptanalysis, the technique identifies expressions similar to those in equation 2.7 that exhibit either a high or low probability. If a cipher shows a pattern where this equation holds or fails with high probability, it indicates a lack of sufficient randomness in the cipher. If we can derive a linear approximation of a cipher with $[Pr = 1]$, it reveals a significant vulnerability. Interestingly, some specific bit positions in the ChaCha stream cipher can be approximated with a probability of 1. This is elaborated on in Lemma 3.1. To aggregate the linear biases across different rounds of

a cipher, we use Lemma 3 from [Mat93], known as the (Piling-up Lemma), which is detailed in Lemma 2.1.

Lemma 2.1 ([Mat93]). *Let X_i $(1 \leq i \leq n)$ represent independent random variables, where each X_i takes the value 0 with probability p_i and 1 with probability $1 - p_i$. The probability that $X_1 \oplus X_2 \cdots \oplus X_n = 0$ can be expressed as*

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right).$$

According to Lemma 2.1, adding more rounds to a cipher increases its security when compared to using fewer rounds.

2.4.3 Differential-linear Cryptanalysis

The differential-linear attack [LH94] resembles standard differential and linear attacks. The key distinction between these methods lies in the specific implementation used to uncover vulnerabilities in the cipher. The fundamental concept of the differential-linear adversary model is to merge the differential bias with linear correlations.

The initial tactic of avoiding long differentials and linear approximations offered a robust defense for the cipher against specific attacks. However, it soon became evident that exploiting shorter characteristics and approximations could still pose a security risk. In 1994, Langford [LH94] introduced the differential-linear cryptanalysis (DL technique), marking a significant advancement. He demonstrated that when a cipher E can be broken down into a sequence $E = E_2 \cdot E_1$, the combination of a differential probability for E_1 and a biased linear approximation for E_2 can effectively distinguish the entire cipher E.

This technique has proven effective against various ciphers. The procedure for implementing the DL attack is as follows: Consider E as a cipher composed of two sub-ciphers, $E1$ and $E2$, where $E1$ consists of m rounds and $E2$ consists of l rounds of the main cipher. We can express this as $E = E_2 \cdot E_1$. To attack cipher E using differential cryptanalysis, we employ differential-linear cryptanalysis on E_1 and linear cryptanalysis on E_2, covering the cipher's m and l rounds, respectively. For E_1, we introduce an input difference $\mathcal{ID}$, $\Delta X^{(0)}$, into the initial states of the sub-cipher E_1 and obtain the output difference $\mathcal{OD}$, $\Delta X^{(m)}$, after m rounds.

Next, linear cryptanalysis is utilized on E_2 with masks Γ_m and Γ_{out} to find linear approximations for the remaining l rounds of the cipher E. This method enables the creation of a differential-linear distinguisher that covers the entire $m + l$ rounds of the cipher E.

Let $\Delta_i^{(r)}[j]$ denote the difference between the bits at the j-th position of the i-th word after r internal rounds, computed as $x_i^{(r)}[j] \oplus x_i'^{(r)}[j]$. Let $\mathcal{L}$ be the set of bits,

and define σ and σ' as linear combinations of bits in $\mathcal{J}$, where $\sigma = \left(\bigoplus_{(i,[j])\in\mathcal{J}} x_{i,[j]}^{(r)}\right)$ and $\sigma' = \left(\bigoplus_{(i,[j])\in\mathcal{J}} x_{i,[j]}^{\prime(r)}\right)$. The linear combination ΔX is represented as $\Delta X = \left(\bigoplus_{(i,[j])\in\mathcal{J}} \Delta x_{i,[j]}^{(r)}\right)$.

$$
\begin{aligned}
&Pr[\Delta_\rho = 0] = Pr[\rho \oplus \rho' = 0] = \frac{1}{2}(+\gamma).\\
&Pr[\Delta_\sigma = \Delta_\rho] = Pr[\sigma = \rho] \cdot Pr[\sigma' = \rho']\\
&\quad + Pr[\sigma = \bar{\rho}] \cdot Pr[\sigma' = \bar{\rho}'] = \frac{1}{2}(1+\varepsilon_L) \cdot \frac{1}{2}(1+\varepsilon_L)\\
&\quad + \frac{1}{2}(1-\varepsilon_L) \cdot \frac{1}{2}(1-\varepsilon_L) = \frac{1}{2}(1+\varepsilon_L^2).
\end{aligned}
$$

Afterwards,

$$
\begin{aligned}
Pr[\Delta_\rho = 0] &= Pr[\Delta_\sigma = 0] \cdot Pr[\Delta_\sigma = \Delta_\rho]\\
&\quad + Pr[\Delta_\sigma = 1] \cdot Pr[\Delta_\sigma = \overline{\Delta_\rho}]\\
&= \frac{1}{2}(1+\varepsilon_d) \cdot \frac{1}{2}(1+\varepsilon_L^2) + \frac{1}{2}(1-\varepsilon_d) \cdot \frac{1}{2}(1-\varepsilon_L^2)\\
&= \frac{1}{2}(1+\varepsilon_d \cdot \varepsilon_L^2). \quad \text{[CM16]}
\end{aligned}
$$

The differential-linear correlation is expressed as $Pr\left[\Delta\rho = 0 | \Delta X^{(0)}\right] = \frac{1}{2}(1+\varepsilon_d \cdot \varepsilon_L^2)$, where $\varepsilon_d \cdot \varepsilon_L^2$ denotes the differential-linear bias. The complexity of the distinguisher is given by $\mathcal{O}\left(\frac{1}{\varepsilon_d^2 \cdot \varepsilon_L^4}\right)$. Generally, to distinguish between two events where one occurs with probability p and the other with a much smaller probability q, a minimum of $\mathcal{O}\left(\frac{1}{pq^2}\right)$ samples are needed. In the Differential-Linear adversary model, the assumption of randomness involves the independence between the sub-ciphers E_1 and E_2.

2.4.4 The Boomerang Attack

Wagner [Wag99] introduced the boomerang attack. It is an innovative method to demonstrate that the absence of long high-probability differentials does not assure the security of a cipher against differential cryptanalysis. The boomerang attack is a strategic method to connect two unrelated high-probability differences in a cipher's upper and lower sections. The motivation for the boomerang attack is obvious, as in many ciphers, it is easier to look for short differentials with a high probability than to search for a long differential with a low probability.

The concept involves encrypting two plaintexts with a single bit difference through the cipher, adding a single bit difference in the resulting ciphertext, and studying their return path. A second-order differential within the cipher bridges the gaps between the upper and lower sections and allows the boomerang attack. The classical analysis

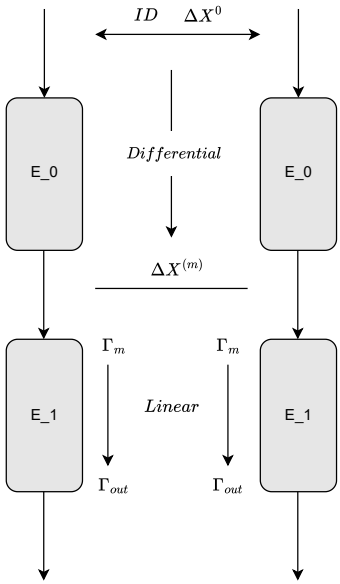

Figure 2.3: Schematic of Differential Linear Cryptanalysis

of the boomerang probability assumes the differentials independently.

Theoritical Analysis of Boomerang Attack

Consider a cipher E broken down as $E = E_1 \cdot E_0$, where E_0 has a differential characteristic $\alpha \xrightarrow{p} \beta$ and E_1 has a differential characteristic $\gamma \xrightarrow{q} \delta$, as shown in Fig. 2.4, where $pq \leq 2^{-n/2}$. By applying Algorithm 1, one can distinguish the cipher E from a random permutation. The algorithm is described as follows:

$$Y_1^{(r)} \oplus Y_2^{(r)} = \beta. \tag{2.8}$$

Equation 2.8 occurs with probability p. Meanwhile, as illustrated in Fig. 2.4, we obtain two new ciphertext states: $Z_1 \oplus \delta = Z_3$ and $Z_2 \oplus \delta = Z_4$. Using the differential characteristics of E_1, we calculate

$$(Y_1^{(r)} \oplus Y_3^{(r)} = \gamma) \wedge (Y_2^{(r)} \oplus Y_3^{(r)} = \gamma). \tag{2.9}$$

Equation 2.9 occurs[2] with a probability of q^2. If both Equations 2.8 and 2.9 hold, then this results in $Y_3 \oplus Y_4 = \beta$. We have

[2]The differential characteristic $\gamma \xrightarrow{q} \delta$ for E_1 is equivalent to $\delta \xrightarrow{q} \gamma$ for E_1^{-1}, as they both consider the same set of input/output pairs for E_1.

Algorithm 1 The Boomerang Cryptanalysis Algorithm [DKRS20]

Input: The initial states (X_1, X_2) with an input difference α.
Output: Identification of the cipher E or a random oracle.

1. Initialize a counter **ctr** $\leftarrow 0$.
2. Generate $(pq)^{-2}$ random plaintext pairs (X_1, X_2, X_3, X_4) with input difference α.
3. For each pair (X_1, X_2, X_3, X_4):
4. Encrypt (X_1, X_2, X_3, X_4) to obtain (Z_1, Z_2, Z_3, Z_4).
5. Calculate $Z_1 \oplus Z_3$ and $Z_2 \oplus Z_4$.
6. If ($Z_1 \oplus Z_3 = \delta$ and $Z_2 \oplus Z_4 = \delta$):
7. Increment **ctr**.
8. End If
9. If **ctr** ≥ 1:
10. **Return:** This is an encryption algorithm.
11. Else, **Return:** This is a random permutation.

$$\begin{aligned} Y_3^{(r)} \oplus Y_4^{(r)} &= (Y_2^{(r)} \oplus Y_4^{(r)}) \oplus (Y_1^{(r)} \oplus Y_3^{(r)}) \oplus (Y_2^{(r)} \oplus Y_1^{(r)}) \\ &= \gamma \oplus \gamma \oplus \beta = \beta. \end{aligned} \tag{2.10}$$

Consequently, using the differential characteristic of E_0, we deduce that $X_3 \oplus X_4 = \alpha$ with a probability of p. Assuming these events are independent, we obtain:

$$Pr[X_3^{(0)} \oplus X_4^{(0)} = \alpha \mid X_1^{(0)} \oplus X_2'^{(0)} = \alpha] = p^2 q^2. \tag{2.11}$$

As per Dunkelman [DKRS20], when considering $1/(pq)^2$ pairs (X_1, X_2), there's a high likelihood (around 63%) that for at least one pair of $X_3^{(0)} \oplus X_4^{(0)} = \alpha$, leading the algorithm to output the cipher E. Conversely, for a random permutation, $\Pr[X_3^{(0)} \oplus X_4^{(0)} = \alpha] = 2^{-n}$, thus expecting the count of pairs $(X_1^{(0)}, X_2^{(0)})$ satisfying $X_3^{(0)} \oplus X_4^{(0)} = \alpha$ to be $2^{-n} \cdot (pq)^{-2} \geq 1$ (given $pq \geq 2^{-n/2}$). Consequently, the algorithm outputs random permutation. Therefore, this approach effectively identifies E from a random permutation.

To differentiate the cipher E from a random oracle, it needs a total of $4(pq)^{-2}$ adaptively chosen plaintexts and ciphertexts denoted as ACPC. In this paper, we apply Wagner's boomerang attack on ChaCha.

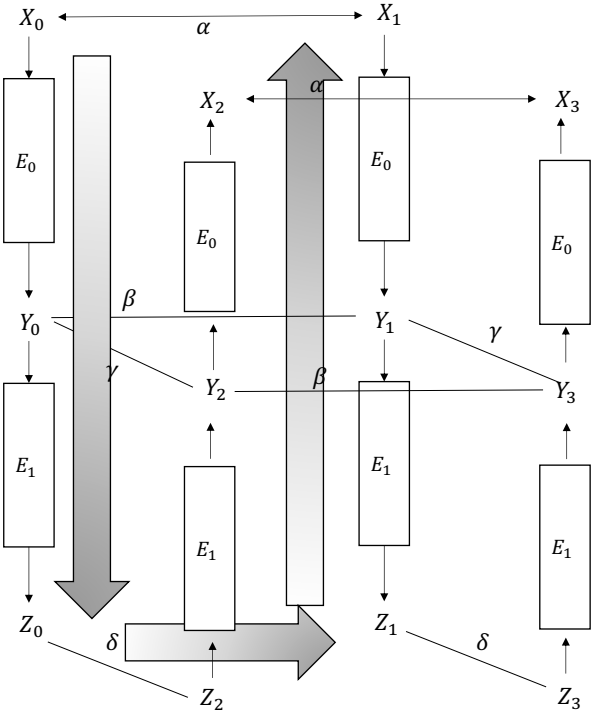

Figure 2.4: The Boomerang Attack

Chapter 3

Previous Works

This chapter introduces the significant studies on Salsa 20 and ChaCha stream ciphers. To understand the existing studies better, we divide the existing attacks into the following categories: differential attacks, differential-linear attacks, higher-order differential attacks, and attacks on ChaCha permutation.

3.1 Existing Attacks on Salsa 20 and ChaCha

In this section, we review the existing studies on Salsa 20 and ChaCha. The significant differential attack against Salsa 20 and ChaCha stream ciphers started with Aumasson's work [AFK+08].

Aumasson introduced a technique to analyze the security of Salsa 20 and ChaCha based on the analysis of probabilistic neutral bits (PNBs) and introduced a framework for probabilistic backward computation (PBC). The PNBs study the effect of each keybit on the output of the cipher. It divides the keybits into two subsets of significant keybits and non-significant keybits. Based on the PNBs framework, Aumasson attacked Salsa 7, Salsa 8, ChaCha 6, and ChaCha 7. Aumasson used the truncated differential to find a pair of input difference $\mathcal{ID}$ and output difference $\mathcal{OD}$ and then looked for differential bias.

Given the pair of $\mathcal{ID}, \mathcal{OD}$ and the probabilistic backward computation framework, the set of PNBs was identified. With the decrease in the complexity of the attack, the selection of $\mathcal{OD}$ position plays a crucial role. The Aumasson's cryptanalysis method encompasses two main stages: Preliminary Computation Stage and real-time execution phases. The overall structure of the attack is represented in the figure 3.1.

3.1.1 Preliminary Computation Stage

Initially, we create two-state matrices X and X'. Both matrices contain the same keywords $(k_1, k_2, \ldots, k_8)$ and constants. However, the matrix X' differs from X by

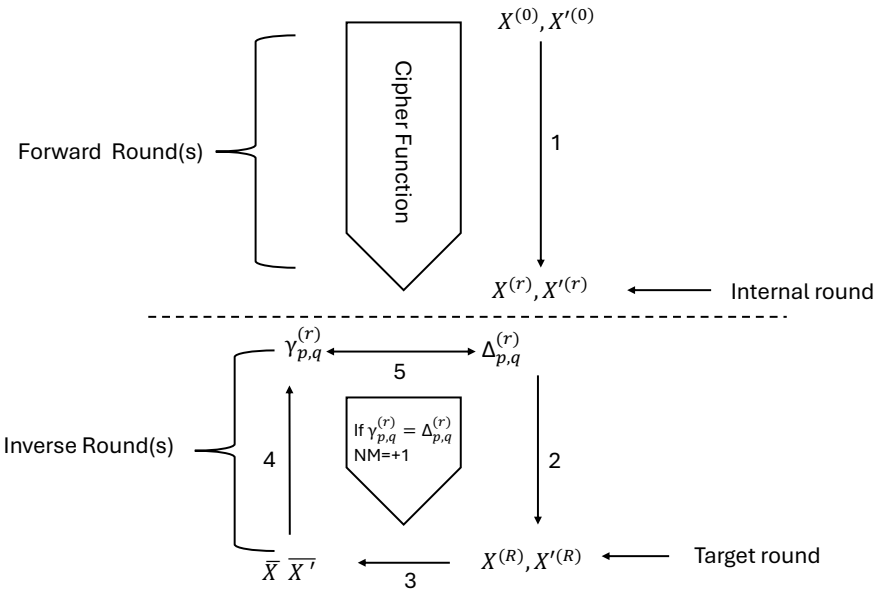

Figure 3.1: Graphic Representation of PNB based cryptanalysis.

a single bit in the nonce v or the counter t. To elaborate further, let $x_i^{(0)}[j]$ denote the j-th bit of the i-th word in the initial state matrix $X^{(0)}$, where $0 \leq i \leq 15$ and $0 \leq j \leq 31$.

Likewise, let $x_i'^{(0)}[j]$ denote the corresponding word with a solitary bit distinction at the j-th position, indicated as $\Delta_i^{(0)}[j] = x_i^{(0)}[j] \oplus x_i'^{(0)}[j]$. If there exists a dissimilarity $\Delta_i^{(0)}[j] = 1$ at the j-th bit of the i-th word in the initial state matrix $X^{(0)}$, identified as the input difference or ID, we obtain the respective initial state matrix X' by adjusting either the nonce v as $v' = v \oplus \Delta v$ or the counter t as $t' = t \oplus \Delta t$, where v' and t' denote the single-bit difference in nonce or counter.

Afterward, we apply the Salsa 20 round function to the initial state matrices $X^{(0)}$ and $X'^{(0)}$, resulting in a single-bit difference output $\Delta_p^{(r)}[q] = x_p^{(r)}[q] \oplus x_p'^{(r)}[q]$ from the r-round internal state matrices $X^{(r)}$ and $X'^{(r)}$.

This difference is termed the output difference (OD), where $1 \leq r < R$, and q represents the q-th bit of the p-th word in the internal state matrix $X^{(r)}$ for $0 \leq p \leq 15$ and $0 \leq q \leq 31$ after r rounds of Salsa 20. Given a fixed key, with random nonces and block counters, the bias ε_d is defined as:

$$\Pr\left(\Delta_p^{(r)}[q] = 1 \mid \Delta_i^{(0)}[j] = 1\right) = \frac{1}{2}(1 + \varepsilon_d). \tag{3.1}$$

When the key is random, ε_d^* is determined as the median of ε_d [AFK+08].

Theorem 3.1 ([MS01, Theorem 2]). *If we have two distributions, denoted as $\mathcal{X}$ and $\mathcal{Y}$, where the event of interest happens with a probability p in $\mathcal{X}$ and with a probability $p \cdot (1 + q)$ in $\mathcal{Y}$. In cases where p and q are small, you can use approximately $\mathcal{O}(\frac{1}{p \cdot q^2})$ samples to effectively differentiate between $\mathcal{X}$ and $\mathcal{Y}$ with a consistent probability of*

success.

Suppose we have two distributions, denoted as $\mathcal{X}$ and $\mathcal{Y}$. In $\mathcal{X}$, the event E occurs with a probability of $\frac{1}{2}$, representing the outcome of a true random number generator. In $\mathcal{Y}$, the event E' occurs with a probability of $\frac{1}{2} \cdot (1+\varepsilon_d)$, representing the probability of the difference in output ($\mathcal{OD}$) derived from the round internal matrices r of the Salsa 20 stream cipher. As stated in Theorem 3.1, the required number of samples to discern between $\mathcal{X}$ and $\mathcal{Y}$ is $\mathcal{O}(\frac{2}{\varepsilon_d^2})$.

3.1.2 Probabilistic Neutral Bits

The idea of probabilistic neutral bits (PNB) enables us to divide the key bits into two separate categories.

We will represent them as m, the set of significant key bits, and n, the nonsignificant key bits, with $m = 256 - n$. To distinguish between these sets, the PNB concept analyzes the impact of each key bit on the output of the Salsa 20 function, denoted here as $\mathcal{OD}$. These impacts are called neutral measures of key bits or γ_k.

Definition 3.1 ([AFK+08, Definition 1])**.** *The neutral evaluation of the key bit position γ_i concerning the $\mathcal{OD}$ is denoted as γ_κ, where $\frac{1}{2}(1+\gamma_\kappa)$ signifies the likelihood that altering the key bit κ at position γ_i doesn't impact the $\mathcal{OD}$.*

According to [AFK+08], the following singular cases of the neutral measure exist:

- When $\gamma_k = 1$, the $\mathcal{OD}$ is unaffected by the i-th key bit, indicating its insignificance.
- When $\gamma_k = 0$, the $\mathcal{OD}$ is statistically independent of the i-th key bit, signifying its significance.
- When $\gamma_k = -1$, $\mathcal{OD}$ shows a linear dependence on the i th key bit.

3.1.3 Probabilistic Inverse Analysis

We determine the forward differential bias and subsequently calculate the internal round differential bias in reverse through probabilistic backward computation. By reversing the Salsa 20 keystream equations, $Z = X + \text{Round}\, R(X)$ and $Z' = X' + \text{Round}\, R(X')$, using the matrices $Z - X$ and $Z' - X'$ as inputs, the single bit differential bias for the r rounds can be obtained from a backward perspective. The probabilistic backward computation (PBC) computation is based on these matrices for the reverse round of Salsa 20. The bias ε is approximated as $\varepsilon \approx \varepsilon_e \cdot \varepsilon_a$ under the independence assumption.

3.1.4 Attack Phase

As outlined in [AFK+08], during the online phase, the Algorithm in section 3.4 of [AFK+08] requires specific parameters as input, including the position of $\mathcal{OD}$, the position of $\mathcal{ID}$, the subset of significant key bits m, and N keystream block elements to recover the secret key potentially.

Algorithm 2 [AFK+08] PNB bits verification

Input: Random(X, X',Z,Z')

Output: The absolute value of $\tilde{\varepsilon}$

1. Calculate $(X^{(R)}, X'^{(R)})$ where $\Delta_i^{(0)}[j] = 1$; and determine $Z = X^{(0)} + X^{(R)}$ and $Z' = X'^{(0)} + X'^{(R)}$.

3. Prepare $(\tilde{X}^{(0)}, \tilde{X'}^{(0)})$ with all key bits initialized to a random binary value from $(X^{(0)}, X'^{(0)})$.

4. Evaluate $(\tilde{Y}^{(r)}, \tilde{Y}'^{(r)})$ using $Z - \tilde{X}^{(0)}$ and $Z' - \tilde{X'}^{(0)}$ as inputs to the inverted round function of Salsa 20.

5. Determine $\tilde{\Gamma}_p^{(r)}[q] = \tilde{y}_p^{(r)}[q] \oplus \tilde{y}_p'^{(r)}[q]$

6. Calculate $\tilde{\varepsilon}$ as $\Pr(\tilde{\Gamma}_p^{(r)}[q] \mid \Delta_i^{(0)}[j] = 1) = \frac{1}{2}(1 + \tilde{\varepsilon})$.

3.1.5 Attack Complexity

After identifying an optimal pair $\left(\Delta_p^{(r)}[q] = 1 \mid \Delta_i^{(0)}[j] = 1\right)$ for both $\mathcal{ID}$ and $\mathcal{OD}$, and segregating the key bits into significant (m) and non-significant (n) categories, and determining the median bias as $|\varepsilon^*| \approx |\varepsilon_e^*| \cdot |\varepsilon_a^*|$, we can then calculate the final attack complexity using the following equation.

$$2^m(N + 2^n P_{fa}) = 2^m N + 2^{256-\alpha}, \text{ where } N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-\varepsilon^2}}{\varepsilon}\right)^2. \tag{3.2}$$

where alpha α is used as a parameter that helps determine the probability of a false alarm (pfa) in hypothesis testing during the attack.

3.2 Differential Attack on Salsa 20 and ChaCha

Differential cryptanalysis is the main and most important attack methodology on Salsa 20 and ChaCha. Aumasson [AFK+08] introduced the most important attack on Salsa 20 and ChaCha.

Table 3.1: The Aumasson [AFK+08] Attack Summary on Salsa

Rounds	$\mathcal{ID}, \mathcal{OD}$	Bias	PNBs	Time/Data
Salsa 7	$\Delta X_1^{(0)}[14], \Delta X_7^{(4)}[31]$	$\lvert\varepsilon_d^\star\rvert = 0.131$	131	2^{151} / 2^{26}
Salsa 8	$\Delta X_1^{(0)}[14], \Delta X_7^{(4)}[31]$	$\lvert\varepsilon_d^\star\rvert = 0.131$	36	2^{251} / 2^{31}

Table 3.2: The Aumasson [AFK+08] Attack Summary on ChaCha

Rounds	$\mathcal{ID}, \mathcal{OD}$	Bias	PNBs	Time/Data
ChaCha 6	$\Delta X_{13}^{(0)}[13], \Delta X_{11}^{(3)}[0]$	$\lvert\varepsilon_d^\star\rvert = 0.026$	147	2^{139} / 2^{30}
ChaCha 7	$\Delta X_{13}^{(0)}[13], \Delta X_{11}^{(3)}[0]$	$\lvert\varepsilon_d^\star\rvert = 0.026$	35	2^{248} / 2^{27}

For Salsa 7 and Salsa 8, the attack utilized the $\mathcal{ID}, \mathcal{OD}$ $\Delta X_1^{(0)}[14], \Delta X_7^{(4)}[31]$. The bias for Salsa 7 had a median value of $|\varepsilon_d^\star| = 0.131$, consistent with the bias for Salsa 8.

The probabilistic neutral bits (PNBs) threshold varied, with an optimal $\gamma = 0.4$ for Salsa 20 and $\gamma = 0.12$ for Salsa 8. The number of PNBs differed based on the threshold value, with Salsa 7 having $n = 131$ for $\gamma = 0.50$, and Salsa 8 exhibiting $n = 36$. The key bits declared as PNBs for Salsa 8 include {26, 27, 28, 29, 30, 31, 71, 72, 120, 121, 122, 148, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 210, 211, 212, 224, 225, 242, 243, 244, 245, 246, 247}.

The attack complexities were notably high, with Salsa 7 requiring a time complexity of 2^{151} and a data complexity of 2^{26}, while Salsa 8 required a time complexity of 2^{251} and a data complexity of 2^{31}. For ChaCha 6 and ChaCha 7, the differential $\Delta X_{13}^{(0)}[13], \Delta X_{11}^{(3)}[0]$ was employed, with both versions exhibiting a bias of $|\varepsilon_d^\star| = 0.026$. The threshold for PNBs was set at $\gamma = 0.6$ for ChaCha 6 and $\gamma = 0.5$ for ChaCha 7.

The number of PNBs found was $n = 147$ for ChaCha 6 and $n = 35$ for ChaCha 7. The key bits identified as PNBs for ChaCha 7 are {3, 6, 15, 16, 31, 35, 67, 68, 71, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 103, 104, 127, 136, 191, 223, 224, 225, 248, 249, 250, 251, 252, 253, 254, 255}. The attack complexities were calculated to be 2^{139} in time and 2^{30} in data for ChaCha6, whereas ChaCha 7 exhibited a time complexity of 2^{248} and a data complexity of 2^{27}.

Following the [AFK+08], Shi [SZFW12] introduced the idea of chaining distinguishes. Column Chaining Distinguishers (CCD) is constructed using a series of distinguishers, each relying on an increasing subset of key bits. The key idea is to chain these distinguishers together so that the output from one step feeds into the next, progressively narrowing down the key space. This process involves using differentials with increasing complexity to filter out incorrect key guesses gradually. Let $S(K_i')$ denote the keybits in the subkey K_i'. For a collection of subkeys $\{K_i'\}_{i\in A}$ with $S(K_i') \subseteq S(K_j')$ for all $i, j \in A$ and $i < j$, if there exists a collection of distinguishers $\{D_i\}_{i\in A}$, where each distinguisher D_i effectively depends on the subkey K_i', then $\{D_i\}_{i\in A}$ is called the

Column Chaining Distinguishers (CCD) for $\{K'_i\}_{i\in A}$. Row Chaining Distinguishers (RCD) are similar to CCD but focus on chaining distinguishers that rely on the same key bits while applying differentials.

The advantage of RCD is that it reduces the chance of false positives in key guesses by requiring a key candidate to pass multiple independent tests. RCD can also be used in conjunction with CCD for more complex attacks. For a fixed subkey K', if there exists a collection of distinguishers $\{D_i\}_{i\in A}$ that effectively depend on the subkey K', then $\{D_i\}_{i\in A}$ is called the Row Chaining Distinguishers (RCD) for K'. Shi mounted an attack on Salsa 7, for the differential $\Delta X_1^{(0)}[14], \Delta X_7^{(4)}[31]$ with $|\varepsilon_d| = 0.131$ construct a 2-step CCD using $\gamma^{(1)} = 0.5$ and $\gamma^{(2)} = 0.6$, with corresponding biases $\varepsilon^{(1)} = 0.0022$ and $\varepsilon^{(2)} = 0.0050$. For the first threshold $\gamma^{(1)} = 0.5$, the author found $ns_1 = 125$ non-significant key bits. For the second threshold $\gamma^{(2)} = 0.6$, the author found $ns_2 = 120$ non-significant key bits.

The total time complexity is approximately 2^{148}, the data complexity is 2^{24}, and the success probability of the attack is $50\% \times 90\% \approx 45\%$. Shi attacked on 256-bit Salsa 8. For the differential $\Delta X_1^{(0)}[14], \Delta X_7^{(4)}[31]$ with $|\varepsilon_d| = 0.131$, Shi construct a 2-step CCD. Using $\gamma^{(1)} = 0.15$ and $\gamma^{(2)} = 0.20$ with $\varepsilon^{(1)} = 0.00047$ and $\varepsilon^{(2)} = 0.00102$ respectively. For the threshold $\gamma^{(1)} = 0.15$, Shi found $ns_1 = 33$ non-significant key bits, and for the threshold $\gamma^{(2)} = 0.20$, the author reported $ns_2 = 30$ non-significant key bits. The value $\varepsilon^{(2)} = 0.00102$ is chosen with a step success probability of 90%. The time complexity is $2^{223} \cdot N_1 + 2^{226-\alpha_1} \cdot N_2 + 2^{256-\alpha_1-\alpha_2}$. Shi selected $\alpha_1 = 2$ and $\alpha_2 = 7$, then get $N_1 = 2^{26.5}$ and $N_2 = 2^{25}$ respectively. So the time complexity is 2^{250}, the data complexity is 2^{27}, and the success probability is $50\% \times 90\% = 45\%$. Shi introduced an attack on Chacha 7.

For the differential $\Delta X_{13}^{(0)}[13], \Delta X_{11}^{(3)}[0]$, Shi construct a 4-step CCD using $\gamma^{(1)} = 0.50$, $\gamma^{(2)} = 0.53$, $\gamma^{(3)} = 0.55$, and $\gamma^{(4)} = 0.58$ with corresponding biases $\varepsilon^{(1)} = 0.00059$, $\varepsilon^{(2)} = 0.00080$, $\varepsilon^{(3)} = 0.00127$, and $\varepsilon^{(4)} = 0.00280$. For the first threshold $\gamma^{(1)} = 0.50$, Shi find $ns_1 = 35$ non-significant key bits. As the thresholds increase, the number of non-significant key bits decreases, with $ns_2 = 34$, $ns_3 = 32$, and $ns_4 = 28$ for the subsequent thresholds. The time complexity of this attack is given by $2^{221} \cdot N_1 + 2^{222-\alpha_1} \cdot N_2 + 2^{224-\alpha_1-\alpha_2} \cdot N_3 + 2^{228-\alpha_1-\alpha_2-\alpha_3} \cdot N_4 + 2^{256-\alpha_1-\alpha_2-\alpha_3-\alpha_4}$. Shi selected $\alpha_1 = 3.8$, $\alpha_2 = 3.5$, $\alpha_3 = 5$, and $\alpha_4 = 9$, and then calculated $N_1 = 2^{26.3}$, $N_2 = 2^{25.3}$, $N_3 = 2^{24.2}$, and $N_4 = 2^{22.4}$. Thus, the total time complexity is approximately $2^{246.5}$, with a data complexity of 2^{27}.

Maitra [Mai16] introduced the chosen IV attack on Salsa 20 and Chacha. In the cryptanalysis of Salsa, Maitra focuses on Salsa 8. The attack employs a chosen IV strategy, where an $\mathcal{ID}, \mathcal{OD}$ is introduced at $\Delta X_1^{(0)}[14], \Delta X_7^{(4)}[31]$. The bias observed in the forward direction (ε_d) for this $\mathcal{ID}, \mathcal{OD}$ pair is significantly higher when specific IVs and keys are chosen. For example, setting $x_3 = k_2 = 0$, $x_{11} = k_4 = 0$, and

$x_{15} = 0xaaaaaaaa$ (where k_2 and k_4 are specific key words and x_{15} is an IV) yields an observed bias ε_d of approximately 0.2245.

This improvement in bias results in a reduction of the overall attack complexity. In this attack, 33 PNBs are identified. The median bias ε^* for these selected PNBs is calculated to be approximately 0.00315. Considering these factors, the attack complexity against Salsa 8 is estimated as follows. With $\alpha = 15$ and $\varepsilon^* = 0.003154$, the overall time complexity of the attack is estimated to be around $2^{245.52}$.

Maitra [Mai16] introduced an attack on ChaCha 7, the attack is based on a chosen IV strategy. The differential is selected as $\Delta X_{13}^{(0)}[13], \Delta X_{11}^{(3)}[0]$. For ChaCha 7, the following 35 PNBs are identified: {3, 6, 15, 16, 31, 35, 67, 68, 71, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 103, 104, 127, 136, 191, 223, 224, 225, 248, 249, 250, 251, 252, 253, 255}.

When more PNBs are considered, the list expands by an additional 10 PNBs: 7, 17, 36, 38, 72, 105, 137, 156, 159, 194. This increases the total number of PNBs to 45. The median bias ε^* for the $\mathcal{ID}, \mathcal{OD}$ pair $\Delta X_{13}^{(0)}[13], \Delta X_{11}^{(3)}[0]$ is 0.00059. It resulted in an attack with a complexity $N \approx 2^{27}$ and a total time complexity of approximately 2^{248}. When exploiting more PNBs (45 total), the results improve: Median bias ε^* decreases slightly to 0.000132, data complexity $N \approx 2^{31.77}$, and total time complexity reduces to $2^{242.82}$. The attack can be further optimized by choosing specific IVs corresponding to the key bits, particularly focusing on minimizing the number of differences after one quarter-round in ChaCha.

In this scenario, the following improvements are observed: With key settings $x_5 = k_1 = 0$, $x_9 = k_5 = 0$, and $x_{13} = 0xaaaaaaaa$, the forward bias ε_d increases to approximately 0.140344. This increases the overall bias ε^* to 0.002012, data complexity $N \approx 2^{24.05}$, and total time complexity further reduces to $2^{239.14}$. Miyashita [MIM22] attacked ChaCha 7 and ChaCha 7.25 based on analysis of PNBs.

The time complexity and data complexity of the attack were calculated to be $2^{231.63}$ and $2^{49.58}$, respectively. The Probabilistic Neutral Bits were identified during the precomputation phase of the attack as: {6, 7, 8, 9, 10, 11, 12, 13, 14, 19, 27, 28, 29, 30, 31, 34, 35, 36, 37, 46, 71, 79, 80, 83, 98, 99, 100, 101, 102, 103, 104, 105, 106, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 122, 123, 127, 128, 129, 130, 148, 149, 150, 159, 187, 188, 189, 190, 191, 200, 223, 224, 225, 231, 232, 239, 240, 243, 244, 251, 252, 253, 254, 255}. The attack was optimized using the following parameters: the $\mathcal{ID}, \mathcal{OD}$ pair $\Delta X_{14}^{(0)}[16], \Delta X_{3}^{(3.5)}[0]$, with a threshold γ of 0.35, the number of non-significant key bits (n) of 74, a bias $|\varepsilon_d|$ of 0.000478, a bias $|\varepsilon_a|$ of 0.000674.

Additionally, Miyashita extended the differential cryptanalysis attack to ChaCha 7.25. The time complexity of this attack was estimated to be $2^{255.62}$, reflecting the increased difficulty in attacking ChaCha 7.25 compared to ChaCha 7, with a data

complexity of $2^{48.36}$. The PNBs identified for the 7.25-round attack are slightly different, reflecting the change in round number.

The identified PNBs are: {2, 3, 10, 13, 14, 19, 20, 26, 27, 31, 40, 44, 45, 46, 51, 59, 60, 61, 62, 63, 128, 129, 130, 135, 136, 143, 144, 147, 148, 155, 156, 157, 158, 159, 160, 161, 162, 180, 181, 182, 191, 219, 220, 221, 222, 223, 224, 232, 255}. The best parameters for this attack were as follows: the $\mathcal{ID}, \mathcal{OD}$ pair $\Delta X_{15}^{(0)}[6], \Delta X_{0}^{(3.5)}[0]$, with a threshold γ of 0.30, the number of non-significant key bits (n) of 49, a bias $|\varepsilon_d|$ of 0.000469, a bias $|\varepsilon_a|$ of 0.000564.

These findings demonstrate the first successful differential attack on the 7.25-round variant of ChaCha. Although the attack remains less efficient than brute force due to its higher time complexity, it establishes a baseline for future cryptanalytic efforts on the ChaCha stream cipher and contributes valuable insights into the behavior of PNBs across different rounds. Wang [WLHL23] introduced a syncopation technique for enhancing the correlation in PNB-based approximations in the backward direction. This technique leverages the properties of the ARX structure to significantly boost correlation. With this approach, the author proposed a new efficient method to select an optimal set of PNBs. The attack starts by examining the inverse structure of ChaCha 7. For the inversion of the last quarter-round function, the following relationships hold:

$$
\begin{aligned}
x_7^{(7)} &= z_7^{(7)} \ominus k_3, \\
x_{11}^{(7)} &= z_{11}^{(7)} \ominus k_7, \\
y_{11}^{(6)} &= \left(z_{11}^{(7)} \ominus x_{15}^{(7)} \ldots\right) \ominus k_7, \\
x_{11}^{(6)} &= \left(z_{11}^{(7)} \ominus x_{15}^{(7)} \ldots \ominus y_{15}^{(6)} \ldots\right) \ominus k_7,
\end{aligned}
$$

$$
\begin{aligned}
x_3^{(7)} &= z_3^{(7)} \ominus c_3, \\
x_{15}^{(7)} &= z_{15}^{(7)} \ominus v_2, \\
y_{15}^{(6)} &= \left(x_{15}^{(7)} \gg 8\right) \oplus x_3^{(7)},
\end{aligned}
$$

The syncopation technique can be applied to the variables $x_7^{(7)}$, $x_{11}^{(7)}$, $y_{11}^{(6)}$, and $x_{11}^{(6)}$ by using the properties of modular subtraction. The underlined variables are already known since they can be derived from constants, IV, and keystream blocks. The same analysis applies to the other three quarterround functions. The set of CPNBs used in this attack was identified by initially narrowing down 151 potential CPNBs using a conditional neutrality measure threshold of 2^{-5}.

Two sets, containing 74 and 89 CPNBs respectively, were chosen by optimizing for minimal time complexity while ensuring that the data complexity remained within the limits of the available IVs. The 74 CPNB set includes the following bits: {2, 3, 4, 5, 47, 48, 49, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 90, 91, 102, 103, 104, 105, 106, 107, 108, 109, 110, 123, 124, 125, 126, 127, 155,

156, 157, 158, 159, 168, 169, 191, 192, 193, 194, 199, 200, 207, 208, 211, 212, 219, 220, 221, 222, 223, 224, 225, 226, 244, 245, 246, 247, 255}. The 89 CPNB set is composed of the following bits: {2, 3, 4, 5, 6, 14, 15, 26, 47, 48, 49, 51, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 90, 91, 95, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 135, 136, 147, 155, 156, 157, 158, 159, 168, 169, 170, 191, 192, 193, 194, 195, 199, 200, 204, 207, 208, 211, 212, 219, 220, 221, 222, 223, 224, 225, 226}.

Next, the author analyzes the complexity of the proposed attack. Through experiments with 2^{10} random keys and $2^{20.3}$ IVs per key, the conditional correlation ε'_a was theoretically estimated to be $2^{-4.14}$. Consequently, the conditional backward correlation is $\varepsilon_a = 2^{-8.28}$. With $n = 89$, $\theta_1 = 27$, and $\alpha = 54$, the resulting attack has a time complexity of $2^{210.3}$ and a data complexity of $2^{103.3}$. The attack on ChaCha 7.5 employs a syncopation technique within a PNB-based differential cryptanalysis framework, targeting the ARX structure to enhance backward correlation.

This approach is adapted from the ChaCha 7 attack but modified to address the additional complexity introduced by the extra half-round in ChaCha 7.5. The attack achieves a time complexity of $2^{244.9}$ and a data complexity of $2^{104.9}$, with another variant reaching a time complexity of $2^{242.9}$ and data complexity of $2^{125.8}$. The set of CPNBs used in the attack on ChaCha 7.5 includes 54 bits: {74, 75, 83, 90, 91, 92, 95, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 155, 156, 157, 158, 159, 168, 169, 170, 191, 192, 193, 194, 195, 199, 200, 204, 207, 208, 211, 212, 213, 216, 219, 220, 221, 222, 223, 224, 225, 226, 227, 232, 244, 245, 246, 247, 255}. These bits were selected based on a conditional neutrality measure threshold and were further refined using Algorithm 2 to optimize the attack's efficiency.

3.2.1 Differential-Linear Attacks on Salsa 20 and ChaCha

In 2016, Choudhuri and Maitra [CM16] revolutionized the cryptanalysis of Salsa 20 and ChaCha by introducing innovative differential-linear techniques [LH94]. Their groundbreaking method provided new linear approximations for these ciphers, leading to an attack on ChaCha 7 with an impressive time complexity of $2^{237.65}$ and a data complexity of $2^{31.6}$.

Arka [CM16] for the 7-round Salsa cipher, an input difference of $\Delta x_7^{(0)}[0]$ is used, and the output difference $\Delta x_9^{(5)}[0] \oplus \Delta x_{13}^{(5)}[0] \oplus \Delta x_1^{(5)}[13]$ is observed, resulting in a bias of -0.1142. The linear approximation applied is $\Delta x_9^{(5)}[0] \oplus \Delta x_{13}^{(5)}[0] \oplus \Delta x_1^{(5)}[13]$, leading to a data complexity of $2^{31.5}$ and a time complexity of $2^{138.5}$. The 8-round attack on Salsa starts with an input difference of $\Delta x_7^{(0)}[0]$ and examines the output difference $\Delta x_9^{(5)}[0] \oplus \Delta x_{13}^{(5)}[0] \oplus \Delta x_1^{(5)}[13]$. In this scenario, 40 Probabilistic Neutral Bits (PNBs) are identified when $\gamma = 0.1$, and through further experimentation, two additional PNBs are added, resulting in a total of 42 PNBs.

The observed biases are $\varepsilon_a = 0.000752$, $\varepsilon_d = -0.233198$, and $\varepsilon = -0.000178$. The chosen IVs result in a doubling of the ε_d bias, improving the complexity metrics. The attack achieves a data complexity of $N = 2^{30.78}$, a time complexity of $2^{244.85}$ for $\alpha = 15.5$, and a worst-case data complexity of 2^{96}, comparable to the results obtained in previous research. For the 6-round ChaCha attack, when running 4 rounds forward and 2 rounds backward, 159 Probabilistic Neutral Bits (PNBs) are identified, with biases of $\varepsilon_a = 0.000534$ and $\varepsilon_d = 0.212786$. This results in a data complexity of $2^{34.39}$ and a time complexity of $2^{131.40}$.

In the case of running 4.5 rounds forward and 1.5 rounds backward, 161 PNBs are used, with biases of $\varepsilon_a = 0.003958$ and $\varepsilon_d = 0.026652$, leading to a data complexity of $2^{34.51}$ and a time complexity of $2^{129.53}$. The best result is achieved when running 5 rounds forward and 1 round backward, with 166 PNBs and biases of $\varepsilon_a = 0.0028$ and $\varepsilon_d = 0.0068$, resulting in a data complexity of $2^{37.5}$ and a time complexity of $2^{127.5}$.

In the case of the 7-round ChaCha, the attack builds upon the 6-round linear approximation, extending it over the additional round. This extension results in a data complexity of $2^{231.6}$ and a time complexity of $2^{237.65}$, with a 0.136828 and a linear approximation bias of 0.001162. The strategy optimizes the exploitation of the differential and linear properties of the cipher to distinguish its output from random permutation effectively. Building on this foundation, Coutinho [CN20] significantly advanced the field in 2020, executing attacks on ChaCha 6 and ChaCha 7 with operational complexities of $2^{102.2}$ and $2^{231.9}$, respectively.

In 2021, Coutinho [CSN21] advanced his research by developing a novel linear approximation, which enabled a significant attack on ChaCha 6 with a complexity of 2^{51} and achieved key recovery on ChaCha 7 with a complexity of $2^{228.51}$. In addition, it proposed important lemmas to approximate the linear approximation of the ChaCha stream cipher which are summarized as follows: The following holds for $i > 0$:

$$x_{a,i}^{(m-1)} = L_{a,i}^{(m)} \oplus x_{b,i+6}^{(m)} \oplus x_{b,i+18}^{(m)} \oplus x_{c,i+11}^{(m)} \oplus x_{d,i-1}^{(m)}, \text{ w.p. } \frac{1}{2}\left(1 + \frac{1}{2^3}\right).$$

For two active input bits in round $m-1$ and multiple active output bits in round m, the following holds for $i > 0$:

$$x_{\lambda,i}^{(m-1)} \oplus x_{\lambda,i-1}^{(m-1)} = L_{\lambda,i}^{(m)} \oplus L_{\lambda,i-1}^{(m)}, \text{ w.p. } \frac{1}{2}\left(1 + \frac{1}{2^\sigma}\right).$$

where $(\lambda, \sigma) \in \{(a,3),(b,1),(c,2),(d,1)$. Suppose that $(\lambda,\sigma) \in \{(i,i-2),(i-1,i-1)\}$, $i > 1$. Then for three active input bits in round $m-1$ and multiple active output bits in round m, the following holds:

$$x_{b,\lambda}^{(m-1)} \oplus x_{c,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = L_{b,i-1}^{(m)} \oplus L_{c,i}^{(m)} \oplus L_{c,i-1}^{(m)} \oplus x_{d,\sigma}^{(m)}, \text{ w.p. } \frac{1}{2}\left(1 + \frac{1}{2^2}\right).$$

Christof [BBC+22] proposed a framework for differential linear adversaries in ARX ciphers, leading to an attack on ChaCha 7 with a time complexity of $2^{230.86}$ and a data complexity of $2^{48.83}$. Christof improved Differential-Linear Attacks with Applications to ARX Ciphers, the authors present several significant advancements in the differential-linear cryptanalysis of the ChaCha cipher. The differential part of the attack involves collecting many right pairs, $(x, x \oplus \Delta_{\text{in}})$, where $E_1(x) \oplus E_1(x \oplus \Delta_{\text{in}}) = \Delta_m$. The set $X = \{x \in F_n^2 \mid E_1(x) \oplus E_1(x \oplus \Delta_{\text{in}}) = \Delta_m\}$ defines these right pairs. The linear part of the attack proceeds by defining several tuples $(T_{p_i}, \Phi_{\text{out}}^{(p_i)}, \gamma^{(p_i)})$, which are used to achieve high correlation.

The attack efficiently extracts linear relations in the key by partitioning the ciphertext space and using multiple linear masks. Several lemmas, including Lemma 1 of [Leu16] support the partitioning technique:

Lemma 1. [Leu16] Let $a, b \in \mathbb{F}_2^m$ and $z = a + b$. For $i \geq 2$, we have

$$z[i] = \begin{cases} a[i] \oplus b[i] \oplus a[i-1] & \text{if } a[i-1] = b[i-1], \\ a[i] \oplus b[i] \oplus a[i-2] & \text{if } a[i-1] \neq b[i-1] \text{ and } a[i-2] = b[i-2]. \end{cases}$$

In this scenario, $c[i-3]$ denotes the carry output from the majority function, and there is a linear approximation $c[i-3] \approx y_0[i-3]$ with a correlation of 2^{-1}. Lemma 2 from [BBC+22] outlines the correlation of the differential-linear distinguisher across various partitions as follows: Let $s = y_0 \oplus \overline{y_1}$ with $i \geq 3$. Define $S_{b_0 b_1}$ as the set $S_{b_0 b_1} := \{(y_1, y_0) \in \mathbb{F}_{2^m}^2 \mid s[i-1] = b_0 \text{ and } s[i-2] = b_1\}\}$. Then, we have:

$$z_0[i] \approx \begin{cases} y_0[i] \oplus y_1[i] \oplus y_0[i-1], & \text{with cor. } -1, \text{ if } (y_1, y_0) \in S_0^*, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-2], & \text{with cor. } -1, \text{ if } (y_1, y_0) \in S_{10}, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-3], & \text{with cor. } -2^{-1}, \text{ if } (y_1, y_0) \in S_{11}, \end{cases}$$

where $S_0^* = S_{00} \cup S_{01}$.

Lemma 3 of [BBC+22]: Let $s = y_0 \oplus \overline{y_1}$ and let $i \geq 3$. Let $S_{b_0 b_1} := \{(y_1, y_0) \in \mathbb{F}_{2m}^2 \mid s[i-1] = b_0 \text{ and } s[i-2] = b_1\}$. We have

$$z_0[i] \oplus z_0[i-1] \approx \begin{cases} y_0[i] \oplus y_1[i], & \text{with cor. } 1, \text{ if } (y_1, y_0) \in S_1^*, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-1] \oplus y_0[i-2], & \text{with cor. } -1, \text{ if } (y_1, y_0) \in S_{00}, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-1] \oplus y_0[i-3], & \text{with cor. } -2^{-1}, \text{ if } (y_1, y_0) \in S_{01}, \end{cases}$$

where $S_1^* = S_{10} \cup S_{11}$. The final attack complexity for the 6-round ChaCha attack is 2^{58} in data complexity and $2^{77.4}$ in time complexity, while for the 7-round ChaCha attack, it is $2^{48.83}$ in data complexity and $2^{230.86}$ in time complexity. In 2022, Dey

[DGSS22] propelled the field of differential-linear cryptanalysis forward by launching an offensive on ChaCha 7 with a time complexity of 2^{221} and a data complexity of 2^{90}. Additionally, Dey [DDSM22] revisited ChaCha cryptanalysis from Crypto 2020 and Eurocrypt 2021.

Furthermore, Dey [DGSS23] unveiled an attack on ChaCha 7.25, yielding a complexity of $2^{244.85}$. Subsequently, Niu et al. [NSLL22] presented an enhanced differential-linear distinguisher tailored for four rounds of ChaCha. In 2023, Coutinho et al. [CPV+23b] proposed a method to select the $\mathcal{ID}, \mathcal{OD}$. Traditionally, a PNB attack considers a single $\mathcal{ID}, \mathcal{OD}$ pair. The attacker first identifies significant key bits (non-PNBs) based on the bias of the output difference, then exhaustively searches the remaining PNBs, making this a two-step process.

Dey [DGSS22] proposed a technique that combines two input-output positions in a PNB attack, using the same $\mathcal{ID}$ with two different output differences ($\mathcal{OD}_1$ and $\mathcal{OD}_2$). The attacker first identifies non-PNBs using the bias of the first $\mathcal{ID}, \mathcal{OD}_1$ pair, then focuses on the non-PNBs corresponding to $\mathcal{ID}, \mathcal{OD}_2$ that were not identified in the first step. The remaining key bits are then exhaustively searched in a three-step process. This technique was originally applied to a 6-round ChaCha cipher with a 128-bit key.

Coutinho [CPV+23b] generalize the [DGSS22] idea to an arbitrary number of differentials. Instead of using a single $\mathcal{ID}$ and two $\mathcal{OD}$, they consider multiple distinct input differences $\mathcal{ID}_i$ and corresponding output differences $\mathcal{OD}_1$ for $1 \leq i \leq r$. This method, which extends the traditional two-step key recovery approach to an $r+1$-step method, is particularly advantageous when the PNB set size is large. The process involves iteratively identifying significant key bits at each step, and progressively narrowing down the key space until the remaining unknown key bits are found via brute-force search.

The attack's complexity is defined by the number of significant key bits and the data samples needed at each step, with the total data complexity being the sum of the complexities at each step. Importantly, since these attacks are applied independently with different samples, statistical dependencies between differentials are unlikely to affect the final complexity. Coutinho introduced a distinguisher requiring 2^{214} operations on ChaCha 7 and introduced a novel linear approximation for the ChaCha sub-round.

Coutinho introduced a key-recovery attack on Salsa 8 and Salsa 7. Coutinho mounted an attack on Salsa 8 involves a differential correlation for $\Delta x_4^{(5)}[7]$ with a bias of $\varepsilon_a = 0.000305$ and a threshold $\gamma = 0.3$. The attack uses 152 Probabilistic Neutral Bits (PNBs) and has a data complexity of $2^{113.14}$ and a time complexity of $2^{217.14}$. The attack process includes 5 forward rounds and 3 backward rounds. On the other hand, the attack on Salsa 7 uses a differential-linear distinguisher for $\Delta x_4^{(5)}[7]$ with a bias of $\varepsilon_a = 0.019531$ and the same threshold $\gamma = 0.3$.

This attack employs 237 PNBs, with a data complexity of $2^{104.47}$ and a time com-

plexity of $2^{125.16}$. The cryptanalysis is performed over 5 forward rounds and 2 backward rounds. Shortly after, Dey [DGM23] reported an attack with $2^{99.48}$ operations on ChaCha 6. [DGM23] introduces a novel cryptanalytic technique that leverages multiple input-output difference $\mathcal{ID}, \mathcal{OD}$ pairs to reduce the attack complexity on the ChaCha 6. This method involves partitioning the key bits into $q+1$ subsets, denoted as $S_1, S_2, \ldots, S_q, S_{q+1}$, where each subset S_i corresponds to a specific $\mathcal{ID}, \mathcal{OD}$ pair. During the pre-processing stage, Dey identifies significant key bits by calculating biases ε_d and ε_a.

These biases are used to assess the impact of key bits on the observed differences. The key recovery process is conducted in $q+1$ stages, with each stage focusing on a subset of key bits. The overall complexity of the attack is reduced to below $2^{k/2}$, where k is the key size in bits, by accumulating the effects of multiple biases across the stages. Dey's complexity analysis begins with the expression for the bias ε as the product of forward and backward biases:

$$\varepsilon = \varepsilon_d \cdot \varepsilon_a.$$

The complexity in each stage i is given by $2^{m_i} \cdot N_i$, where m_i is the number of significant key bits in subset S_i, and N_i is the required data complexity. The total complexity is calculated as:

$$\text{Total Complexity} = \sum_{i=1}^{q} 2^{m_i} \cdot N_i + 2^{m_{q+1}},$$

with $2^{m_{q+1}}$ representing the exhaustive search required for the remaining key bits in S_{q+1}. To derive N_i, the data complexity for stage i, Dey employs hypothesis testing, comparing two scenarios: - H_0: The guessed significant key bits are incorrect, implying no bias ($\epsilon \approx 0$). - H_1: The guessed significant key bits are correct, with a bias ϵ.

Dey uses the Neyman-Pearson lemma to determine N_i:

$$N_i \approx \left(\frac{\sqrt{\alpha_i \ln 4} + \Phi^{-1}\left(\frac{Pr_{nd}}{q}\right)\sqrt{1-\epsilon^2}}{\epsilon} \right)^2,$$

where α_i is chosen to minimize false alarms, Φ^{-1} is the inverse cumulative distribution function, and Pr_{nd} is the non-detection error probability. By setting α_i to ensure a negligible false alarm rate, the complexity for each stage is minimized.

In the application to ChaCha 6, Dey selects three $\mathcal{ID}, \mathcal{OD}$ pairs, denoted as $\mathcal{ID}_1$ $\mathcal{OD}_1$, $\mathcal{ID}_2$ $\mathcal{OD}_2$, and $\mathcal{ID}_3$ $\mathcal{OD}_3$, to achieve an attack complexity of $2^{99.48}$. The order of these pairs is chosen to minimize the term $2^{m_1} \cdot N_1$, as this term predominantly determines the overall complexity. For each $\mathcal{ID}, \mathcal{OD}$ pair, Dey calculates the significant key bits and the required N_i based on the biases ϵ_d and ε_a. The number of remaining key bits after the first three stages is sufficiently small to allow for an efficient exhaustive

search in the final stage. In the first stage, for (ID_1, OD_1), Dey sets the threshold at 0.565, identifies 58 significant bits, and calculates $N_1 = 2^{41.47}$. For (ID_2, OD_2), the threshold is 0.756, yielding 56 significant bits after removing overlaps with S_1, and $N_2 = 2^{34.26}$.

The third stage, using $\mathcal{ID}_3$ $\mathcal{OD}_3$, with a threshold of 0.92, identifies 50 significant bits, with $N_3 = 2^{30.32}$. The remaining 92 key bits are recovered via exhaustive search with a complexity of 2^{92}. The total complexity of the attack is:

$$2^{58} \cdot 2^{41.47} + 2^{56} \cdot 2^{34.26} + 2^{50} \cdot 2^{30.32} + 2^{92} \approx 2^{99.48}.$$

Dey's implementation involves recovering the significant key bits for each $\mathcal{ID}, \mathcal{OD}$ pair, followed by an exhaustive search for the remaining key bits. The theoretical analysis is validated through experiments on a toy version of ChaCha, confirming the effectiveness of the multiple $\mathcal{ID}, \mathcal{OD}$ pair technique and its reduced complexity compared to traditional methods.

In 2023, Bellini [BGG+23] scrutinized differential-linear cryptanalysis, launching a key recovery attack on ChaCha 7, with a complexity of $2^{206.8}$, and a distinguisher on ChaCha 7 and ChaCha 7.5 with complexities of $2^{166.89}$ and $2^{251.54}$, respectively. Bellini et al. propose an enhanced differential-linear cryptanalysis approach for the ChaCha 7 cipher, utilizing Mixed Integer Linear Programming (MILP) to improve the efficiency and effectiveness of existing cryptanalytic techniques. The authors introduce several key enhancements that collectively reduce the complexity of differential-linear attacks on ChaCha 7 and lead to the discovery of new distinguishers for both 7-round and 7.5-round versions of the cipher.

Firstly, Bellini et al. extend the traditional search space by considering 2-bit input differences in the differential part of the attack, contrasting with the conventional single-bit differences. This extension allows for the identification of new differential paths, denoted as $\Delta_{\text{in}} \to \Delta_{\text{out}}$, which connect more effectively with the linear part of the attack.

For instance, the authors discovered differential paths such that the probability of the differential trail is improved, allowing the linear trail to exhibit a higher correlation. The new differential-linear distinguisher is mathematically represented as:

$$\text{Cor}_{\mathbf{x} \in \mathbb{F}_2^n}[\langle \Gamma_{\text{out}}, E(\mathbf{x}) \rangle \oplus \langle \Gamma_{\text{out}}, E(\mathbf{x} \oplus \Delta_{\text{in}}) \rangle] = p \cdot q^2,$$

where p represents the probability of the differential trail and q is the correlation of the linear trail. The introduction of 2-bit differences enables the authors to find trails where the product $p \cdot q^2$ is maximized, thus reducing the complexity of the distinguished.

To further enhance the attack, Bellini et al. optimize the mask selection between the differential and linear parts at round 3.5. By carefully choosing masks, they reduce

the number of active bits in the linear part, which directly impacts the correlation. This strategic choice results in a new 7-round differential-linear distinguisher requiring only $2^{166.89}$ computations, significantly improving the previously known best distinguisher for ChaCha 7, which had a complexity of 2^{214}. The paper also presents a distinguisher for ChaCha 7.5, with a complexity of $2^{251.4}$.

This distinguisher, the first of its kind for 7.5 rounds, is achieved by extending the same methodology used for the 7th round case. The authors leverage their MILP tool to identify linear trails with higher correlations, which is crucial in achieving such a low complexity. For example, a linear trail starting at round 3.5 and ending at round 7 was found with a correlation of 2^{-37}, an improvement over the previous best of 2^{-47}:

$$\mathrm{Cor}[\Gamma_{\mathrm{out}}, E_2(\mathbf{x})] = \epsilon,$$

where $\epsilon = 2^{-37}$ represents the correlation of the improved linear trail. Bellini et al. integrate the improved differential-linear distinguisher into the Probabilistic Neutral Bits (PNB) framework to perform a key-recovery attack on ChaCha 7. The computational complexity of this attack is $2^{206.8}$. The key-recovery attack is based on the following complexity formula:

$$\text{Time Complexity} = 2^{|k|-n}N + 2^{|k|-m} + 2^m,$$

where $|k|$ is the key size, n is the number of Probabilistic Neutral Bits, m is the number of significant bits, and N is the data complexity.

3.2.2 Significant Linear Approximations

In this section, we present the main linear approximations. The findings in the subsequent chapters are largely based on these Lemmas. We explore the existing linear approximation of the ChaCha stream cipher. In [CM16], the author analyzed the ChaCha quarter-round function at the bit level. In 2021, Coutinho [CSN21] updated the notation and proposed a new linear approximation. For consistency, we will use Coutinho's notation. Below, we examine the bit-level representation of the ChaCha quarter-round function.

$$x'^{(m-1)}_{a,i} = x^{(m-1)}_{a,i} \oplus x^{(m-1)}_{b,i} \oplus \Theta_i(x^{(m-1)}_a, x^{(m-1)}_b). \tag{3.3}$$
$$x'^{(m-1)}_{d,i+16} = x^{(m-1)}_{d,i} \oplus x'^{(m-1)}_{a,i}. \tag{3.4}$$
$$x'^{(m-1)}_{c,i} = x^{(m-1)}_{c,i} \oplus x'^{(m-1)}_{d,i} \oplus \Theta_i(x^{(m-1)}_c, x'^{(m-1)}_d). \tag{3.5}$$
$$x'^{(m-1)}_{b,i+12} = x^{(m-1)}_{b,i} \oplus x'^{(m-1)}_{c,i}. \tag{3.6}$$
$$x^{(m-1)}_{a,i} = x'^{(m-1)}_{a,i} \oplus x'^{(m-1)}_{b,i} \oplus \Theta_i(x'^{(m-1)}_a, x^{(m-1)}_b). \tag{3.7}$$
$$x^{(m)}_{d,i+8} = x'^{(m-1)}_{d,i} \oplus x^{(m)}_{a,i}. \tag{3.8}$$

$$x_{c,i}^{(m)} = x_{c,i}^{\prime(m-1)} \oplus x_{d,i}^{(m-1)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}). \tag{3.9}$$
$$x_{b,i+7}^{(m)} = x_{b,i}^{\prime(m-1)} \oplus x_{c,i}^{(m)}. \tag{3.10}$$

Upon reversing these equations, we obtain:

$$x_{b,i}^{\prime(m-1)} = x_{b+7}^{(m)} \oplus x_{c,i}^{(m)}. \tag{3.11}$$
$$x_{c,i}^{\prime(m-1)} = x_{c,i}^{(m)} \oplus x_{d,i}^{(m)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}). \tag{3.12}$$
$$x_{d,i}^{\prime(m-1)} = x_{a,i}^{(m)} \oplus x_{d,i+8}^{(m)}. \tag{3.13}$$
$$x_{a,i}^{\prime(m-1)} = x_{a,i}^{(m)} \oplus x_{b,i+7}^{(m)} \oplus x_c^{(m)} \oplus \Theta_i(x_a^{\prime(m-1)}, x_b^{\prime(m-1)}). \tag{3.14}$$
$$x_{b,i}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}). \tag{3.15}$$
$$x_{c,i}^{(m-1)} = \mathcal{L}_{c,i}^{(m)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_c^{(m-1)}, x_d^{\prime(m-1)}). \tag{3.16}$$
$$x_{d,i}^{(m-1)} = \mathcal{L}_{d,i}^{(m)} \oplus \Theta_i(x_a^{\prime(m-1)}, x_b^{\prime(m-1)}). \tag{3.17}$$
$$x_{a,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \Theta_i(x_a^{\prime(m-1)}, x_b^{\prime(m-1)}). \\ \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}). \tag{3.18}$$

Where

$$\mathcal{L}_{a,i}^{(m)} = x_{a,i}^m \oplus x_{b,i+7}^m \oplus x_{b,i+19}^m \oplus x_{c,i+12}^m \oplus x_{d,i}^m. \tag{3.19}$$
$$\mathcal{L}_{b,i}^{(m)} = x_{b,i+19}^m \oplus x_{c,i}^m \oplus x_{c,i+12}^m \oplus x_{d,i}^m. \tag{3.20}$$
$$\mathcal{L}_{c,i}^{(m)} = x_{a,i}^m \oplus x_{c,i}^m \oplus x_{d,i}^m \oplus x_{d,i+8}^m. \tag{3.21}$$
$$\mathcal{L}_{d,i}^{(m)} = x_{a,i}^m \oplus x_{a,i+16}^m \oplus x_{b,i+7}^m \oplus x_{c,i}^m \oplus x_{d,i+24}^m. \tag{3.22}$$

Lemma 3.1 ([CM16]). *In the least significant bit (LSB) positions, the quarter-round function allows us to set $i = 0$ and precisely estimate the linear correlation between the $(m-1)$-th and m-th rounds with a probability of 1. Let*

$$\Delta A^{(m)} = \Delta x_{\alpha,0}^{(m)} \oplus \Delta x_{\beta,7}^{(m)} \oplus \Delta x_{\beta,19}^{(m)} \oplus \Delta x_{\gamma,12}^{(m)} \oplus \Delta x_{\delta,0}^{(m)}. \tag{3.23}$$
$$\Delta B^{(m)} = \Delta x_{\beta,19}^{(m)} \oplus \Delta x_{\gamma,0}^{(m)} \oplus \Delta x_{\gamma,12}^{(m)} \oplus \Delta x_{\delta,0}^{(m)}. \tag{3.24}$$
$$\Delta C^{(m)} = \Delta x_{\delta,0}^{(m)} \oplus \Delta x_{\gamma,0}^{(m)} \oplus \Delta x_{\delta,8}^{(m)} \oplus \Delta x_{\alpha,0}^{(m)}. \tag{3.25}$$
$$\Delta D^{(m)} = \Delta x_{\delta,24}^{(m)} \oplus \Delta x_{\alpha,16}^{(m)} \oplus \Delta x_{\alpha,0}^{(m)} \oplus \Delta x_{\gamma,0}^{(m)} \oplus \Delta x_{\beta,7}^{(m)}. \tag{3.26}$$

Then, the following equations for four biases hold:

$$\begin{aligned}\left|\varepsilon(A^{(m)})\right| &= \left|\varepsilon(x_{\alpha}^{(m-1)}[0])\right|.\\ \left|\varepsilon(B^{(m)})\right| &= \left|\varepsilon(x_{\beta}^{(m-1)}[0])\right|.\\ \left|\varepsilon(C^{(m)})\right| &= \left|\varepsilon(x_{\gamma}^{(m-1)}[0])\right|. \text{ and}\\ \left|\varepsilon(D^{(m)})\right| &= \left|\varepsilon(x_{\delta}^{(m-1)}[0])\right|.\end{aligned}$$

where these relations are divided into two cases depending on m,

1. *If m is an odd number:*

$$\begin{aligned}(\alpha,\beta,\gamma,\delta) \in \{&(0,4,8,12),(1,5,9,13),\\ &(2,6,10,14),(3,7,11,15)\}.\end{aligned}$$

2. *If m is an even number:*

$$\begin{aligned}(\alpha,\beta,\gamma,\delta) \in \{&(0,5,10,15),(1,6,11,12),\\ &(2,7,8,13),(3,4,9,14)\}.\end{aligned}$$

For the proof, please refer to [CM16]

Lemma 3.2 ([CM16])**.** *To derive the linear approximation for a half-round of ChaCha, [CM16] introduced the following lemma:*

$$x_{a,i}^{(m)} = x_{a,i}^{(m+0.5)} \oplus x_{b,i+12}^{(m+0.5)} \oplus x_{c,i}^{(m+0.5)} \oplus C_{carry,i}^{1}. \tag{3.27}$$
$$x_{b,i}^{(m)} = x_{b,i+12}^{(m+0.5)} \oplus x_{c,i}^{(m+0.5)}. \tag{3.28}$$
$$x_{c,i}^{(m)} = x_{c,i}^{(m+0.5)} \oplus x_{d,i}^{(m+0.5)} \oplus C_{carry,i}^{2}. \tag{3.29}$$
$$x_{d,i}^{(m)} = x_{d,i+16}^{(m+0.5)} \oplus x_{a,i}^{(m+0.5)}. \tag{3.30}$$

Remarkably, we observe that the bias of variables such as $x_{b[i]}^{(m)}$ and $x_{d[i]}^{(m)}$ can be derived from round $m + 0.5$ without any reduction in their value for all i. While the word positions $x_{b[i]}^{(m)}$ and $x_{d[i]}^{(m)}$ can be extended to a half round with probability 1 for all bit positions. However, this is not true for $x_{a[i]}^{(m)}$ and $x_{c[i]}^{(m)}$, which occur with a probability less than 1. Therefore, we describe the following Lemma concerning the half-round extension of $x_{a[i]}^{(m)}$ and $x_{c[i]}^{(m)}$.

Lemma 3.3 ([CPV+23b])**.** *For $i = 0$, the subsequent linear approximations are valid with a certainty of 1, assuming a single active input bit in half-round $m-1$ and several output bits in half-round $m + 0.5$.*

$$x_{c,i}^{(m)} = x_{c,i}^{(m+0.5)} \oplus x_{d,i}^{(m+0.5)}. \tag{3.31}$$

$$x_{a,i}^{(m)} = x_{a,i}^{(m+0.5)} \oplus x_{b,i+7}^{(m+0.5)} \oplus x_{c,i}^{(m+0.5)}. \tag{3.32}$$

For $i > 0$, the following linear approximations hold with a probability of $\frac{1}{2}(1 + \frac{1}{2})$, given a single active input bit in half-round $m-1$ and several output bits in half-round m.

$$x_{c,i}^{(m)} = x_{c,i}^{(m+0.5)} \oplus x_{d,i}^{(m+0.5)} \oplus x_{d,i-1}^{(m+0.5)}. \tag{3.33}$$

$$x_{a,i}^{(m)} = x_{a,i}^{(m+0.5)} \oplus x_{b,i+7}^{(m+0.5)} \oplus x_{c,i}^{(m+0.5)} \oplus x_{b,i+6}^{(m+0.5)} \oplus x_{c,i-1}^{(m+0.5)}. \tag{3.34}$$

Lemma 3.4 ([CM16]). *When there is a single active input bit in round $m-1$ and multiple active output bits in round m the following statement holds for $i > 0$:*

$$\begin{aligned} x_{b,i}^{(m-1)} &= x_{b,i+19}^{(m)} \oplus x_{c,i+12}^{(m)} \oplus x_{d,i}^{(m)} \oplus x_{c,i}^{(m)} \\ &\quad \oplus x_{d,i-1}^{(m)}, \quad W.P. \frac{1}{2}\left(1 + \frac{1}{2}\right). \end{aligned} \tag{3.35}$$

$$\begin{aligned} x_{a,i}^{(m-1)} &= x_{a,i}^{(m)} \oplus x_{b,i+7}^{(m)} \oplus x_{b,i+19}^{(m)} \oplus x_{c,i+12}^{(m)} \oplus x_{d,i}^{(m)} \\ &\quad \oplus x_{b,i+18}^{(m)} \oplus x_{c,i+11}^{(m)} \oplus x_{d,i-2}^{(m)} \oplus x_{d,i+6}^{(m)}, \\ &\quad W.P. \frac{1}{2}\left(1 + \frac{1}{2^4}\right). \end{aligned} \tag{3.36}$$

$$\begin{aligned} x_{c,i}^{(m-1)} &= x_{d,i}^{(m)} \oplus x_{c,i}^{(m)} \oplus x_{d,i+8}^{(m)} \oplus x_{a,i}^{(m)} \oplus x_{a,i-1}^{(m)} \\ &\quad \oplus x_{d,i+7}^{(m)} \oplus x_{d,i-1}^{(m)}, \quad W.P. \frac{1}{2}\left(1 + \frac{1}{2^2}\right). \end{aligned} \tag{3.37}$$

$$\begin{aligned} x_{d,i}^{(m-1)} &= x_{d,i+24}^{(m)} \oplus x_{a,i+16}^{(m)} \oplus x_{a,i}^{(m)} \oplus x_{c,i}^{(m)} \oplus x_{b,i+7}^{(m)} \\ &\quad \oplus x_{c,i-1}^{(m)} \oplus x_{b,i+6}^{(m)}, \quad W.P. \frac{1}{2}\left(1 + \frac{1}{2}\right). \end{aligned} \tag{3.38}$$

Using Lemma 3.1, we can obtain a linear approximation for one round of ChaCha with (probability 1). Conversely, Lemma 3.4 offers a linear approximation with a probability less than one. Specifically, equations 3.36 and 3.37, which relate to the words 'A' and 'C', have lower probabilities of occurrence. The selection of these words can notably impact the overall complexity of an attack. Therefore, it is recommended to avoid using these words.

Lemma 3.5. *([CSN21]). Linear approximation between two input bits in $m-1$ rounds and multiple output bits in m rounds $x_{\lambda,i}^{(m-1)} \oplus x_{\lambda,i-1}^{(m-1)} = \mathcal{L}_{\lambda,i}^{(m)} \oplus \mathcal{L}_{\lambda,i-1}^{(m)}$ is satisfied with probability $\frac{1}{2}\left(1 + \frac{1}{2^\sigma}\right)$, where $(\lambda, \sigma) \in \{(a,3),(b,1),(c,2),(d,1)\}$ for $i > 0$.*

Lemma 3.6. *([CSN21]). For $i > 0$,*

$$x_{a,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus x_{b,i+6}^{(m)} \oplus x_{b,i+18}^{(m)} \oplus x_{c,i+11}^{(m)} \oplus x_{d,i-1}^{(m)}.$$

is satisfied with probability $\frac{1}{2}\left(1 + \frac{1}{2^3}\right)$.

Lemma 3.7. *(Equation 19 and 25 in Lemma 9 of [CSN21] respectively). The subsequent linear approximations between multiple input bits in the $m-1$ rounds and multiple output bits in the m rounds are valid with the following probability.*

$$x_{b,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)} \quad w.p.\ \frac{1}{2}\left(1 + \frac{1}{2}\right) \quad for\ i > 0. \tag{3.39}$$

$$x_{a,i-1}^{(m-1)} \oplus x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{d,i-2}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)}$$
$$w.p.\ \frac{1}{2}\left(1 + \frac{1}{2^4}\right) \quad for\ i > 1. \tag{3.40}$$

Lemma 3.8. *([BBC+22]) Let $s = y_0 \oplus \overline{y_1}$ and let $i \geq 3$. Let $S_{b_0 b_1} := \{(y_1, y_0) \in \mathbb{F}_2^{2m} \mid s[i-1] = b_0$ and $s[i-2] = b_1\}$. We have*

$$z_0[i] \approx \begin{cases} y_0[i] \oplus y_1[i] \oplus y_0[i-1], & with\ cor.\ -1,\ if\ (y_1, y_0) \in S_0^* \\ y_0[i] \oplus y_1[i] \oplus y_0[i-2], & with\ cor.\ -1,\ if\ (y_1, y_0) \in S_{10} \\ y_0[i] \oplus y_1[i] \oplus y_0[i-3], & with\ cor.\ -2^{-1},\ if\ (y_1, y_0) \in S_{11} \end{cases}$$

where $S_0^* = S_{00} \cup S_{01}$.

3.2.3 Higher Order Differential-Linear Attack

Kai [HPTY23] applied higher-differential-linear attacks to ChaCha 3.5, ChaCha 4, and ChaCha 4.5, achieving biases of 1/2, $2^{-1.19}$, and $2^{-4.81}$, respectively. In the area of ChaCha security analysis, it is clear that all current attacks have focused solely on single-bit differential or single-bit differential-linear techniques. Consequently, there is a notable gap in the literature concerning higher-order differential and higher-order differential-linear attacks on ChaCha.

This dissertation aims to address this gap by investigating higher-order differential-linear cryptanalysis, thereby assessing the resilience of the ChaCha stream cipher against a more advanced adversary model. To provide a clearer understanding of higher-order differential cryptanalysis, we review Kai's work on higher-order differential-linear cryptanalysis [HPTY23].

Kai explored higher-order differential-linear attacks from an algebraic viewpoint and introduced a higher-order differential-linear distinguisher for several internal rounds

of ChaCha. The research involved finding high-bias second-order differential-linear (DL) with a single bit $\mathcal{OD}$ and then appending a 1.5-round deterministic linear approximation. The 3.5-round $2nd$ order higher-order differential-linear was shown to have a bias close to $\frac{1}{2}$.

In this work, Kai used $\Delta X^{(0)}_{12[0]} \oplus \Delta X^{(0)}_{14[0]}$ as the $\mathcal{ID}$ position, with the $\mathcal{OD}$ selected as $\Delta X^{(2)}_{8,[0]}$.

$$\begin{aligned} X^{(2)}_{8,[0]} = X^{3.5}_{0,[0]} \oplus X^{3.5}_{0,[8]} \oplus X^{3.5}_{3,[0]} \oplus X^{3.5}_{4,[12]} \oplus X^{3.5}_{9,[0]} \oplus X^{3.5}_{11,[0]} \\ \oplus X^{3.5}_{12,[0]} \oplus X^{3.5}_{15,[16]} \oplus X^{3.5}_{15,[24]}. \end{aligned} \tag{3.41}$$

Kai discovered a 4-round second-order higher-order differential-linear (HDL) with a $2^{-1.19}$ bias. The input difference $\mathcal{ID}$ was positioned as $\Delta X^{(0)}_{13[16]} \oplus \Delta X^{(0)}_{14[0]}$, and the output difference $\mathcal{OD}$ was chosen as $\Delta X^{(2.5)}_{8,[0]}$.

$$\begin{aligned} X^{(2.5)}_{8,[0]} = X^{4}_{1,[0]} \oplus X^{4}_{1,[16]} \oplus X^{4}_{2,[0]} \oplus X^{4}_{6,[7]} \oplus X^{4}_{8,[0]} \oplus X^{4}_{11,[0]} \\ \oplus X^{4}_{12,[24]} \oplus X^{4}_{13,[0]} \oplus X^{4}_{13,[8]}. \end{aligned} \tag{3.42}$$

For ChaCha 4.5, Kai identified a second-order differential-linear bias of roughly $2^{-4.81}$. The input difference $\mathcal{ID}$ was placed at $\Delta X^{(0)}_{14[12]} \oplus \Delta X^{(0)}_{15[15]}$, and the output difference $\mathcal{OD}$ was selected as $\Delta X^{(3)}_{8,[0]}$. The 1.5-round linear approximation occurred with a probability of 1/2.

$$\begin{aligned} X^{(3)}_{8,[0]} = X^{4.5}_{0,[0]} \oplus X^{4.5}_{0,[8]} \oplus X^{4.5}_{1,[0]} \oplus X^{4.5}_{5,[12]} \oplus X^{4.5}_{9,[0]} \oplus X^{4.5}_{11,[0]} \\ \oplus X^{4.5}_{12,[16]} \oplus X^{4.5}_{12,[24]} \oplus X^{4.5}_{15,[0]}. \end{aligned} \tag{3.43}$$

Kai combined the initial 3-round second-order HDL approximation with an additional 1.5-round linear approximation, creating a 4-round second-order HDL distinguisher with a bias of approximately $2^{-4.81}$. The biases found in the second-order DL distinguishers for these three versions exceeded those of all previous DL distinguishers. Due to these significantly higher biases, it is possible to strengthen the distinguishing attacks on ChaCha 3.5, ChaCha 4, and ChaCha 4.5.

3.2.4 Attacks on ChaCha permutation

The research domain surrounding Salsa 20 and ChaCha primarily focuses on key recovery and distinguishing attacks. However, studies on permutation are less prevalent compared to those on key recovery and distinguishing attacks. In this section, we will review the most significant studies on ChaCha permutation. Stefano [BBM20] studied

the rotational cryptanalysis of the ChaCha stream cipher, specifically focusing on its underlying permutation. The paper examines whether ChaCha's permutation behaves like a random permutation for up to 17 rounds.

The Stefano derives lower and upper bounds for the probability of rotational collisions occurring within a single ChaCha quarter round. Let p denote the likelihood of rotational collision propagation through the quarter round. The derived bounds are as follows:

$$D^3P(r,w) \leq p \leq \left(\frac{D(2r+2)(2w-r+2)}{9 \cdot 2^w}\right)^2$$

where D represents the probability of rotational pairs propagating through modular addition, r is the rotational amount, and w is the bit size (in ChaCha's case, $w = 32$). Stefano [BBM20] extends these bounds to the full round and the entire permutation, showing that the probability of a parallel rotational collision after 17 rounds is significantly higher in ChaCha compared to a random permutation. Specifically, for 17 rounds, the probability of a rotational collision is less than 2^{-488} for ChaCha, whereas for a random permutation of the same input size, it is 2^{-511}. The paper also introduces a distinguisher that uses these bounds to differentiate between ChaCha and random permutations.

The distinguisher checks if a rotational collision occurs after making multiple oracle calls. If such a collision is found, it suggests the oracle is running the ChaCha permutation. The complexity of this distinguisher is dominated by the number of calls to the oracle, which increases with the number of rounds. Experimental results on toy versions of ChaCha, using reduced word sizes of 4, 5, and 6 bits, confirmed the theoretical bounds. The authors in [BBM20] conducted exhaustive searches through all possible values of (x_0, x_1, x_2, x_3), and observed the frequency of rotational collisions, aligning well with their predicted probabilities.

In conclusion, the study demonstrates that ChaCha's permutation is distinguishable from a random permutation for up to 17 rounds due to the higher occurrence of rotational collisions. While this does not imply an attack on the ChaCha stream cipher, it highlights interesting non-random properties of the ChaCha permutation.

The [DS23] presents improvements in differential-linear distinguishing attacks on the ChaCha permutation. The authors introduce a new single-bit differential distinguisher for ChaCha 3.5, which helps construct a more efficient differential-linear distinguisher. The paper presents that a 7-round ChaCha permutation can be distinguished with a time complexity of 2^{207}. Furthermore, the paper extends the differential-linear distinguisher to 7.25 rounds. The differential-linear attack divides the cipher into three parts: E_1, E_m, and E_2, where E_1 and E_m correspond to differential cryptanalysis, and E_2 corresponds to linear cryptanalysis. The authors in [DS23] analyze the propagation

of differential and linear correlations through these parts.

For the 3.5-round differential distinguisher, they identify the input difference at position $(p, 6)$, with $p \in \{12, 13, 14, 15\}$, which results in an output difference with a differential correlation of approximately $2^{-28.65}$. This is computed based on the quarter-round function (QRF) and the symmetry in ChaCha's round updates. The [DS23] introduces a differential-linear distinguisher for the ChaCha 7.25 rounds permutation. This extension is achieved by utilizing a new linear approximation path that connects active bits from the 6th to the 7th round with high probability.

Specifically, they derive a differential-linear correlation of approximately 2^{-113} for the 7.25 rounds, leading to a time complexity of 2^{231} for the distinguisher. This marks the first distinguishing attack that extends beyond 7 rounds of the ChaCha permutation. Key linear approximations are introduced, expressed through several lemmas, and validated theoretically and experimentally. For example, Lemma 4 in [DS23] presents that for two consecutive active bits in one round, multiple active bits in the next round can be approximated with a probability of $\frac{1}{2}\left(1 + \frac{1}{2^n}\right)$, where n depends on the specific bits and rounds. These approximations are crucial for extending the distinguisher to 7.25 rounds, leading to a differential-linear correlation of 2^{-113} for the 7.25-round distinguisher, with a corresponding time complexity of 2^{231}.

Chapter 4

Differential Cryptanalysis of Salsa 20 and ChaCha

This chapter studies the differential cryptanalysis of Salsa 20 and ChaCha. First, we study the differential cryptanalysis of Salsa 20 based on the analysis of PNBs and attack the reduced round of Salsa 20 and ChaCha. We presented the distribution of neutral measures across different intermediate rounds of Salsa and ChaCha. We present that the neutral measure of keybits is mainly affected by the $\mathcal{OD}$ position. In addition, we mounted a key-recovery attack on Salsa 8 and ChaCha 7.25 with time complexity of $2^{241.62}$, $2^{254.01}$ and data complexity of $2^{31.5}$, $2^{51.81}$ respectively.

4.1 Examination of Probabilistic Neutral Bits

This section explores the neutrality measure of 256 key bits across all 512 output differential positions. We examine the experimental findings and identify the output differential position with the highest neutrality measure, which serves as the attack target for both Salsa 20 and ChaCha. We show that the neutrality measure significantly depends on the output differential bit position rather than the input differential bit position.

We introduce a differential attack on Salsa 20/8 with a time complexity of $2^{241.62}$ and a data complexity of $2^{31.5}$. We evaluated all feasible internal rounds and identified the optimal one for attacking Salsa 20/8. Specifically, we examined $4r$, $4.25r$, $4.5r$, $4.75r$, $5r$, $5.25r$, $5.5r$, and $5.75r$ internal rounds, concluding that the $4.75r$ internal round is optimal for attacking Salsa 20/8. Furthermore, we achieved a key-recovery attack on ChaCha 7.25 with a time complexity of $2^{254.01}$ and a data complexity of $2^{51.81}$.

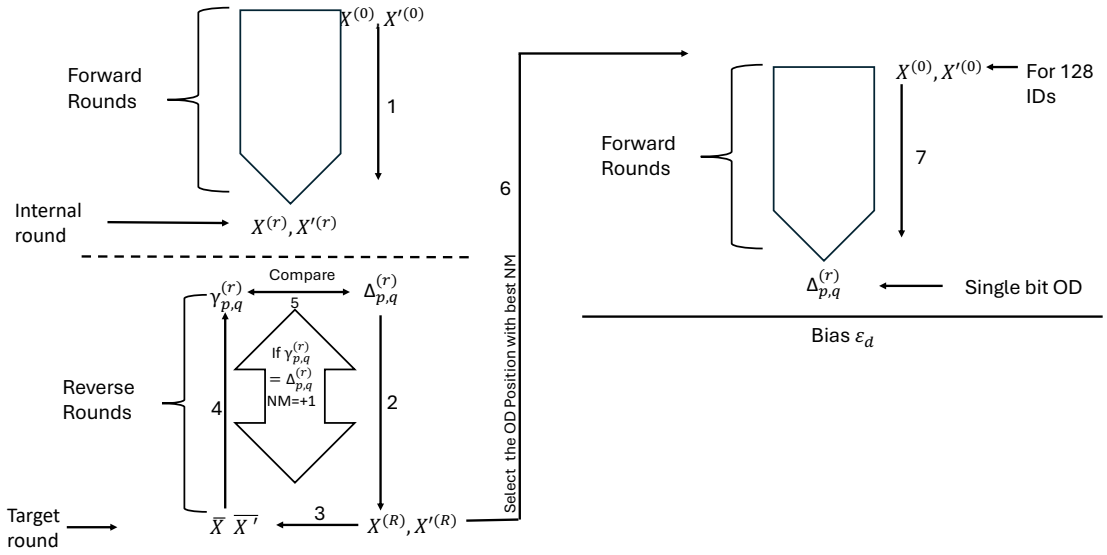

Figure 4.1: $\mathcal{ID}, \mathcal{OD}$ Selection Procedure.

4.1.1 Analysis of PNBs

The cryptanalysis of Salsa 20 and ChaCha involves studying the $\mathcal{ID}$-$\mathcal{OD}$ with the highest bias ε_d among all possible pairs. The $\mathcal{ID}$ is determined and followed by selecting the corresponding $\mathcal{OD}$ with the highest forward bias. Essentially, previous studies have focused on analyzing the differential bias at specific $\mathcal{ID}, \mathcal{OD}$ pairs and then attempted to identify the subset of PNBs to target the desired round of Salsa 20. However, based on our experimental results and previous research findings [AFK+08], the neutrality measures of all 256 key bits primarily rely on $\mathcal{OD}$ bits.

Consequently, when executing a key-recovery attack within this framework, it becomes challenging to determine whether the combination of $\mathcal{ID}, \mathcal{OD}$ and the subset of PNBs is truly optimal. This section delves into a comprehensive analysis of the neutrality measures of 256 key bits across all possible $\mathcal{OD}$ bits. Additionally, we examine the conditions and factors that may contribute to high neutral measures, considering that the size of PNBs influences the time complexity of the attack. Presumably, no detailed study on analyzing PNBs has been reported thus far. If we can elucidate the conditions that lead to high neutral measures γ_κ, it may suggest that existing attacks still have the potential for improvement. We employed Algorithm 3 to compute the neutrality measures of 256 key bits across 512 $\mathcal{OD}$ bits. To determine an optimal $\mathcal{ID}$ position, we explored all 128 possible $\mathcal{ID}$ positions.

Surprisingly, we did not observe a significant impact of the $\mathcal{ID}$ on the neutrality measure of key bits. Consequently, we decided to select a random $\mathcal{ID}$ at this experimental stage[1] In this section, we study the neutrality measure of all possible rounds of Salsa 20 and ChaCha (i.e., quarter round, half round, and three quarters). Further-

[1]In our experiment, we utilized the $\mathcal{ID}$ (7,31) as reported in [AFK+08].

Algorithm 3 Optimal $\mathcal{OD}$ Bit Determination

Input: Random values (X, X', Z, Z')
Output: The $\mathcal{OD}$ bit with the highest neutrality measure

1: Trigger random keywords $k = (k_0, \ldots, k_7)$
2: Choose a random $\mathcal{ID}$ $\Delta_i^{(0)}[j]$
3: Compute $v = (v_0, v_1)$ and $t = (t_0, t_1)$
4: Initialize $X^{(0)}$
5: Set $X'^{(0)} = X^{(0)} \oplus \Delta_i^{(0)}[j]$
6: Derive $(X^{(r)}, X'^{(r)})$ and $(X^{(R)}, X'^{(R)})$ from $(X^{(0)}, X'^{(0)})$
7: Calculate $\mathcal{OD}$ bits $\Delta_p^{(r)}[q] = X_p^{(r)}[q] \oplus X_p'^{(r)}[q]$ for all indices p and q
8: Obtain keybits $Z = X^{(0)} + X^{(R)}$ and $Z' = X'^{(0)} + X'^{(R)}$ from $(X^{(R)}, X'^{(R)})$
9: Flip a keybit κ ($\kappa \in \{0, \ldots, 255\}$)
10: Calculate $\overline{X}^{(0)}$ and $\overline{X'}^{(0)}$ from initial states $(X^{(0)}, X'^{(0)})$
11: Calculate $(Y^{(r)}, Y'^{(r)})$ using $Z - \overline{X}^{(0)}$ and $Z' - \overline{X'}^{(0)}$ as inputs to the inverse function.
12: Determine $\Gamma_p^{(r)}[q] = Y_p^{(r)}[q] \oplus Y_p'^{(r)}[q]$ for every combination of p and q
13: Increment the tally for each p, q, and κ if $\Delta_p^{(r)}[q] = \Gamma_p^{(r)}[q]$
14: Determine the probability by dividing the sum of occurrences where $\Delta_p^{(r)}[q] = \Gamma_p^{(r)}[q]$ by the number of key trials and $\mathcal{ID}$ samples

more, we study the attack impact on ChaCha stream cipher and consider the median bias $|\varepsilon_d^*|$ for the attack complexity estimation.

4.1.2 The Impact on Salsa 20

This subsection presents the experimental results for the neutrality measure of all possible internal rounds of the Salsa 20 stream cipher. We studied the neutrality measure of 256 keybits for all possible $\mathcal{OD}$ positions given the $4, 4.25, 4.5, 4.75$, and $5, 5.25, 5.5, 5.75$ rounds. We investigated the neutrality measure of key bits across all possible $\mathcal{OD}s$ by performing experiments with a complexity of 2^{30} (i.e., 2^6 key trials and 2^{24} IV samples).

The probability was determined over the key trials, nonce, and counter. Let $\mathcal{X}$ represent the distribution of $\Delta_p^{(r)}[q] = \Gamma_p^{(r)}[q]$ from r-round internal state matrices in a true random number generator, and let $\mathcal{Y}$ denote the distribution of $\Delta_p^{(r)}[q] = \Gamma_p^{(r)}[q]$ from the r-round internal state matrices of Salsa 20. The target event occurs in $\mathcal{X}$ with a probability of $\frac{1}{2}$ and in $\mathcal{Y}$ with a probability of γ_κ; hence, the number of samples required to distinguish between $\mathcal{X}$ and $\mathcal{Y}$ is $\mathcal{O}(\frac{2}{\gamma_\kappa^2})$.

Our experimental results are considered reliable when the derived neutral measures γ_κ exceed $2^{-14.5}$, given that we used a total of 2^{30} IV samples and key trials.

Figures. 4.2 and 4.3 illustrate the experimental results from a thorough analysis of key bit neutrality measures across all possible $\mathcal{OD}$ bit positions in Salsa 20. In these

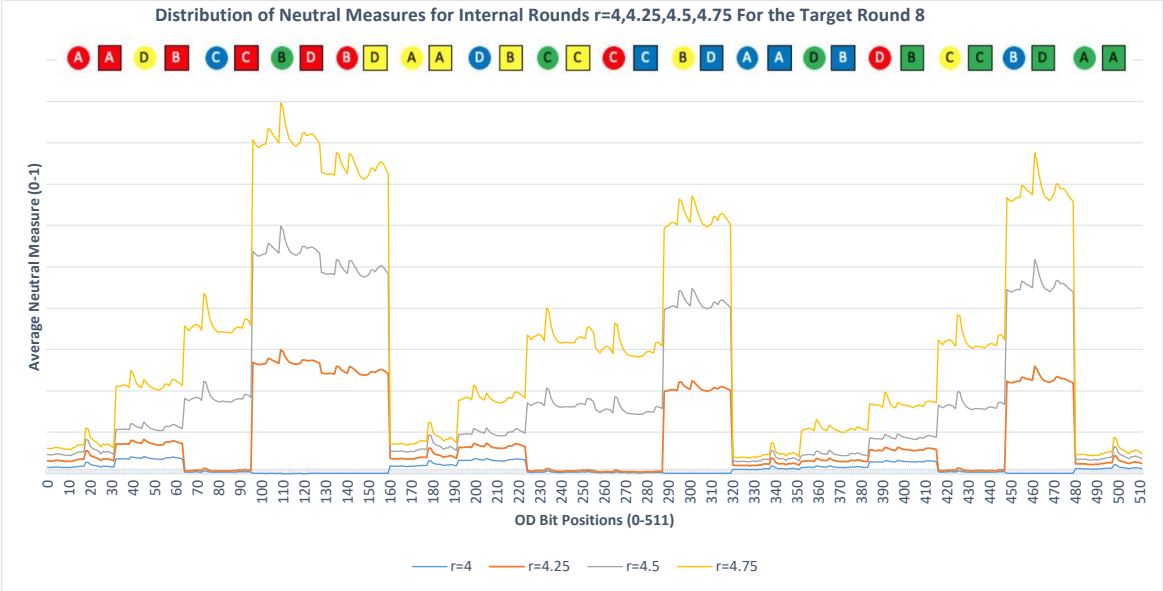

Figure 4.2: Neutral measures of internal rounds r

graphs, the vertical axis depicts the average neutrality measures for key bits at each $\mathcal{OD}$ position, while the horizontal axis represents the $\mathcal{OD}$ bit positions.

The blue, orange, gray, and yellow lines represent the distribution of neutrality measures in the full internal round, quarter internal round, half internal round, and three-quarters internal rounds, respectively. We analyzed all potential internal rounds within the 4th and 5th rounds. Figures 4.2 and 4.3 display results for $r = 4, r = 4.25, r = 4.5, r = 4.75$, and $r = 5, r = 5.25, r = 5.5, r = 5.75$, respectively. Circles and squares at the top of the graphs denote input word positions, such as vectors (A, B, C, D), for the inverse quarter-round function in odd and even rounds, respectively.

If these symbols share the same color, the word position for the inverse quarter-round function is the same, regardless of whether the Salsa 20 round is odd or even. From Figures. 4.2 and 4.3, we observe the following:

- The neutrality measure distribution of key bits varies with each $\mathcal{OD}$ bit position, as evidenced by Figures 4.2 and 4.1.
- Internal rounds $4.25r$, $4.5r$, and $4.75r$ produce identical neutrality measures for some $\mathcal{OD}$ positions. Specifically, the $\mathcal{OD}$ words at positions $3rd, 4th, 5th, 9th, 10th, 14th, 15th$ exhibit the same neutrality measure. However, these $\mathcal{OD}$ positions differ for internal rounds $r = 5, r = 5.25, r = 5.5, r = 5.75$. This will be further examined in Subsection 4.1.4.
- The inverse-quarter round function of Salsa 20 influences the neutrality measure of key bits. As the number of reverse rounds increases, the neutrality measure of key bits decreases. Further discussion on this will be in Subsection 4.1.4.

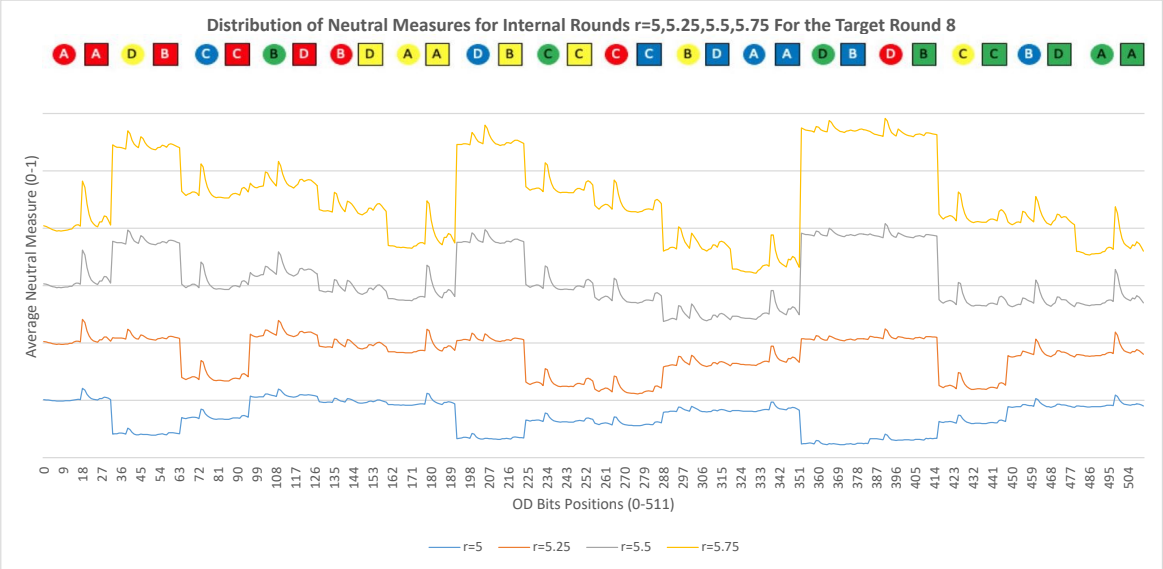

Figure 4.3: Neutral measures of internal rounds r.

- $\mathcal{OD}$ bit positions with high neutrality measures vary across different word positions.
- The distributions of neutrality measures with respect to $\mathcal{OD}$ bit positions are similar, regardless of the internal round r.

Table 4.1: For $R = 7$, the $\mathcal{OD}$ bit with the optimal neutral measure using 2^{30} samples.

Internal Round	$\mathcal{OD}$ Position	Average Neutral Measure
$3r$	$\Delta_{11}^{(3)}[0]$	$2^{-2.63846}$
$3.5r$	$\Delta_{9}^{(3.5)}[0]$	$2^{-1.38836}$
$4r$	$\Delta_{4}^{(4)}[0]$	$2^{-0.62482}$

4.1.3 The Impact on ChaCha

We utilized Algorithm 3 to assess the average neutrality measure of all key bits concerning every possible $\mathcal{OD}$ bit position for $r = 3$, $r = 3.5$, and $r = 4$ internal rounds. Table 4.1 summarizes our experiment's results. According to Table 4.1, regardless of the internal rounds, the $\mathcal{OD}$ bit 0 exhibits the best neutral measure. This is influenced by factors such as the cumulative number of modular subtractions, the input word position to the inverse quarter-round of ChaCha, and the structure of the ChaCha quarter-round function.

Fig. 4.4 illustrates the distribution of the neutrality measure of ChaCha20/7 considering the $r = 3$, $r = 3.5$, and $r = 4$ internal rounds. The X-axis represents the neutrality measure of 256 key bits, while the Y-axis denotes the $\mathcal{OD}$ bit position.

Table 4.1 and Fig. 4.5 indicate that a higher number of internal rounds affects the neutrality measure. Specifically, the neutrality measure for the internal round $r = 3$ is lower than for $r = 3.5$. This disparity is directly influenced by the cumulative number of modular subtractions executed for different internal rounds. We opted to target more ChaCha rounds with the internal round $r = 3.5$.

Although we could have selected the $r = 4$ internal round to attack $R = 7$, $R = 7.25$, or $R = 7.75$, the forward bias ε_d would notably decrease for the $r = 4$ internal round. Since researchers have not concentrated on evaluating the security of ChaCha7.25 rounds, we chose to use $r = 3.5$ to attack ChaCha7.25.

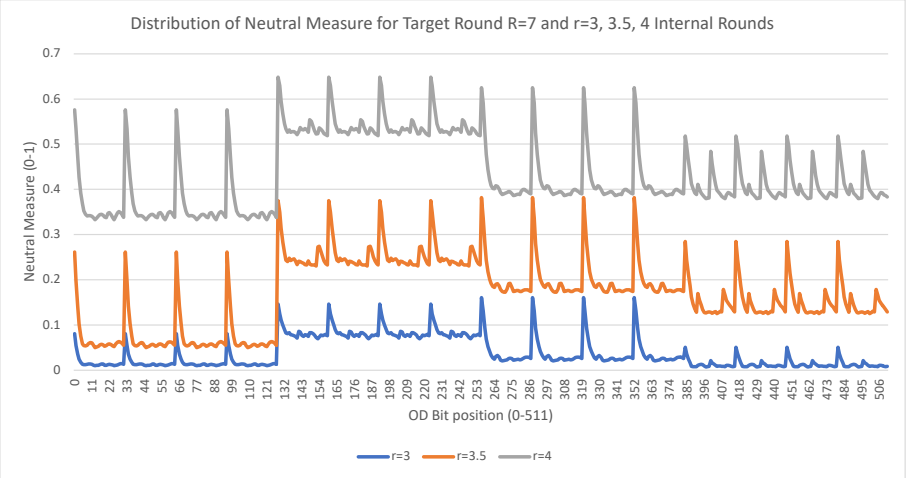

Figure 4.4: Distribution of neutral measures for $R = 7$

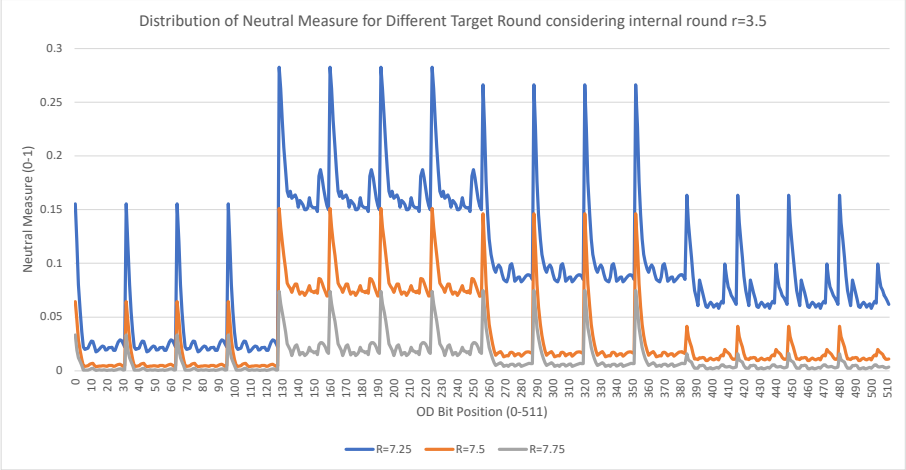

Figure 4.5: Neutral measures for $r = 3.5$ internal rounds

4.1.4 Correlation Between Neutrality Measures and Inversed Rounds

This section studies the correlation between neutrality measures and inversed rounds of Salsa 20 for quarter rounds, half rounds, and three-quarters rounds. To explore the association between the neutrality measure of key bits and the inverse quarter-round function of Salsa 20, we examine how the input word position to the inverse quarter-round function and the total number of modular additions vary for each input vector (A, B, C, D) across specific reverse rounds.

Tables 4.2 and 4.3 detail the cumulative number of modular additions for various input word positions to the inverse quarter-round function across different internal rounds, as mentioned in the respective tables above.

Table 4.2: Total modular additions in reverse rounds $3, 2.25, 2.5, 2.75$ from $R = 8$

R	Word position		Summative modular additions			
	Even round	Odd round	2.25 rounds	2.5 rounds	2.75 rounds	3 rounds
Even	$A \rightarrow$	A	24	24	24	24
	B $\rightarrow$	D	7	7	7	76
	C $\rightarrow$	C	11	11	52	52
	D $\rightarrow$	B	16	35	35	35

Table 4.3: Total modular additions in reverse rounds $4, 3.25, 3.5, 3.75$ from $R = 8$

R	Input word position		Summative modular additions			
	Even round	Odd round	3.25 rounds	3.5 rounds	3.75 rounds	4 rounds
Even	$A \rightarrow$	A	112	112	112	112
	B $\rightarrow$	D	76	164	164	164
	C $\rightarrow$	C	52	52	241	241
	D $\rightarrow$	B	35	35	35	277

The column R indicates the number of target rounds in our attack (i.e., 8 rounds in this investigation). The combination of input word positions for the inverse quarter-round function differs depending on whether the target round R is odd or even. Specifically, the input word positions, represented as a vector (A, B, C, D), for the inverse quarter-round function vary between odd and even rounds. The reverse round is executed in even rounds (row rounds) when the target round R is even.

Conversely, the reverse round is executed in odd rounds (column rounds) when the target round R is odd. To illustrate, let's consider the scenario where the number of target rounds R is even and the input word position for the quarter-round function is B. When the number of inverse rounds is 3, the word position shifts from B (even-numbered round) to D (odd-numbered round) through the execution of the inverse quarter-round function.

The inverse quarter-round function rearranges the same element within the Salsa 20 matrix differently depending on whether the reverse round is odd or even. For instance, element X_3 is denoted as D in the vector (A, B, C, D) when the target round is even, while it is represented as B in odd rounds. Similarly, in scenarios with three inverse rounds, the transition of word positions goes from B (even round) to D (odd round) back to B (even round). The cumulative modular addition columns in Tables 4.2 and 4.3 showcase the total number of modular additions executed by the inverse quarter-round function for each transition of word positions across different inverse rounds.

At this juncture, our focus lies solely on the cumulative modular addition execution due to its critical role in upholding the security of ARX ciphers. As evidenced by Tables 4.2 and 4.3, the execution of modular addition varies based on the input word position to the inverse round function and the number of reverse rounds. For instance, consider the scenario where the target round R is even, the word position transition is A (even round) to A (odd round), and the inverse round numbers are $2.25r, 2.5r, 2.75r, and 3r$, respectively. Here, the cumulative modular additions remain at 24 for each round. Similarly, with three inverse round functions, the maximum and minimum values of cumulative modular addition are 76 and 24, respectively.

Consequently, as the number of inverse rounds increases, the disparity between the maximum and minimum values of cumulative modular addition also widens. Equally significant, the cumulative modular additions executed for 2.25 reverse rounds are equivalent to those for 3 reverse rounds for the input word position A. Likewise, the cumulative modular additions for word positions, such as the transition from word position B to D and the transition from word position D to B, remain unchanged for specific internal rounds like $2.25, 2.5$, and 2.75. This analysis extends to the forward quarter-round function of Salsa 20 as well. Given these observations, it becomes apparent that targeting higher rounds of Salsa 20, such as $R = 8.75$, would essentially be equivalent to targeting $R = 8.25$ for certain specific words. This highlights a vulnerability in the design of the Salsa 20 quarter-round function.

Table 4.4 displays the results obtained from Algorithm 3. Notably, the $\mathcal{OD}$ word with the best neutrality measure in internal rounds r $4.25r, 4.5r, 4.75r$, with corresponding reverse rounds $3.25, 3.5, 3.75$, appears in X_3, which corresponds to the input word position $D \rightarrow B$ in Table 4.3. This transition exhibits the same cumulative number of modular additions (i.e., 35) and generates an identical average neutrality measure $\gamma_k = 0.598177289$. Similar observations apply to internal rounds $4r, 5r, 5.25r, 5.5r, 5.75r$. Experimental results are depicted in Figures 4.2 and 4.1, which graphically represent our findings from Tables 4.2, 4.3, and 4.4. The input word position inducing lower neutrality measures in the $4r$ internal round, D, corresponds to X_3 in the Salsa 20 matrix, corresponding to the input word position $D \rightarrow B$ when

$R - r = 4$, with 277 cumulative modular additions. The transitions from $D \rightarrow B$ traverse high cumulative modular additions for the internal $4r$ round. Additionally, X_4, X_9, and X_{14} produce the lowest average neutrality measure when $r = 4$, corresponding to word transitions from $D \rightarrow B$. Similar observations apply across all potential internal rounds. In summary, the neutrality measure is contingent upon the input word position to the inverse round function, with the cumulative number of modular additions exerting significant influence. The circumstances leading to high neutrality measures depend on the $\mathcal{OD}$ bit position relative to the inverse quarter-round function, as previously discussed in section 3.5 of [AFK+08].

Table 4.4: The $\mathcal{OD}$ position with the best average neutral measure across for $R = 8$ using 2^{30} samples

Internal Round	$\mathcal{OD}$ Word	$\mathcal{OD}$ Bit	Average Neutral Measure
$4r$	1	13	$2^{-3.58}$
$4.25r$	3	13	$2^{-0.74}$
$4.5r$	3	13	$2^{-0.74}$
$4.75r$	3	13	$2^{-0.74}$
$5r$	0	18	$2^{-0.73}$
$5.25r$	11	13	$2^{-0.91}$
$5.5r$	11	13	$2^{-0.91}$
$5.75r$	11	13	$2^{-0.91}$

Table 4.5: Neutral measures γ_κ at $r = 4$, where p and q represent the word and bit positions of $\mathcal{OD}$ using 2^{30} samples.

R	Maximum			Minimum			Average	Median
	γ_κ	p	q	γ_κ	p	q		
6	$2^{-0.00144}$	2	1	$2^{-0.5451}$	9	7	$2^{-0.1907}$	$2^{-0.1473}$
7	$2^{-0.8143}$	0	18	$2^{-2.4426}$	14	7	$2^{-1.2708}$	$2^{-1.1496}$
8	$2^{-3.5850}$	1	13	$2^{-10.4926}$	2	9	$2^{-4.9755}$	$2^{-5.3066}$

Table 4.6: Neutral measures γ_κ at $r = 5$, where p and q represent the word and bit positions of $\mathcal{OD}$ using 2^{30} samples.

R	Maximum			Minimum			Average	Median
	γ_κ	p	q	γ_κ	p	q		
7	$2^{-0.1076}$	0	18	$2^{-0.4300}$	12	7	$2^{-0.2206}$	$2^{-0.1808}$
8	$2^{-0.7301}$	0	18	$2^{-2.7284}$	11	7	$2^{-1.2703}$	$2^{-1.2019}$
9	$2^{-3.6889}$	4	13	$2^{-10.39}$	12	28	$2^{-5.37}$	$2^{-5.9}$

To examine the upper limits of the inversed round function within our attack strategy, we conduct a detailed analysis of the neutral measures for each inversed round function. Tables 4.5 and 4.6 present the maximum, minimum, average, and

median values of neutral measures γ_κ corresponding to each target round R for $r = 4$ and 5 respectively.

These findings result from an extensive analysis of experimental data outlined in Section 4.1 Algorithm 3. As depicted in the tables, the maximum neutral measures for a specific inversed round function (e.g., $R - r = 3$ in this context) never surpass the minimum neutral measure observed in a smaller reverse round (e.g., $R - r = 2$) within the same internal round r. For example, when $R = 7$ and $R - r = 3$ (refer to Table 4.5), the maximum neutral measure is $\gamma_\kappa = 2^{-0.8143}$, while the minimum neutral measure is $\gamma_\kappa = 2^{-0.5451}$ for $R = 6$ and $R - r = 2$. It is evident that the minimum neutral measure for $R - r = 2$ exceeds the maximum neutral measure for $R - r = 3$, and vice versa. As discussed, this observation is influenced by the cumulative number of modular additions. Our experimental results are deemed reliable when the derived neutral measures γ_κ exceed $2^{-14.5}$, given our utilization of 2^{30} $\mathcal{ID}$ samples.

Table 4.5 shows that all neutral measure values are reliable when $R = 6, 7, 8$ and $R - r = 2, 3$, respectively. Additionally, the complexity of the attack for $R - r = 4$ is computed in Table 4.11. This complexity is lower than Salsa 20's security threshold, leading us to conclude that the maximum number of reverse rounds in our proposed attack is $4r$. Similarly, from Table 4.6, all neutral measure values for $R = 7$, 8, and $R - r = 2$, 3, and 4 are reliable.

4.2 Differential Cryptanalysis with PNB Approach

This section introduces a probabilistic neutral bit (PNB) based approach for conducting differential cryptanalysis on the reduced round of the Salsa 20 and ChaCha stream ciphers. Initially, we investigate the neutrality measures of all key bit positions across all possible $\mathcal{OD}$ bit positions using Algorithm 3. This algorithm aids in identifying the $\mathcal{OD}$ bit exhibiting the most favorable average neutral measure.

Once identified, we proceed to explore all potential $\mathcal{ID}$ positions seeking the best differential bias ε_d at the predetermined $\mathcal{OD}$ bit position with the optimal average neutral measure[2]. Subsequently, we employ the identified $\mathcal{ID}$ and $\mathcal{OD}$ pair to delineate the subset of PNBs. Following this, we compute the reverse bias ε_a for each threshold γ. Consequently, we estimate the complexity of our attack on the reduced round of Salsa 20 and ChaCha.

We present a differential attack based on the comprehensive analysis of PNBs on Salsa 8 and ChaCha 7.25 with a time complexity of $2^{241.62}$, $2^{254.011}$ and a data complexity of $2^{31.5}$, $2^{51.81}$, respectively. It's essential to note that the obtained neutral measure in Section 4.1 is solely used to select the $\mathcal{OD}$ bit with the best neutral measure;

[2]A search is conducted among 128 possible $\mathcal{ID}$ bit positions for the given $\mathcal{OD}$ position with the best average neutral measure

we do not factor it into the attack complexity estimation. The subsequent subsections provide a detailed description of the proposed cryptanalysis methodology.

4.2.1 The Attack on Salsa 8

In Section 4.1, we conducted a thorough analysis of the neutrality of 256 key bit positions across all possible 512 $\mathcal{OD}$ bit positions, selecting the $\mathcal{OD}$ bit position with the most favorable average neutral measure[3]. As evident from the data presented in Table 4.4, the $\mathcal{OD}$ position located at $(11, 13)$ within the 5.75 internal round yielded the best average neutral measure.

These $\mathcal{OD}$ bit positions, as listed in Table 4.4, were chosen as target $\mathcal{OD}$ positions for the Salsa 8 attack. To determine the corresponding $\mathcal{ID}$ bit positions exhibiting the best differential bias at the predefined $\mathcal{OD}$ positions outlined in Table 4.4, we scrutinized all possible 128 $\mathcal{ID}$ bit positions for each $\mathcal{OD}$ position listed in Table 4.4, selecting the $\mathcal{ID}, \mathcal{OD}$ position with the most favorable differential bias. This subsection presents the outcome of our quest for the $\mathcal{ID}$ position with the most favorable median bias ε_d^* across all specified $\mathcal{OD}$ positions in Table 4.4.

Table 4.7: $\mathcal{ID}$ with the optimal median bias ε_d^* using 2^{40} samples.

Intermediate round	$\mathcal{ID}$	$\mathcal{OD}$	ε_d^*
$4r$	7,0	1,13	$2^{-2.69}$
$4.25r$	8,11	3,13	$2^{-14.97}$
$4.5r$	8,11	3,13	$2^{-14.97}$
$4.75r$	8,11	3,13	$2^{-14.97}$
$5r$	7,22	0,18	$2^{-15.34}$
$5.25r$	6,3	0,18	$2^{-15.23}$
$5.5r$	9,28	11,13	$2^{-14.88}$
$5.75r$	9,28	11,13	$2^{-14.88}$

Table 4.8: $\mathcal{OD}$ positions with consistent bias in various internal rounds.

Intermediate round	$\mathcal{OD}$ Word	$\mathcal{OD}$ bit
$4.5r, 5.25r, 5.75r$	10	13
	11	13
	12	13
	13	13
	14	13
	15	13

As depicted in Table 4.7, the differential bias ε_d notably decreases after the 4^{th} round, directly impacted by the substantial increase in the cumulative number of

[3]We opted for the average neutral value as we computed the neutrality measure of all 256 key bits for each $\mathcal{OD}$ bit position.

modular additions. Additionally, the cumulative number of modular additions for specific words in the Salsa 20 state matrix remains consistent across the listed internal rounds, resulting in identical differential biases. Except for the $4r$ differential bias $\varepsilon_d^* = 0.154433$, none of the obtained biases in Table 4.7 could be consistently verified.

The biases can be trusted when they exceed 2^{-14}, considering using 2^{30} IV samples. However, the obtained biases fall short of this threshold, rendering them unreliable for further attack stages. To identify $\mathcal{ID}, \mathcal{OD}$ pairs with reliable median bias ε_d^*, we proceeded to search for differential biases across various $\mathcal{OD}$ positions for different internal rounds, as outlined in Table 4.8.

To determine the $\mathcal{ID}$ position with the best median bias ε_d^*, we conducted tests with 2^{30} IVs for each of 2^5 key trials. The findings are presented in Table 4.9. This marks the first instance of reporting new $\mathcal{ID}, \mathcal{OD}$ pairs for single-bit differential cryptanalysis of the Salsa 20 stream cipher.

To determine the number of Probabilistic Neutral Bits (PNBs), we then applied a threshold $0.1 \leq \gamma \leq 0.3$ to partition the set of key bits into two subsets: significant bits m and non-significant bits n. Given that the $\mathcal{ID}$ position $(8, 27)$ and $\mathcal{OD}$ position $(11, 13)$ exhibit the highest median bias, we focused our analysis on this pair to identify PNBs. The results are summarized in Table 4.10. Our emphasis was on the $4.75r$ internal rounds of Salsa 20/8, driven by the following considerations:

Table 4.9: $\mathcal{ID}$ positions with optimal median bias ε_d^* given $\mathcal{OD}$ positions using 2^{40} samples.

Internal Rounds	$\mathcal{ID}$	$\mathcal{OD}$	ε_d^*
$4.75r$	8,27	11,13	$2^{-3.94}$
$4.75r$	8,20	11,13	$2^{-6.73}$
$4.75r$	8,8	11,13	$2^{-7.28}$
$4.75r$	9,9	11,13	$2^{-7.73}$
$4.75r$	9,18	12,13	$2^{-5.24}$
$4.75r$	8,5	12,13	$2^{-9.98}$
$4.75r$	6,2	12,13	$2^{-11.084}$
$4.75r$	8, 24	12,13	$2^{-11.27}$

Table 4.10: The PNBs for $R = 8$

Threshold γ	$4.75r$
$\gamma = 0.1$	67
$\gamma = 0.2$	46
$\gamma = 0.25$	43
$\gamma = 0.27$	40
$\gamma = 0.3$	37

- Since the cumulative number of modular additions and average neutral measure remain consistent across quarter, half, and three-quarter internal rounds, we opted to compute the complexity of one internal round based on these metrics[4].
- Efficiently executing our attack on Salsa 20/9 poses challenges due to the absence of high average neutral measures for any $\mathcal{OD}$ bit. This issue arises from the substantial number of modular additions performed in both forward and backward rounds, resulting in a drastic drop in forward and backward bias.
- A comparison with the results from Table 4.4 reveals that $\mathcal{OD}$ bits yield superior neutral measures when using Algorithm 3 with $r = 5.75$ and $R - r = 2.25$. However, as valid biases could not be found in this internal round, we opted to attack Salsa 20/8 using the $\mathcal{OD}$ position $(11, 13)$ with $r = 4.75$ and $R-r = 3.25$, as indicated in Table 4.9.
- Furthermore, due to the lack of reliable median bias for $5r, 5.25r, 5.5r$, and $5.75r$, we chose to focus on the $4.75r$ internal round to attack Salsa 8. This decision was based on the observation that a higher number of internal rounds increases the number of PNBs, resulting in better neutral measures for key bit positions due to a lower number of inverse cumulative modular additions.

Next, we will estimate the complexity of the attack on Salsa 20/8 considering the $4.75r$ internal rounds. To accurately estimate the time and data complexities of our proposed attack on Salsa 20/8, the following steps need to be performed:

Table 4.11: The attack complexity on Salsa 20/8 when $r = 4$.

γ	n	$\lvert\varepsilon_d^*\rvert$	$\lvert\varepsilon_a^*\rvert$	α	Time	Data
0.1	40	0.154433	0.0004109	11	$2^{248.9}$	$2^{32.8}$
0.2	32	0.154433	0.00744507	12	$2^{248.6}$	$2^{24.54}$
0.3	26	0.154433	0.04336255	11	$2^{249.46}$	$2^{19.4}$

Table 4.12: The attack complexity Salsa 20/8 when $r = 4.75$

γ	n	$\lvert\varepsilon_d^*\rvert$	$\lvert\varepsilon_a^*\rvert$	α	Data	Time
0.3	37	0.0651375	0.01462765	16	$2^{25.9}$	$2^{244.36}$
0.2	**46**	**0.0651375**	**0.00173888**	**19**	$\mathbf{2^{31.5}}$	$\mathbf{2^{241.62}}$

Step 1. We recalculate neutral measures corresponding to the determined $\mathcal{ID}$-$\mathcal{OD}$ pair $(\Delta_i^0[j], \Delta_p^r[q])$, and divide the secret key bits into two subsets: the m-bit subset of significant key bits and the n-bit subset of non-significant key bits.

[4]Specifically, we chose to compute the complexity based on the $4.75r$ internal round out of the quarter, half, and three-quarter internal rounds.

Step 2. In the second phase, we run the probabilistic backward computation algorithm to derive the r-round differential biases ε_a in reverse for each threshold $0.1 < \gamma \leq 0.3$ from the extracted keystream, estimating the overall median bias as $|\varepsilon^*| \approx |\varepsilon_d^*| \cdot |\varepsilon_a^*|$. [3] This step is vital for our attack on Salsa 8.

Step 3. We execute the online phase algorithm and assess the time and data complexities needed to retrieve the unknown key.

To execute the above steps, for each of 2^8 key trials, we perform 2^{25} $\mathcal{ID}$ samples to compute the neutrality measure of key bits and obtain the subset of significant and non-significant key bits. The attack on Salsa 20/8 with $r = 4.75$ is summarized in Table 4.12. The identified PNBs are listed as follows: $0, 13, 14, 15, 16, 17, 31, 43, 44, 45, 70, 71, 76, 96, 97, 101, 113, 114, 115, 116, 117, 123, 124, 125, 126, 127, 135, 153, 154, 155, 156, 157, 158, 159, 170, 171, 172, 190, 229, 230, 231, 232, 233, 234, 247, 248$.

After selecting the subset of PNBs, it is necessary to validate the correctness of the PNB bits. We use Algorithm 2 to achieve this. As stated in [Mai16], if $\hat{\varepsilon}$ exhibits a low bias (close to a random occurrence), it confirms that the PNBs have been appropriately chosen. We can attack Salsa 20/8 with a time complexity of $2^{241.62}$ and a data complexity of $2^{31.5}$, given a threshold of $\gamma = 0.2$. Next, we focus on $\varepsilon_a^* = 0.00173888$ $(= 2^{-9.16})$ when $\gamma = 0.2$.

A total of $\frac{2}{\varepsilon_a^2} = 2^{19.33}$ $\mathcal{ID}$ samples is sufficient to distinguish the differential bias with a constant probability of success. Therefore, our experimental results are valid for $\gamma = 0.2$ as we have used 2^{25} $\mathcal{ID}$ samples for each of the 2^8 key trials. For $\gamma = 0.1$, we identified 67 PNBs with $\varepsilon_a^* = 0.00001713635 = (2^{-15.832})$, but our experiment was unable to consistently validate this.

Thus, the results are not reliable for $\gamma = 0.1$. In conclusion, we have demonstrated that a differential attack on Salsa 20/8 is feasible, based on the thorough analysis of probabilistic neutral bits, with a time complexity of $2^{241.62}$ and a data complexity of $2^{31.5}$. The current best key-recovery attack on Salsa 20 is the differential attack on Salsa 20/8, with time complexity of $2^{243.7}$, as proposed by Dey and Sarkar [DS17].

4.2.2 The Cryptanalysis of ChaCha 7.25

This section outlines the outcomes of our investigation into the ChaCha stream cipher. Specifically, we scrutinized the neutrality measure of key bits for ChaCha20/7, ChaCha20/7.25, and ChaCha20/7.5. However, regarding the attack's complexity, our attention was solely directed towards ChaCha20/7.25. We assessed the average neutrality measure of all key bits to all potential $\mathcal{OD}$ bit positions for internal rounds $r = 3$, $r = 3.5$, and $r = 4$.

[3] Based on [AFK+08], assuming reasonable independence, the equation $\varepsilon = \varepsilon_d \cdot \varepsilon_a$ applies.

The findings of our experiment are summarized in Table 4.1. Considering Table 4.1, $\mathcal{OD}$ bit 0 consistently exhibits the best neutral measure across all internal rounds. This measure is influenced by factors such as the cumulative number of modular subtractions, the input word position in the inverse quarter round of ChaCha, and the structure of the ChaCha quarter round function. We visualized the distribution of neutrality measures for ChaCha20/7 across internal rounds $r = 3$, $r = 3.5$, and $r = 4$ in Figure 4.4, where the X-axis represents the neutrality measure of 256 key bits and the Y-axis represents the $\mathcal{OD}$ bit position. The results from Table 4.1 and Figure 4.5 indicate that the higher number of internal rounds influences the neutrality measure.

Specifically, the neutrality measure for internal round $r = 3$ is lower than $r = 3.5$, which is directly affected by the cumulative number of modular subtractions executed for different internal rounds. We targeted the higher number of ChaCha rounds with internal round $r = 3.5$. While we could have selected internal round $r = 4$ to target $R = 7$, 7.25, or 7.75, the forward bias ε_d would have significantly decreased for $r = 4$. As there has been little focus on evaluating the security of ChaCha7.25 rounds, we chose $r = 3.5$ to attack ChaCha7.25.

We analyze the neutrality measures of ChaCha7.25, ChaCha7.5, and ChaCha7.75, considering the internal round $r = 3.5$. We employed Algorithm 3 to assess the neutrality measures of different target rounds. We have decided to target ChaCha7.25 with an internal round of $r = 3.5$.

Table 4.13: $\mathcal{OD}$ bit with best neutral measure using 2^{30} samples.

Target Round	$\mathcal{OD}$ Word	$\mathcal{OD}$ Bit	Average Neutral Measure
$R = 7.25$	7	0	$2^{-1.82}$
$R = 7.5$	4	0	$2^{-2.72}$
$R = 7.75$	11	0	$2^{-3.74}$

Table 4.14: The differential bias using 2^{40} samples.

$\mathcal{ID}$ Position	$\mathcal{OD}$ Position	Bias ε_d
$\Delta^{(0)}_{15[6]}$	$\Delta^{(3.5)}_{0[0]}$	$2^{-11.07}$
$\Delta^{(0)}_{12[6]}$	$\Delta^{(3.5)}_{1[0]}$	$2^{-11.23}$
$\Delta^{(0)}_{13[6]}$	$\Delta^{(3.5)}_{2[0]}$	$2^{-11.04}$
$\Delta^{(0)}_{14[6]}$	$\Delta^{(3.5)}_{3[0]}$	$2^{-11.04}$

In our attack on the 7.25-round ChaCha, we selected the $\mathcal{OD}$ position $\Delta^{(3.5)}_{7,0}$ from Table 4.13 and examined all possible 128-bit $\mathcal{ID}$ positions to find the one with the optimal differential bias ε_d. To determine the $\mathcal{ID}$ position with the best ε_d at the chosen $\mathcal{OD}$ position, we conducted trials with a total complexity of 2^{34}, which involved 2^6 key trials and 2^{28} IV samples. A bias ε_d greater than $2^{-15.5}$ is considered reliable.

Table 4.15: The attack complexity on ChaCha7.25 for $r = 3.5$.

γ	$\mathcal{ID}, \mathcal{OD}$	n	ε_d	ε_a	α	Time	Data
0.25	$\Delta^{(0)}_{15[6]}\|\Delta^{(3.5)}_{0[0]}$	54	0.000463	0.000151	5	$2^{254.19}$	$2^{52.026}$
0.25	$\Delta^{(0)}_{12[6]}\|\Delta^{(3.5)}_{1[0]}$	54	0.000414	0.000156	5	$2^{254.38}$	$2^{52.24}$
0.25	$\mathbf{\Delta^{(0)}_{13[6]}\|\Delta^{(3.5)}_{2[0]}}$	**54**	**0.000474**	**0.000158**	**5**	$\mathbf{2^{254.011}}$	$\mathbf{2^{51.81}}$
0.27	$\Delta^{(0)}_{14[6]}\|\Delta^{(3.5)}_{3[0]}$	50	0.000472	0.000413	4	$2^{255.12}$	$2^{48.95}$

Table 4.16: Subset of PNBs n for various thresholds γ in $r = 3.5$ rounds.

Threshold γ	$\mathcal{ID}$, $\mathcal{OD}$ Pair	Number of PNBs
$\gamma = 0.25$	$\Delta^{(0)}_{15[6]}\|\Delta^{(3.5)}_{0[0]}$	54
$\gamma = 0.25$	$\Delta^{(0)}_{12[6]}\|\Delta^{(3.5)}_{1[0]}$	54
$\gamma = 0.25$	$\Delta^{(0)}_{13[6]}\|\Delta^{(3.5)}_{2[0]}$	54
$\gamma = 0.27$	$\Delta^{(0)}_{14[6]}\|\Delta^{(3.5)}_{3[0]}$	50

From our analysis, we identified the maximum forward bias ε_d at $\Delta^{(0)}_{12[18]}|\Delta^{(3.5)}_{7[0]} = 0.000019$. Since this value, $0.000019 = 2^{-15.68}$, could not be confirmed with our sample size, we explored the differential bias ε_d at $\Delta^{(3.5)}_{0[0]}$, $\Delta^{(3.5)}_{1[0]}$, $\Delta^{(3.5)}_{2[0]}$, and $\Delta^{(3.5)}_{3[0]}$ $\mathcal{OD}$ positions. The findings are summarized in Table 4.14.

All the biases presented in Table 4.14 are valid, as they were confirmed by the number of samples used in the experiment. Consequently, we used the pairs with the strongest differential bias ε_d from Table 4.14 to determine the sets of significant key bits m and non-significant key bits n.

Complexity Estimation of Attack on ChaCha7.25 To estimate the attack complexity on ChaCha7.25 rounds, we repeat the procedures outlined in Section 4.2.2 using the $\mathcal{ID}, \mathcal{OD}$ positions detailed in Table 4.14, taking into account the structure of the ChaCha stream cipher. To identify the subsets m and n, we conducted experiments with a total complexity of 2^{36}.

Based on the threshold value γ, the elements' numbers differ in subsets m and n, as summarized in Table 4.16. Notably, key bits corresponding to $\gamma = 0.25$ and $\mathcal{ID}, \mathcal{OD}$ position $\Delta^{(0)}_{13[6]}|\Delta^{(3.5)}_{2[0]}$ are identified as *probabilistic neutral bits*, including [66, 67, 74, 77, 78 , 83, 84, 90, 91, 95, 104, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 135, 155, 156, 157, 158, 159, 160, 168, 169, 191, 192, 193, 194, 199, 200, 207, 208, 211, 212, 219, 220, 221, 222 , 223, 224, 225, 226, 227, 244, 245, 246, 247, 255]. Table 4.15 summarizes our attack on ChaCha7.25/20 for the internal round $r = 3.5$, with the optimal $\mathcal{ID}, \mathcal{OD}$ pair $\Delta^{(0)}_{13[6]}|\Delta^{3.5}_{3[0]}$, threshold $\gamma = 0.27$, and $n = 54$. The attack's estimated time and data complexity were $2^{254.01}$ and $2^{51.81}$, respectively. Miyashita [MIM22] employed a similar attack methodology on ChaCha 7.25, utilizing the following parameters.

The input difference ($\mathcal{ID}$) and output difference ($\mathcal{OD}$) were chosen as $\Delta^{(0)}_{15[6]}|\Delta^{(3.5)}_{0[0]}$.

The forward bias was 0.000469, and the backward bias was 0.000564. To obtain the PNBs, the threshold was set to $\gamma = 0.3$, resulting in a total of 49 PNBs: {2, 3, 10, 13, 14, 19, 20, 26, 27, 31, 40, 44, 45, 46, 51, 59, 60, 61, 62, 63, 128, 129, 130, 135, 136, 143, 144, 147, 148, 155, 156, 157, 158, 159, 160, 161, 162, 180, 181, 182, 191, 219, 220, 221, 222, 223, 224, 232, 255}. With these parameters, the attack exhibits a time complexity of $2^{255.62}$ and a data complexity of $2^{48.36}$.

Chapter 5

Higher-Order Differential-Linear Cryptanalysis of ChaCha

This chapter explores the application of higher-order differential-linear cryptanalysis to the ChaCha. The research extends the linear approximation from the 4th round to the 6th round using higher-order differential analysis. We investigate higher-order differential-linear cryptanalysis on a reduced number of rounds of the ChaCha stream cipher employing second-order differentials for the differential component.

This study reveals new higher-order differential biases for ChaCha 3, ChaCha 3.5, and ChaCha 4. Moreover, novel $\mathcal{ID}, \mathcal{OD}$ positions for performing higher-order differential cryptanalysis on ChaCha are identified. Based on our recent journal publication [1], we computed the distinguisher of ChaCha 5, ChaCha 5.5, and ChaCha 6. In addition, this chapter introduces distinguishing attacks for ChaCha 5, ChaCha 5.5, and ChaCha 6 that incorporate the linear approximation of the final modular addition used in key generation.

5.1 Attack Points Selection

This section presents new $\mathcal{ID}, \mathcal{OD}$ positions associated with higher-order differentials. Our methodology for selecting the $\mathcal{OD}$ positions involves two interconnected steps. Firstly, we aimed to narrow down the potential $\mathcal{OD}$ positions from an initial set of 512 to just two choices (specifically, word B and D). This was achieved by identifying $\mathcal{OD}$ positions based on the ChaCha quarter-round function structure and linear approximation properties. We concentrated on $\mathcal{OD}$ positions that generate a greater number of least significant bits when the bit position is set to zero ($i = 0$).

Next, we validated the chosen positions using Algorithm 4 to perform an extensive search for $\mathcal{OD}$ positions that produce the highest count of probabilistic neutral bits

[1] The journal paper was published in IEEE Access and it foundation of this chapter.

(PNBs) concerning the lth-order differential. For more details on PNBs, please see [AFK+08]. This algorithm was applied to the 2nd and 3rd-order differentials, targeting the 7th, 7.25th, and 7.5th rounds, including both internal and reverse rounds. The outcomes of these experiments are summarized in Tables 5.1 and 5.2.

Additionally, this higher-order differential-linear cryptanalysis allows the attacker to focus on more forward rounds, thereby addressing the challenge of having fewer reverse rounds with this technique.

Algorithm 4 Computing Neutrality Measure of Key bits concerning the $\mathcal{OD}$ position

Input: Random(X) and associate states (X_1, X_2, X_3) and counter $T = 0$

Output: The $\mathcal{OD}$ position where the key bits generate the best average neutral measure given higher-order differentials.

1. Randomly select key values $k = (k_4, \cdots, k_{11})$.
2. Define $\mathcal{ID}$ $\Delta_{12}^{(0)}[15], \Delta_{13}^{(0)}[20]$ and find a new initial states $X_1^{(0)} = X^{(0)} \oplus \Delta_{12}^{(0)}[15]$, $X_2^{(0)} = X^{(0)} \oplus \Delta_{13}^{(0)}[20]$ and $X_3^{(0)} = X^{(0)} \oplus \Delta_{12}^{(0)}[15] \oplus \Delta_{13}^{(0)}[20]$.
3. From the first states $(X^{(0)}, X_1^{(0)}, X_2^{(0)}, X_3^{(0)})$, calculate the states after $r = 4$ rounds $(X^{(4)}, X_1^{(4)}, X_2^{(4)}, X_3^{(4)})$ and the last states $(X^{(7)}, X_1^{(7)}, X_2^{(7)}, X_3^{(7)})$.
4. From $(X^{(4)}, X_1^{(4)}, X_2^{(4)}, X_3^{(4)})$ obtain output differentials $\mathcal{OD}$ $\Delta^{(4)}4[0] = X^{(4)}4[0] \oplus X_1^{(4)}4[0] \oplus X_2^{(4)}4[0] \oplus X_3^{(4)}4[0])$.
5. From the final states $(X^{(7)}, X_1^{(7)}, X_2^{(7)}, X_3^{(7)})$ obtain the key-stream $Z = X^{(0)} + X^{(7)}$, $Z_1 = X_1^{(0)} + X_1^{(7)}$, $Z_2 = X_2^{(0)} + X_2^{(7)}$, and $Z_3 = X_3^{(0)} + X_3^{(R)}$.
6. Flip a key bit κ ($\kappa \in \{0, \ldots, 255\}$) and obtain new initial states $\overline{X}^{(0)}$, $\overline{X}_1^{(0)}$, $\overline{X}_2^{(0)}$, $\overline{X}_3^{(0)}$ from initial states $(X^{(0)}, X_1^{(0)}, X_2^{(0)}, X_3^{(0)})$.
7. Compute the states $(Y^{(4)}, Y_1^{(4)}, Y_2^{(4)}, Y_3^{(4)})$ with $Z - \overline{X}^{(0)}, Z_1 - \overline{X}_1^{(0)}, Z_2 - \overline{X}_2^{(0)}, Z_3 - \overline{X}_3^{(0)}$ as inputs to the reverse function.
8. Obtain the output differentials $\Gamma^{(4)}4[0] = Y^{(4)}4[0] \oplus Y_1^{(4)}4[0] \oplus Y_2^{(4)}4[0] \oplus Y_3^{(4)}4[0]$ for all possible choices of 4 and 0.
9. If $\Delta_4^{(4)}[0] = \Gamma_4^{(4)}[0]$ increment the T.
10. To determine the probability of each key bit relative to $\Delta_4^{(4)}[0]$, divide the total T by the number of key trials and $\mathcal{ID}$ samples.

Table 5.1: Optimal Average Neutral Measure for Second-Order Differentials using 2^{30} samples.

Rounds	Internal Rounds	$\mathcal{ID}$	$\mathcal{OD}$	Highest Average NM
7	3	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}$	$\Delta X^{(3)}_{4,[0]}$	$2^{-7.477}$
7	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}$	$\Delta X^{(3.5)}_{8,[0]}$	$2^{-7.049}$
7	4	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}$	$\Delta X^{(4)}_{4,[0]}$	$2^{-6.108}$
7.25	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}$	$\Delta X^{(3.5)}_{6,[19]}$	$2^{-7.099}$
7.5	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}$	$\Delta X^{(3.5)}_{6,[6]}$	$2^{-7.99}$
7.75	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}$	$\Delta X^{(3.5)}_{8,[0]}$	$2^{-9.07}$

Table 5.2: Optimal Average Neutral Measure for Third-Order Differentials using 2^{30} samples.

Rounds	Internal Rounds	$\mathcal{ID}$	$\mathcal{OD}$	Highest Average NM
7	3	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}, X^{(0)}_{14,[21]}$	$\Delta X^{(3)}_{4,[0]}$	$2^{-7.57}$
7	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}, X^{(0)}_{14,[21]}$	$\Delta X^{(3.5)}_{6,[18]}$	$2^{-7.22}$
7	4	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}, X^{(0)}_{14,[21]}$	$\Delta X^{(4)}_{0,[0]}$	$2^{-6.38}$
7.25	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}, X^{(0)}_{14,[21]}$	$\Delta X^{(3.5)}_{6,[18]}$	$2^{-7.23}$
7.5	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}, X^{(0)}_{14,[21]}$	$\Delta X^{(3.5)}_{6,[6]}$	$2^{-7.99}$
7.75	3.5	$\Delta X^{(0)}_{12,[15]}, \Delta X^{(0)}_{13,[20]}, X^{(0)}_{14,[21]}$	$\Delta X^{(3.5)}_{4,[7]}$	$2^{-13.28}$

Table 5.3: Second-order Bias of ChaCha using 2^{40} samples

$\mathcal{ID}$	$\mathcal{OD}$	ε_d	$\lvert\varepsilon_d *\rvert$
$\Delta X^{(0)}_{12,[0]}, \Delta X^{(0)}_{13,[0]}$	$\Delta X^{(4)}_{4,[0]}$	0.000096	$2^{-15.53}$
$\Delta X^{(0)}_{12,[0]}, \Delta X^{(0)}_{13,[0]}$	$\Delta X^{(3.5)}_{8,[0]}$	0.000103	$2^{-15.53}$
$\Delta X^{(0)}_{12,[0]}, \Delta X^{(0)}_{13,[0]}$	$\Delta X^{(3)}_{4,[0]}$	0.00021	$2^{-15.53}$

Table 5.4: Third-order bias of ChaCha using 2^{40} samples

$\mathcal{ID}$	$\mathcal{OD}$	$\lvert\varepsilon^*_d\rvert$
$\Delta X^{(0)}_{12,[0]}, \Delta X^{(0)}_{13,[0]}, \Delta X^{(0)}_{13,[31]}$	$\Delta X^{(4)}_{0,[0]}$	$2^{-15.53}$
$\Delta X^{(0)}_{12,[0]}, \Delta X^{(0)}_{13,[0]}, \Delta X^{(0)}_{13,[31]}$	$\Delta X^{(3.5)}_{6,[18]}$	$2^{-15.53}$
$\Delta X^{(0)}_{12,[0]}, \Delta X^{(0)}_{13,[0]}, \Delta X^{(0)}_{13,[31]}$	$\Delta X^{(3)}_{4,[0]}$	$2^{-15.53}$

Given the nature of differential-linear attacks on ChaCha, our primary focus is on $\mathcal{OD}$ positions at the 0th bit, and we, therefore, exclude all other positions where $i > 0$. From the data in Tables 5.1 and 5.2, we chose the $\mathcal{OD}$ position $\Delta X^{(4)}_{4,[0]}$ to target ChaCha 6 due to its superior neutrality measure compared to other positions.

Identifying $\mathcal{ID}$ positions for higher-order differentials is a challenging task, and an exhaustive search is not an efficient solution. For the 2nd-order differential, selecting

two $\mathcal{ID}$s at a time from a set of 128 possible $\mathcal{ID}$s yields 8128 possible positions. Given that a brute-force search over these 8128 bit combinations is impractical, we employed the Hamming Weight method to choose pairs of two bits in the context of 2nd-order differentials, which allowed us to examine the forward differential ε_d.

Table 5.5: Hamming weight of ChaCha matrix

Rounds	HM	Percentage of Changes
1	10	2.14 %
2	63	12.5%
3	204	40.03%

We sought to determine the Hamming weight of the ChaCha matrix with the $\mathcal{ID}$ in the initial state after the first, second, and third rounds. Notably, the ChaCha quarter-round function (Equation 2.3) distributes the difference evenly, regardless of the $\mathcal{ID}$ position.

This behavior differs from that of the Salsa 20 stream cipher, where the Hamming weight of the Salsa 20 matrix after certain rounds was influenced by the $\mathcal{ID}$ position. On average, the change in the Hamming Weight of the ChaCha matrix after each round was found to be 18.21%.

As indicated in Table 5.5, the $\mathcal{ID}$ position does not have a significant impact on the Hamming Weight, which might not notably affect the forward differential. Therefore, we selected $\Delta X^{(0)}_{12,[0]} \oplus \Delta X^{(0)}_{13,[0]}$ as the input differential positions for calculating the second-order differential, and $\Delta X^{(0)}_{12,[0]} \oplus \Delta X^{(0)}_{13,[0]} \oplus \Delta X^{(0)}_{13,[31]}$ for exploring the third-order differential bias in ChaCha. We conducted an experiment[2] with a complexity of 2^{40} to look for the 2nd and third-order differentials. The findings are outlined as follows. A maximum of 2^{32} random values were required to distinguish the biases presented in Tables 5.3 and 5.4. These results show that while the highest bias, denoted by ε_d, differs for every location, the absolute median bias, $|\varepsilon^*_d|$, remains consistent for both second- and third-order biases. We also analyzed the biases at $\Delta X^{(3.5)}_{6[0]}$, $\Delta X^{(3.5)}_{8[0]}$, and $\Delta X^{(3.5)}_{11[0]}$ for the second-order differential, and at $\Delta X^{(3.5)}_{15[0]}$, $\Delta X^{(3.5)}_{6[0]}$, $\Delta X^{(3.5)}_{11[0]}$, and $\Delta X^{(3.5)}_{9[0]}$ for the third-order differentials. The findings align with those reported in Tables 5.3 and 5.4. The complexity of the attack, given the second-order and third-order differential cryptanalysis, is $2^{31.07}$ for a 3.5 round of ChaCha. Therefore, linear cryptanalysis is necessary to strengthen the attack on the higher rounds of ChaCha.

[2]For our experiment, we employed the Maximum Length (M-Sequence) random number generator. The experiment was executed on an Intel(R) Xeon(R) CPU E7-4830 v4 @ 2.00GHz.

5.2 Applying Linear Cryptanalysis

Enhancing higher-order differential cryptanalysis with linear cryptanalysis depends on the complementary strengths of these two methods. The following explains how this combination works:

- Linear cryptanalysis helps to detect linear patterns in a cipher, which can be leveraged to simplify the process of breaking the cipher [Mat93], [LH94].
- Higher-order differential cryptanalysis includes complex calculations due to the numerous input differences that must be analyzed (based on the differential order $\mathcal{L}$). Attacking a larger number of rounds with higher-order differential cryptanalysis alone can be computationally prohibitive. Incorporating linear approximations can enhance the number of attack rounds and significantly reduce the complexity of these calculations. For practical insight, see Algorithm 4.
- The effectiveness of higher-order differential cryptanalysis relies on the likelihood of specific differential patterns. Linear cryptanalysis can boost these probabilities by pinpointing and exploiting linear relationships between the plaintext, ciphertext, and key bits [LH94].
- While higher-order differential cryptanalysis might be less effective against ciphers with high algebraic degrees, linear cryptanalysis can examine the linear aspects of these ciphers. This allows for a more in-depth cipher analysis, potentially resulting in a more efficient attack.
- Some ciphers, such as the ChaCha stream cipher, are resistant to differential cryptanalysis and its variants. However, they can be vulnerable when a combination of techniques is employed. By merging linear cryptanalysis with higher-order differentials, cryptanalysts can explore a broader spectrum of weaknesses in the cipher.

Given these considerations, our approach involved executing a higher-order differential-linear attack on the ChaCha cipher. Subsection 5.2 provides a detailed explanation of how the linear component is integrated into our proposed attack strategy.

Linear Approximations

Our attack was developed based on the following analysis: Given the current state of research and the absence of higher-order differential cryptanalysis studies on ChaCha, we sought to utilize the advantages of higher-order differentials in conjunction with linear cryptanalysis to attack reduced rounds of the ChaCha stream cipher.

As indicated in Tables 5.3 and 5.4, the higher-order differential bias in ChaCha consistently resulted in the same bias across different internal rounds (namely, $r = 3, 3.5$, and 4). This vulnerability is inherent to the structure of ChaCha itself. Prior studies have focused on differential analysis for 3.5 internal rounds of ChaCha and have reported linear approximations for only 2.5 rounds. However, no research has identified a differential-linear bias for 4 rounds. In differential attacks, increasing the number of internal rounds enables us to target a larger number of rounds, thereby reducing the overall complexity of the attack. Furthermore, as stated in Lemma 2.1, increasing the number of linear approximations reduces the linear correlation.

Building on this understanding, we used the 4th round differential bias in ChaCha alongside 2 rounds of linear approximation. For this analysis, we chose the $\mathcal{OD}$ position $\Delta x^{(4)}_{4,0}$, which corresponds to the word B in the ChaCha matrix, and validated it using Algorithm 4. To extend $\Delta x^{(4)}_{4,0}$ to the 5th round and obtain a linear approximation with a probability of 1, we applied Lemma 3.1.

Lemma 5.1. *The following linear approximation from 4th to 6th rounds of ChaCha holds with probability* $\frac{1}{2}(1 + \frac{1}{2^2})$

$$\begin{aligned}\Delta x^{(4)}_{4[0]} = \Delta x^{(6)}_{1[0]} \oplus \Delta x^{(6)}_{2[0,11,12]} \oplus \Delta x^{(6)}_{4[6]} \oplus \Delta x^{(6)}_{6[7]} \oplus \Delta x^{(6)}_{8[0,12]} \oplus \Delta x^{(6)}_{9[19,31]} \oplus \Delta x^{(6)}_{11[0]} \oplus \Delta x^{(6)}_{12[8]} \\ \oplus \quad \Delta x^{(6)}_{13[0,8,11,12,19,20]} \oplus \Delta x^{(6)}_{14[18,19]} \ \ W.P.\ \frac{1}{2}\left(1 + \frac{1}{2^2}\right). \end{aligned} \tag{5.1}$$

Proof: First, we extend from 4th round to 5th round with probability 1. For this purpose, we use the Lemma 3.1 and the approximation for the word B given the position of $\mathcal{OD}$ $\Delta x^{(4)}_{4[0]}$:

$$\begin{aligned}\Delta x^{(m-1)}_{b[i]} &= \Delta x^{(m)}_{b[19]} \oplus \Delta x^{(m)}_{c[12]} \oplus \Delta x^{(m)}_{d[0]} \oplus \Delta x^{(m)}_{c[0]} \ \ W.\ P.\ 1, \text{ and} \\ \Delta x^{(4)}_{4[0]} &= \Delta x^{(5)}_{4[19]} \oplus \Delta x^{(5)}_{8[0,12]} \oplus \Delta x^{(5)}_{12[0]} \ \ W.\ P.\ 1.\end{aligned}$$

Since the linear extension has a probability of 1 in this scenario, $\varepsilon^*_d \cdot \varepsilon^2_l = \varepsilon^*_d$ and $\varepsilon^*_d \cdot \varepsilon^2_l = 2^{-15.5}$. Next, we aim to extend the linear approximation from the 5th round to the 5.5th round. To achieve this, we employ Lemma 3.2. With the linear extension of the 5th round, all expressions can be extended with a probability of 1, except for $\Delta x^{(5)}_{8[12]}$, which is located in position C of the input vector to the ChaCha quarter-round function. Consequently, applying Lemma 3.3 to extend $\Delta x^{(5)}_{8[12]}$ to $\Delta x^{(5.5)}_{8[12]}$ with a probability of 1/2 and use Lemma 3.2 to extend the other expressions to half a round with a probability of 1.

$$\Delta x^{(5)}_{4[19]} = \Delta x^{(5.5)}_{4[31]} \oplus \Delta x^{(5.5)}_{9[19]} \ \ W.P\ 1 \tag{5.2}$$

$$\Delta x^{(5)}_{8[12]} = \Delta x^{(5.5)}_{8[12]} \oplus \Delta x^{(5.5)}_{13[12]} \oplus \Delta x^{(5.5)}_{13[11]} \ W.P\ \frac{1}{2}.$$
$$\Delta x^{(5)}_{12[0]} = \Delta x^{(5.5)}_{12[0]} \oplus \Delta x^{(5.5)}_{1[0]} \ W.P\ 1, \text{ and} \tag{5.3}$$
$$\Delta x^{(5)}_{8[0]} = \Delta x^{(5.5)}_{8[0]} \oplus \Delta x^{(5.5)}_{13[0]} \ W.P\ 1. \tag{5.4}$$

Consequently,

$$\Delta x^{(4)}_{4[0]} = \Delta x^{(5.5)}_{1[0]} \oplus \Delta x^{(5.5)}_{4[31]} \Delta x^{(5.5)}_{8[0,12]} \oplus \Delta x^{(5.5)}_{9[19]} \oplus \Delta x^{(5.5)}_{12[0]} \oplus \Delta x^{(5.5)}_{13[11,12,0]} \ W.P.\ \frac{1}{2}.$$

Consequently, we can calculate the distinguisher and differential-linear bias for ChaCha 5.5 as $\varepsilon_d^* \cdot \varepsilon_l^2 = 2^{-18.53}$*. The attack complexity for ChaCha 5.5 is outlined in Table 5.7. The higher-order differential biases for ChaCha 5 are presented in Table 5.3. Our next objective is to extend the linear approximation from ChaCha 5.5 to ChaCha 6 rounds. To accomplish this, we apply Lemma 3.2 to word positions in B and D, which can be extended with a probability of 1. For word positions in A and C, we use Lemma 3.3 with a probability of* $\frac{1}{2}(1 + \frac{1}{2})$*.*

$$\Delta x^{(5.5)}_{4[31]} = \Delta x^{(6)}_{4[6]} \oplus \Delta x^{(6)}_{9[31]} \ W.P\ 1. \tag{5.5}$$
$$\Delta x^{(5.5)}_{9[19]} = \Delta x^{(6)}_{9[19]} \oplus \Delta x^{(6)}_{14[19]} \oplus \Delta x^{(6)}_{14[18]} \ W.P.\ \frac{1}{2}\left(1 + \frac{1}{2}\right). \tag{5.6}$$
$$\Delta x^{(5.5)}_{8[12]} = \Delta x^{(6)}_{8[12]} \oplus \Delta x^{(6)}_{13[12]} \oplus \Delta x^{(6)}_{13[11]} \ W.P.\ \frac{1}{2}\left(1 + \frac{1}{2}\right).$$
$$\Delta x^{(5.5)}_{13[12]} = \Delta x^{(6)}_{2[12]} \oplus \Delta x^{(6)}_{13[20]} \ W.P\ 1. \tag{5.7}$$
$$\Delta x^{(5.5)}_{12[0]} = \Delta x^{(6)}_{1[0]} \oplus \Delta x^{(6)}_{12[8]} \ W.P\ 1. \tag{5.8}$$
$$\Delta x^{(5.5)}_{1[0]} = \Delta x^{(6)}_{1[0]} \oplus \Delta x^{(6)}_{6[7]} \oplus \Delta x^{(6)}_{11[0]} \ W.P\ 1. \tag{5.9}$$
$$\Delta x^{(5.5)}_{8[0]} = \Delta x^{(6)}_{8[0]} \oplus \Delta x^{(6)}_{13[0]} \ W.P\ 1. \tag{5.10}$$
$$\Delta x^{(5.5)}_{13[0]} = \Delta x^{(6)}_{2[0]} \oplus \Delta x^{(6)}_{13[8]} \ W.P\ 1. \tag{5.11}$$
$$\Delta x^{(5.5)}_{13[11]} = \Delta x^{(6)}_{2[11]} \oplus \Delta x^{(6)}_{13[19]} \ W.P\ 1. \tag{5.12}$$

As a consequence, the term $\Delta x^{(6)}_{1[0]}$ *is eliminated, resulting in the following linear approximation with a probability of* $1/2(1+1/2^2)$ *from ChaCha 4 to ChaCha 6. To the best of our knowledge, this is the first and most optimal linear approximation reported for two rounds to date. The attack complexity is outlined in Table 5.7.*

$$\Delta x^{(4)}_{4[0]} = \Delta x^{(6)}_{2[0,11,12]} \oplus \Delta x^{(6)}_{4[6]} \oplus \Delta x^{(6)}_{6[7]} \Delta x^{(6)}_{8[0,12]} \oplus \Delta x^{(6)}_{9[19,31]} \oplus \Delta x^{(6)}_{11[0]} \oplus \oplus \Delta x^{(6)}_{12[8]} \oplus$$
$$\Delta x^{(6)}_{13[0,8,11,12,19,20]} \oplus \Delta x^{(6)}_{14[18,19]} \ W.P.\ \frac{1}{2}\left(1 + \frac{1}{2^2}\right). \tag{5.13}$$

5.3 The Attack Complexity of Higher-Order Differential-Linear Cryptanalysis

To address the complexity of higher-order differential-linear attack, we will break down the problem and analyze the effect of changing the complexity from first-order differential-linear cryptanalysis $O\left(\frac{1}{pq^2}\right)$ to second-order differential-linear cryptanalysis $O\left(\frac{1}{pq^4}\right)$ on the given distinguisher complexity $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^4}\right)$.

The original complexity of the differential-linear distinguisher is given by $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^4}\right)$. Generally, for the first-order differential-linear cryptanalysis we require $O\left(\frac{1}{pq^2}\right)$ samples when distinguishing between two events with probabilities p and $p(1+q)$. If we change the requirement from the first-order differential-linear $O\left(\frac{1}{pq^2}\right)$ to second-order differential-linear $O\left(\frac{1}{pq^4}\right)$, we need to analyze how this affects the $O\left(\frac{1}{pq^2}\right)$.

The second-order differential-linear samples requirement is $O\left(\frac{1}{pq^4}\right)$. In first-order differential-linear cryptanalysis, the complexity of the distinguisher is proportional to the inverse of the product of the squares of ϵ_d and the fourth power of ϵ_L $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^4}\right)$. To match the second-order differential-linear sample requirement $O\left(\frac{1}{pq^4}\right)$, we infer the following:

- The term $\frac{1}{pq^4}$ implies a higher order dependence on q, which translates to ϵ_L in our distinguisher complexity.
- The relationship should adapt to a higher degree polynomial in ϵ_L.

Thus, the new complexity should be adjusted to reflect a stronger dependence on ϵ_L. If the first-order differential-linear complexity is $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^4}\right)$, increasing the power of ϵ_L from 4 to 8 would match the new requirement $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^8}\right)$. Changing the requirement from $O\left(\frac{1}{pq^2}\right)$ to $O\left(\frac{1}{pq^4}\right)$ affects the original complexity $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^4}\right)$ by increasing the exponent of ϵ_L from 4 to 8. Thus, the new complexity would be $O\left(\frac{1}{\epsilon_d^2 \cdot \epsilon_L^8}\right)$. This adjustment accounts for the higher degree of dependence on the parameter associated with the bias in the distinguisher.

5.3.1 The Distinguisher of ChaCha

Given the obtained linear probability and the second-order differential bias, the differential-linear distinguisher of ChaCha 5, ChaCha 5.5, and ChaCha 6 are reported in Table 5.6. The linear approximation from ChaCha 4 to ChaCha 6 is discussed earlier. Expanding the linear approximation to the 7th round challenges handling significant bits.

When i is assigned a value of 0 for ChaCha, the least significant bits can be linearly approximated with a probability of 1. As a result, the major impact on computational

effort stems from the variables related to the significant bits. The attack complexity is determined by tracking how often significant bit variables appear and noting their frequency (Variable Type, Count of Significant Bit Occurrences).

For the linear approximation between the 4th and 6th rounds, the number of significant variables is $2, 2, 3, 8$ for words A, B, C, D respectively. These are represented as $(x_a, 2), (x_b, 2), (x_c, 3), (x_d, 8)$. The Lemma 3.4 which defines the probability of each word linear approximation when $i > 1$ and Lemma 5.1, the linear correlation can be computed as $\varepsilon_L = \frac{1}{2^{2+2\cdot4+2\cdot1+3\cdot2+8\cdot1}}$. This leads to a 7 rounds distinguisher with a complexity of $2^{239.21}$.

Table 5.6: The distinguisher complexity.

Target	$\mathcal{OD}$	$\lvert\varepsilon_d^*\rvert$	$\varepsilon_d^2 \times \varepsilon_l^8$	Distinguisher Complexity
5	$\Delta X^{(4)}_{4,[0]}$	$2^{-15.53}$	$2^{-31.21}$	$2^{31.21}$
5.5	$\Delta X^{(4)}_{4,[0]}$	$2^{-15.53}$	$2^{-39.21}$	$2^{39.21}$
6	$\Delta X^{(4)}_{4,[0]}$	$2^{-15.53}$	$2^{-47.21}$	$2^{47.21}$

5.3.2 The Distinguishing Attack Complexity

To determine the final distinguishing attack on ChaCha 5, ChaCha 5.5, and ChaCha 6, it is necessary to calculate the probability of the final modular addition used to generate the ChaCha keystream $Z = X^0 + X^R$. This probability is combined with the likelihood of the linear approximation of the ChaCha QR function to obtain the final complexity. The approach utilizes Lemma 3.8, as introduced in, [BBC+22]. Based on the ChaCha structure, we approximate the final modular addition for keywords in the ChaCha matrix (i.e., $X_4 \cdots X_{11}$), where $i > 1$, with a probability of 2^{-1}.

To estimate the complexity of ChaCha 5, the linear approximation from ChaCha 4 to ChaCha 5 occurs with probability 1, as per Lemma 5.1. The following bits are active in the $\mathcal{OD}$ position: $\Delta x^{(5)}_{4[19]} \oplus \Delta x^{(5)}_{8[0,12]} \oplus \Delta x^{(5)}_{12[0]}$. Given the active $\mathcal{OD}$ positions, we consider the non-LSB positions of keywords. For ChaCha 5, we compute the linear approximation of $\Delta x^{(5)}_{4[19]} \oplus \Delta x^{(5)}_{8[12]}$. In this case, the linear approximation for the final modular addition occurs with correlation $\frac{1}{2}(1 + 2^{-2})$, and the final complexity is presented in Table 5.7. To compute the attack complexity for ChaCha 5.5, we determine the linear approximation of the final modular addition for the following active bits at $\Delta x^{(5.5)}_{4[31]} \oplus \Delta x^{(5.5)}_{8[12]} \oplus \Delta x^{(5.5)}_{9[19]}$, where the final modular addition occurs with probability 2^{-3}. The final complexity of the distinguishing attack for ChaCha 5.5 is $2^{63.21}$, as presented in Table 5.7.

For ChaCha 6, we compute the linear approximation of the final modular addition for the following active bits in the $\mathcal{OD}$ position: $\Delta x^{(6)}_{4[6]} \oplus \Delta x^{(6)}_{6[7]} \oplus \Delta x^{(6)}_{8[12]} \oplus \Delta x^{(6)}_{9[19,31]}$. The linear approximation of the final modular addition occurs with probability 2^{-5},

and the final complexity occurs with probability $2^{87.21}$.

Table 5.7: The Distinguishing Attack Complexity

Target	$\mathcal{OD}$	$\lvert\varepsilon_d^*\rvert$	$\lvert\varepsilon_d^*\rvert \cdot \varepsilon_L^4$	Complexity
5	$\Delta X^{(4)}_{4,[0]}$	$2^{-15.53}$	$2^{-15.60964}$	$2^{33.21}$
5.5	$\Delta X^{(4)}_{4,[0]}$	$2^{-15.53}$	$2^{-19.60964}$	$2^{63.21}$
6	$\Delta X^{(4)}_{4,[0]}$	$2^{-15.53}$	$2^{-23.60964}$	$2^{87.21}$

5.4 The attack specification

This part outlines the characteristics of our proposed attack, highlighting the main tactics that enhance its efficiency.

- Our approach is based on higher-order differential cryptanalysis, distinguishing it from earlier ChaCha attacks that primarily relied on first-order differential bias. This method allowed us to leverage ChaCha's $4th$ internal round bias.
- To achieve an accurate attack, we used the median bias ε_d^*. Our focus was primarily on the 4th round forward bias produced through higher-order differential cryptanalysis. When analyzed using higher-order differential cryptanalysis, the ChaCha stream cipher exhibits a consistent median bias at the 3rd, 3.5th, and 4th internal rounds. By taking advantage of this weakness in the ChaCha QR function, we targeted the 4th round, which notably reduced the attack's complexity. See Table 5.3 and 5.4 for details.
- We developed specific strategies for selecting multiple $\mathcal{ID}$ and $\mathcal{OD}$ positions. By applying the Hamming Weight method to the $\mathcal{ID}$ positions, we reduced the number of potential $\mathcal{ID}$. In contrast to the techniques mentioned in chapter 3, which rely on either random selection or exhaustive searches of $\mathcal{ID}$ positions, our approach does not involve these methods. Instead, it benefits from the higher bias generated through the Hamming Weight technique during the selection process, thereby strengthening the attack. For more details, see Subsection 5.1.
- The selection of the $\mathcal{OD}$ position is guided by a neutrality measure calculated across 256 key bit positions for each $\mathcal{OD}$. We employed Algorithm 4 to facilitate this selection, allowing us to pinpoint an optimal $\mathcal{OD}$ position with a higher bias. Positions with a greater neutrality measure lead to a more advantageous bias, simplifying the attack. Our experiments supported this analysis. Please see Tables 5.1, 5.2, 5.3, 5.4, as well as Figures 2 and 3.

- Many existing attacks primarily rely on linear approximation to simplify the attack, often focusing on the 3rd or 3.5th round of differentials. Our approach aimed to strengthen the attack by concentrating on the differential component rather than the linear one. Consequently, we focused on the confirmed bias in the 4th round. Although the core of the attack is based on differential analysis using 4th rounds, we also utilized 2 and 3 rounds of linear approximation, which further enhanced the attack's overall effectiveness. For more details, see Subsection 5.2.
- We consider the differential and linear elements of the attack as two distinct ciphers.

Chapter 6

The Boomerang Attack on ChaCha Permutation

This chapter presents the boomerang attack on reduced rounds of the ChaCha permutation function. We have conducted the inaugural study on boomerang cryptanalysis of the ChaCha permutation. Our research includes the introduction of a distinguisher of ChaCha 6 and ChaCha 7 permutations. We introduced the boomerang bias for ChaCha 6 and ChaCha 7 and proposed an algorithm that can be used to mount a boomerang distinguisher of ChaCha permutation. The complexity of the boomerang attack is calculated as p^2q^2.

We found that with proper selection of the output difference ($\mathcal{OD}$) position, the probability of differentials can rise to p^2. We introduced the boomerang attack positions on ChaCha 6 and ChaCha 7. We proposed the boomerang distinguisher for ChaCha 6 and ChaCha 7 as $2^{4.87}$ and $2^{5.99}$, respectively. The boomerang attack and its variation can potentially attack the higher rounds of ChaCha stream cipher.

6.1 Boomerang Attack on ChaCha

Given the structure of the ChaCha stream cipher and the framework of differential cryptanalysis, it is nearly impossible to identify a valid differential bias ε_d beyond 3.5 rounds of ChaCha. Consequently, researchers employ the differential-linear adversary model to target ChaCha 6, ChaCha 7, and ChaCha 7.25. This model combines 3 or 3.5 differential rounds with 3 or 3.5 linear rounds, producing a differential-linear bias denoted as $\varepsilon_d \cdot \varepsilon_l$ for ChaCha 6 and ChaCha 7. ChaCha's resistance to standard differential attacks significantly increases after 3.5 rounds, making it an optimal target for a boomerang attack.

In this approach, we exploit the differential properties of ChaCha's sub-ciphers, merging the differential characteristics found in either 3 or 3.5 rounds. For ChaCha 6, we divide the cipher into two sub-ciphers, each containing 3 rounds, and combine

the differentials of each sub-round. Similarly, for ChaCha 7, we partition it into two groups of 3.5 rounds, merging the differentials of each sub-cipher to determine the differential bias for ChaCha 7. This method enables the creation of a comprehensive differential bias, ε_d, representing both ChaCha 6 and ChaCha 7. The differential probability in ChaCha's reverse round is higher than in the forward round, facilitating the execution of a boomerang attack (see Section 6.2.3). Consequently, the boomerang attack generates a differential bias ε_d for up to 6 and 7 rounds of ChaCha.

We applied the boomerang cryptanalysis method to the ChaCha stream cipher to assess the resistance of ChaCha 6 and ChaCha 7. This is the first purely differential attack reaching these rounds of ChaCha. Previous research discussed in Section 3 has relied on the differential basis of ChaCha 3 or ChaCha 3.5 rounds. In this paper, we utilize the boomerang differential bias of ChaCha 6 and ChaCha 7 by dividing ChaCha 6 into 3 sub-rounds and ChaCha 7 into 3.5 sub-rounds. The decision on the number of sub-cipher rounds is based on the following considerations:

- Since we used a single-bit difference for the boomerang attack, exceeding 3.5 rounds in ChaCha introduces an unreliable bias, affecting the overall accuracy of the attack. Therefore, we limit our attack to 3 and 3.5 sub-rounds.
- Splitting the cipher into more than two sub-ciphers for ChaCha 6 and ChaCha 7 significantly decreases the probability of differentials. However, this method could be adapted for different scenarios.
- The arrangement of rounds for sub-ciphers can be altered. Nonetheless, the choice of sub-cipher rounds must consider ChaCha's bias in each sub-round. For example, dividing ChaCha 7 into two sub-ciphers with 5 and 2 rounds each would be ineffective due to the difficulty in obtaining a verified bias for 5 rounds of ChaCha.
- The fundamental concept of the boomerang attack involves dividing the cipher into two sub-ciphers with high differential probabilities. Thus, we focus our study on two sub-ciphers.

We assume each sub-round operates independently. Algorithm 5 is proposed for attacking ChaCha.

We provide a comprehensive explanation of Algorithm 5. To construct the distinguisher for the reduced rounds of ChaCha (specifically ChaCha 6 and ChaCha 7), we utilized Algorithm 5. Initially, two ChaCha matrices were initialized with a single-bit difference at the $\Delta X_{12}^{(0)}[0]$ position. For attacking ChaCha 7, we divided the cipher into two segments, each comprising 3.5 rounds. In the case of ChaCha 6, the cipher was divided into two sub-ciphers, each containing 3 rounds.

Algorithm 5 The Boomerang Attack Algorithm on ChaCha

Input: : The initial states (X_1, X_2) with input difference α.

Output: The Differential Bias of ChaCha reduced rounds.

1. Initialize a counter **ctr** $\leftarrow 0$
2. For allrandom pairs key and IV trials
3. Initialize $X_1^{(0)} \oplus X_2^{(0)} = \Delta X_{12,[0]}^{(0)}$.
4. Encrypt the $X_1^{(0)}$ and $X_2^{(0)}$ with ChaCha E_0 to get $Y_1^{(r)}$ and $Y_2^{(r)}$.
5. If $Y_1^{(r)} \oplus Y_2^{(r)} = \beta$ after r rounds
6. Increment **ctr**
7. Encrypt the $Y_1^{(r)}$ and $Y_2^{(r)}$ with ChaCha E_1 to get $Z_1^{(R)}$ and $Z_2^{(R)}$.
8. Compute $Z_3^{(R)} = Z_1^{(R)} \oplus \Delta X_{12,[0]}^{(R)}$ and $Z_4^{(R)} = Z_2^{(R)} \oplus \Delta X_{12,[0]}^{(R)}$.
9. Ask for the decryption of $(Z_3^{(R)}, Z_4^{(R)})$ to $(Y_3^{(r)}, Y_4^{(r)})$ with ChaCha E_1^{-1}.
10. If $Y_1^{(r)} \oplus Y_3^{(r)} = \alpha$, $Y_2^{(r)} \oplus Y_4^{(r)} = \alpha$, and $Y_3^{(r)} \oplus Y_4^{(r)} = \beta$.
11. Increment **ctr** for each.
12. Compute the probability of $Y_1^{(r)} \oplus, Y_2^{(r)} = \beta$ as p, $Y_1^{(r)} \oplus Y_3^{(r)} = \gamma$ as q, $Y_2^{(r)} \oplus Y_4^{(r)} = \gamma$ as q and $Y_3^{(r)} \oplus Y_4^{(r)} = \beta$ as p.
13. Get the quartet probability as p^2q^2
14. Calculate the bias as $2 \cdot p^2q^2 - 1$

We encrypted the initial state matrices using E_0, a sub-cipher of the encryption function E, to obtain $Y_1^{(r)}$ and $Y_2^{(r)}$. The difference after r rounds ($r = 3$, $r = 3.5$) is computed as $Y_1^{(r)} \oplus Y_2^{(r)} = \beta$. It is essential to retain the two resulting ciphers Y_1 and Y_2 generated by E_0 to analyze the difference produced by E_1^{-1}. Following the E_0 operation, we applied the E_1 sub-cipher to obtain $Z_1^{(R)} = E_1(Y_1^{(r)})$ and $Z_2^{(R)} = E_1(Y_2^{(r)})$. A single-bit difference was introduced at the $\Delta X_{12,[0]}^{(R)}$ position in the $Z_1^{(R)}$ and $Z_2^{(R)}$ matrices generated by E_1 to produce $Z_3^{(R)} = Z_1^{(R)} \oplus \Delta X_{12,[0]}^{(R)}$ and $Z_4^{(R)} = Z_2^{(R)} \oplus \Delta X_{12,[0]}^{(R)}$. Since each sub-cipher is assumed to operate independently, the difference in $Z_1^{(R)}$ and $Z_2^{(R)}$ matrices can differ from the difference in the initial state matrices.

Next, we decrypted $Z_3^{(R)}$ and $Z_4^{(R)}$ using the inverse round E_1^{-1} (as depicted in Fig 6.2) to get $Y_3^{(r)} = E_1^{-1}(Z_3^{(R)})$ and $Y_4^{(r)} = E_1^{-1}(Z_4^{(R)})$. The E_1^{-1} process produced intermediate states, allowing us to identify and evaluate three potential probabilistic

differences: $Y_1^{(r)} \oplus Y_3^{(r)} = \alpha$, $Y_2^{(r)} \oplus Y_4^{(r)} = \alpha$. According to Equation 2.10, if $Y_1^{(r)} \oplus Y_3^{(r)} = \alpha$ and $Y_2^{(r)} \oplus Y_4^{(r)} = \alpha$, this would imply $Y_3^{(r)} \oplus Y_4^{(r)} = \beta$. This is further elaborated in step 11 of Algorithm 5.

We utilized the Maximum Length random number generator to create random keys and IVs, conducting our experiment on an Intel(R) Xeon(R) CPU E7-4830 v4 @ 2.00GHz machine with a total complexity of 2^{40}. To ensure the reproducibility of our results, we used the hexadecimal value 0xAFFFFFFF as the seed for generating random initialization vectors (IVs) and 0xAEEEEEEE as the seed for generating random keys. We performed the same attack on different positions using various seeds; however, the results did not show significant variations.

6.2 The Boomerang Distinguisher

When selecting a single $\mathcal{ID}$ in ChaCha, there can be up to 512 possible $\mathcal{OD}$ positions. Chapter 4 emphasized that the influence of $\mathcal{OD}$ on bias is more significant than that of $\mathcal{ID}$. Therefore, careful selection of $\mathcal{OD}$ is essential. In this study, we chose the non-significant bits of the $\mathcal{OD}$ words (i.e., the zero positions). These non-significant bits are less influenced by the ChaCha quarter-round function (refer to Equation 2.3). Our experimental results also supported this observation. Further details will be provided in Section 6.2.3.

6.2.1 Attack on ChaCha 7 Permutation

To target ChaCha 7, we initialized $X_1^{(0)}$ and $X_2^{(0)}$ as the starting values with $X_1^{(0)} \oplus X_2^{(0)} = \Delta X_{12}^{(0)}[0]$. Next, we applied the encryption function E_0 to these initial values to obtain the intermediate states $Y_1^{(3.5)} = E_0(X_1^{(0)})$ and $Y_2^{(3.5)} = E_0(X_2^{(0)})$. The $\mathcal{OD}$ for $Y_1^{(3.5)}$ and $Y_2^{(3.5)}$ was calculated as $Y_1^{(3.5)} \oplus Y_2^{(3.5)} = \Delta_{0,1,2,12,13,14,15}^{(3.5)}[0]$ across 7 possible positions. We represent the probability of $Y_1^{(3.5)} \oplus Y_2^{(3.5)}$ as p_1. According to the technical details of Algorithm 5, the outcome of the E_0 encryption function is retained before passing it to E_1. These results from E_0 will be used to calculate the differences with the output of E^{-1} to determine new probabilities.

Subsequently, we encrypt the states $Y_1^{(3.5)}$ and $Y_2^{(3.5)}$ using E_1 (i.e., 3.5 rounds) to obtain $Z_1^{(7)} = E_1(Y_1^{(3.5)})$ and $Z_2^{(7)} = E_1(Y_2^{(3.5)})$. We then perform the following steps to derive the new matrices for the E_1^{-1} cipher.

$$Z_3^{(7)} = Z_1^{(7)} \oplus \Delta X_{12,[0]}^{(R)}. \tag{6.1}$$

$$Z_4^{(7)} = Z_2^{(7)} \oplus \Delta X_{12,[0]}^{(R)}. \tag{6.2}$$

The $\mathcal{ID}$ position for Equations 6.1 and 6.2 was chosen as $\Delta X_{12}^{(7)}[0]$, although a different position could also be selected. Next, we input the newly generated states $Z_3^{(7)}$ and $Z_4^{(7)}$ into the reverse round E_1^{-1}.

$$Y_3^{(3.5)} = E_1^{-1}(Z_3^{(7)}). \tag{6.3}$$
$$Y_4^{(3.5)} = E_1^{-1}(Z_4^{(7)}). \tag{6.4}$$

At this stage, to construct the distinguisher, we compute the following differences based on Equation 2.9:

$$Y_1^{(3.5)} \oplus Y_3^{(3.5)} = \gamma. \tag{6.5}$$
$$Y_2^{(3.5)} \oplus Y_4^{(3.5)} = \gamma. \tag{6.6}$$

We denote the probability of $Y_1^{(3.5)} \oplus Y_3^{(3.5)}$ and $Y_2^{(3.5)} \oplus Y_4^{(3.5)}$ with q_1 and q_2, respectively. Given the Equation 2.10, the Equations 6.5 and 6.6 will force the $Y_3^{(3.5)}$ and $Y_4^{(3.5)}$ to have the have the following difference.

$$Y_3^{(3.5)} \oplus Y_4^{(3.5)} = \beta.$$

The differences are computed over 3.5 rounds for the $\mathcal{OD}$ positions $\Delta_{0,1,2,12,13,14,15}^{(3.5)}[0]$ with the same $\mathcal{ID}$ positions at both $\Delta X_{12}^{(0)}[0]$ and $\Delta X_{12}^{(7)}[0]$. Analyzing the results of ChaCha 7 produced by the boomerang attack, we observed that the probability and bias for all $\mathcal{OD}$ positions yielded identical values, differing from those noted at $R = 6$. The individual probability distribution in our experiment for boomerang attacks is quite similar to the probabilities in standard differential attacks. To confirm this, we calculated the bias for the individual probabilities. For example, the bias for p_2 and q_2 is 0.000002, which can be verified with approximately $2^{38.86}$ queries. We ran the experiment with a total complexity of 2^{40}. This indicates that individual bias can also be verified. The probability distribution of boomerang probability (i.e., p^2q^2) is illustrated in Fig 6.1.

Among the $\mathcal{OD}$ positions we targeted, we only presented the results for $\Delta X_0^{(3.5)}[0]$. Table 6.1 provides a summary of the median probability values and the details of the bias. The bias is computed based on the probability of boomerangs, p^2q^2. For the bias calculation, see Step 17 of Algorithm 5.

According to Dunkelman [DKRS20], a total of $4(pq)^{-2}$ adaptively chosen plaintexts and ciphertexts are required to distinguish ChaCha from a random permutation. Therefore, given the value of p^2q^2, approximately $2^{5.9999971}$ adaptively chosen plaintexts and ciphertexts are necessary to distinguish ChaCha 7 from a random permutation.

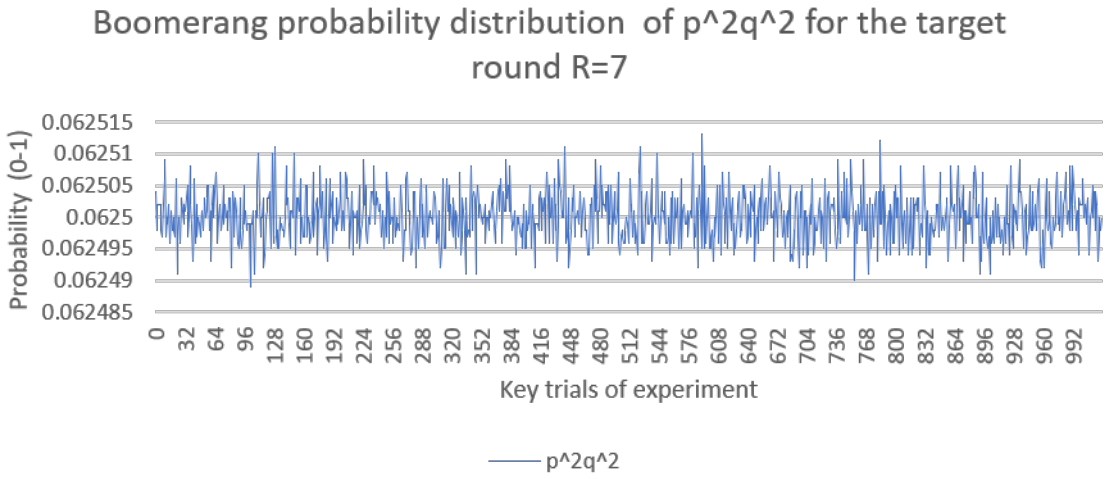

Figure 6.1: Boomerang probability distribution

While the concepts of adaptively chosen plaintexts and ciphertexts are different for block ciphers, block ciphers process fixed-size data blocks, and analyzing multiple plaintext-ciphertext pairs can reveal patterns in the encryption. However, in the context of our proposed attack, we assume that the adversary can manipulate the resulting matrix produced by the ChaCha stream cipher to introduce a difference, thereby converting the attack into an adaptively chosen plaintext and ciphertext attack.

Table 6.1: The Boomerang Attack Probability for ChaCha 7

Intermediate $\mathcal{OD}$	p_1^*	p_2^*	q_1^*	q_2^*	p^2q^2	$\lvert\varepsilon_d^*\rvert$	Complexity
$\Delta X_0[0]^{(3.5)}$	0.5	0.5	0.4999	0.5	0.0625	0.875	$2^{5.9999}$

6.2.2 Attack on 6-rounds ChaCha Permutation

To analyze the ChaCha 6 cipher, we applied the same method as the one used for attacking ChaCha 7. We divided the ChaCha 6 encryption into two sub-rounds, separately for E_0 and E_1. ChaCha 6 was examined for all possible $\mathcal{OD}$ positions $\Delta^{(3)}_{0,1,2,4,5,6,7,8,9,10,11,12,13,14,15}[0]$ to assess the impact of $\mathcal{OD}$ positions in a boomerang attack. The overall complexity of the distinguisher varies for each $\mathcal{OD}$ position.

In our experiment, we observed that the probability of $Y_1^{(r)} \oplus Y_3 = \gamma$ and $Y_2 \oplus Y_4 = \gamma$ at $\Delta X_1^{(3)}[0]$ and $\Delta X_5^{(3)}[0]$ occurs with a probability of 1, which increases the boomerang probability to p^2. The reduction from p^2q^2 to p^2q was discussed in [KT22].

Table 6.2: The Boomerang Attack Probability for ChaCha 6

Intermediate $\mathcal{OD}$	p_1^*	p_2^*	q_1^*	q_2^*	p^2q^2	$\lvert\varepsilon_d^*\rvert$	Complexity
$\Delta X_0[0]^{(3)}$	0.4999	0.4999	0.9841	0.9841	0.2411	0.5177	$2^{4.587}$

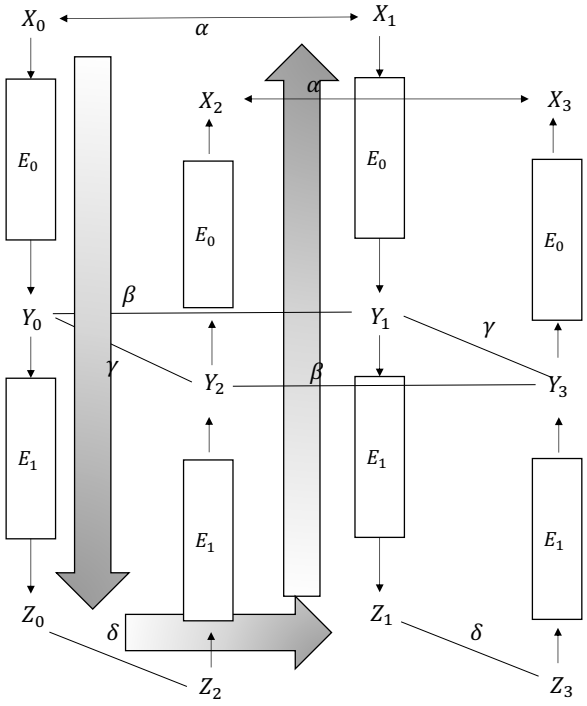

Figure 6.2: The boomerang attack on ChaCha

Table 6.2 provides a summary of the probabilities and bias details. For the $\Delta X_0[0]^{(3)}$ position, up to $2^{4.578}$ adaptively chosen plaintexts and ciphertexts are required to distinguish ChaCha 6 from a random permutation.

6.2.3 The Boomerang Differential Trails of ChaCha

This section examines the propagation of an input difference $\mathcal{ID}$ in the context of a boomerang attack. We focus on the target attack points $X_0[0]^{(3)}$ and $X_0[0]^{(3.5)}$ for ChaCha 6 and ChaCha 7, respectively. For both ChaCha 6 and ChaCha 7, an input difference $\mathcal{ID}$ is introduced at $\Delta X_{12}^{(0)}[0]$. After the first round, this difference propagates to the first column of the ChaCha matrix, affecting $X_0^{(1)}[28]$, $X_4^{(1)}[31, 28, 23, 12, 3]$, $X_8^{(1)}[31, 28, 24, 23, 16, 12, 4, 3]$, and $X_{12}^{(1)}[16]$. By the end of the second round, the difference spreads to all words. Since we are concentrating on the $X_0^{(3)}[0]$ and $X_0^{(3.5)}[0]$ positions, we trace changes in these specific locations. In the second round, $X_0^{(2)}[24]$ is affected. During the third round, the position $X_0^{(3)}[29, 28, 27, 26, 25, 23, 22, 21, 19, 17, 15$ $, 14, 12, 11, 8, 7, 5, 4, 3, 2]$ is altered, with $X_0[0]^{(3)}$ remaining unchanged.

Moving to the $X_0[0]^{(3.5)}$ rounds, the difference propagates to $X_0^{(3.5)}[29, 28, 27, 26, 24,$ $21, 19, 18, 14, 13, 9, 8, 7, 2]$, while the attack position $X_0[0]^{(3.5)}$ remains predictable and unaffected. The reverse round of ChaCha operates differently than the forward round. After the first reverse round, the difference affects the first column of the ChaCha matrix, altering only one bit at $X_0^{(1)}[0]$, all bits of $X_4^{(1)}$, a single bit of $X_8^{(1)}[24]$, and all bits of $X_{12}^{(1)}$ except $X_{12}^{(1)}[16]$. After the second reverse round, unlike in the forward

round, not all words are updated. For example, $X_5^{(2)}$ and $X_9^{(2)}$ remain unchanged. Additionally, a single bit in $X_1[8]^{(2)}$, $X_3[1]^{(2)}$, $X_{12}[16]^{(2)}$, and $X_{14}[0]^{(2)}$ is updated. $X_4^{(2)}$ and $X_{10}^{(2)}$ are updated at all positions.

The difference propagation in the ChaCha reverse round is predictable and occurs with high probability (refer to Table 6.2). The third reverse round propagates changes to all positions. Our attack points $X_0^{(3)}$ are altered at $X_0^{(3)}[16, 12, 1]$ and $X_0^{(3.5)}[24, 17, 16, 13]$ after three and three and a half rounds, respectively. The weak and predictable propagation of the ChaCha reverse round makes it vulnerable to a boomerang attack. The median probability of a forward round for ChaCha 3 is 0.499, whereas it is 0.9841 for the reverse rounds.

6.2.4 Factors Driving Significant Improvements in Attacks

The improvement exclusively comes from the way we calculate the boomerang distinguisher compared to the differential cryptanalysis or differential-linear attacks. In differential attacks, the distinguisher is calculated as $\mathcal{O}(1/p)$ where p is the differential probability. In a differential-linear attack, we calculate the distinguisher as $\mathcal{O}(1/pq^2)$ where the p is differential probability and q is linearity. However, for the boomerang attack, we consider four probabilities from two independent sub-ciphers, and the boomerang distinguisher can be calculated as $4(pq)^{-2}$. In addition, the differential probability in the reverse round of ChaCha is higher than the forward round, especially for ChaCha 3, increasing the attack efficiency.

Chapter 7

Discussion

The analysis of the Salsa 20 and ChaCha stream ciphers is a hot topic in the current domain of cryptanalysis. Given their structure, differential cryptanalysis is the primary adversary model for studying the security of ChaCha and Salsa 20. Even though the structure of differential cryptanalysis is already defined, the refinement and strategy to mount an attack are crucial parts of an attack on Salsa 20 and ChaCha stream ciphers. This dissertation studied the security of Salsa 20 and ChaCha stream ciphers based on differential, differential-linear, and boomerang cryptanalysis methodologies. We analyze and interpret the research findings, and discuss future possible improvements. In addition, it briefly discusses the limitations of the existing studies and our proposed methods for increasing attack efficiency.

The selection of $\mathcal{ID}$ and $\mathcal{OD}$ is crucial in mounting the attack. The studies reported in Chapter 3 used the select input difference first and then looked for the output differential position to mount the attack. This method could be refined and applied differently to improve the attack efficiency of Salsa 20 and ChaCha. To get an optimal pair of $\mathcal{ID}, \mathcal{OD}$, we look for the $\mathcal{OD}$ position first and then search for the $\mathcal{ID}$ position with the best differential bias ε_d.

To achieve this, we first looked for the $\mathcal{OD}$ position with the highest average neutrality measure concerning all 256 keybit positions. We executed the Algorithm 3 and analyzed the generated result to find the $\mathcal{OD}$ position with the highest average neutral measure. To attack Salsa 8, we selected the $\mathcal{OD}$ position $\Delta_1^{(4)}[13]$ which generates the average neutral measure $\gamma_k = 0.083327$. We searched for the possible $\mathcal{ID}$ positions and selected the $\Delta_8^{(0)}[27], \Delta_{11}^{(4.75)}[13]$ as an attacking point with the median basis $\varepsilon_d^* = 0.0651375$.

We achieved an attack on Salsa 8 with time complexity of $2^{241.62}$ and data complexity of $2^{31.4}$. We applied the same approach on ChaCha 7.25 and selected the $\Delta_{13}^{(0)}[6], \Delta_2^{(3.5)}[0]$ as an attacking point with the median basis $\varepsilon_d^* = 0.000474$. As a result, we mounted a key-recovery attack on ChaCha 7.25 with a time complexity of $2^{254.022}$ and data complexity of $2^{51.81}$. The application of PNB-based differential

cryptanalysis discussed in Chapter 4 presented several outcomes.

The experimental result revealed the relationship between the $= \mathcal{OD}$ and the neutrality measure of 256 keybit positions. Moreover, the method helped to find a new attack pair of $\mathcal{ID}, \mathcal{OD}$ with the best differential bias, and as a result, we could decrease the attack complexity on Salsa 20 and ChaCha stream ciphers. While we introduced a new attacking point and refined the attack introduced by [AFK+08], we believe applying differential cryptanalysis methods such as differential-linear attacks could improve the attack.

Chapter 4 introduced the 4.75 internal round to attack the ChaCha 8. We tried to use the 5th internal round to attack ChaCha 8 but, we were not able to find reliable bias. We attempted to increase the number of PNBs but we could not get reliable backward bias with the 4.75 internal rounds. The Salsa 20 quarter round could be further scrutinized to increase the number of PNBs and select a better attacking point. In addition, the relationship between the PNBs and the ChaCha quarter round is not unexplored yet, this can potentially add to the number of PNBs and reduce the final attack complexity. We believe the attack can be further improved by new refinement and attack methodologies.

Chapter 5 presented the higher-order differential-linear attack on the ChaCha. The existing studies focused on extending linear approximation after 3.5 rounds and improving attack mainly by concentrating on linear approximation for higher rounds. This methodology could be improved in two proposed ways. The research focused on two strategies to decrease the attack complexity.

The primary aim was to improve the differential bias in the forward rounds. We used second-order differentials to target ChaCha's 4 internal rounds. We limited our attack to ChaCha 6. As a result, we extended the $4-th$ round differential attack to the $5-th$ round with probability 1, and from the $5-th$ round to the $6-th$ round with a probability less than one. This approach helped us obtain a higher linear bias, which we combined with the differential bias to attack ChaCha 6. Consequently, we introduced a distinguishing attack on ChaCha 6 with a complexity of $2^{87.21}$.

We tried to extend the linear approximations from ChaCha 4 to ChaCha 7, however with the $\mathcal{ID}, \mathcal{OD}$ in chapter 5 we could not improve beyond ChaCha 6 due to the final modular addition used to generate the key stream. probability. The attack could be extended using new linear approximations from the $4-th$ round to the $6-th$ rounds of ChaCha with new the $\mathcal{ID}, \mathcal{OD}$. Moreover, the attack in Chapter 5 is a distinguishing attack, this attack can turn into a key-recovery attack. Additionally, we used the second-order differential and introduced the $\mathcal{ID}$ positions, however, the attack could turn into third-order or fourth-order and the selection of $\mathcal{ID}$ position would be a challenging part and it should be addressed in future studies.

Chapter 6 introduced a distinguisher on the ChaCha permutation function. The

research applied boomerang cryptanalysis to attack the reduced rounds of the ChaCha stream cipher using the chosen plaintext and ciphertext attack methodology. As a result, we introduced a distinguisher on ChaCha 6 and ChaCha 7 permutation functions with complexities of $2^{4.587}$ and $2^{5.99}$, respectively.

Chapter 8

Conclusion and Future Works

This chapter concludes the dissertation, summarizing the research findings and outlining possible future work. The dissertation explores the cryptanalysis of the Salsa 20 and ChaCha stream ciphers.

This dissertation assesses the security of the Salsa 20 and ChaCha stream ciphers, extensively utilized across various digital platforms. Our study explores possible vulnerabilities in these ciphers using differential, differential-linear, and boomerang attacks. We seek to evaluate the resilience of Salsa 20 and ChaCha against these attacks and propose techniques to exploit their weaknesses. The results offer valuable insights into the security characteristics of these stream ciphers and may inform the design of more secure cryptographic algorithms in the future.

Chapter 4 presented an attack on Salsa 20 and ChaCha stream ciphers. The attack is based on [AFK+08] idea with refinements. The research introduced new attack points called $\mathcal{ID}, \mathcal{OD}$ to mount a key-recovery attack on reduced rounds of Salsa 20 and ChaCha stream ciphers. The chapter analyzed the relationship between the Salsa 20 quart1er-round function and probabilistic neutral bits to justify the experimental results. We identified $\Delta_8^{(0)}[27], \Delta_{11}^{(4.75)}[13]$ as an $= \mathcal{ID}, \mathcal{OD}$ with the bias $\varepsilon_d = 0.065137$. As a result, we introduced a key-recovery attack on Salsa 20 with a time complexity of $2^{241.62}$. We applied the same attack scenario on ChaCha. We mount the attack with the $\Delta_{13}^{(0)}[6], \Delta_2^{(3.5)}[0]$ and the bias $\varepsilon_d = 0.000474$. As a result, we presented a key-recovery attack on ChaCha 7.25 with a complexity of $2^{254.011}$.

Chapter 5 investigates the security of ChaCha against higher-order differential-linear cryptanalysis. In this research, we explored higher-order and higher-order differential-linear cryptanalysis and applied these methods to the ChaCha stream cipher. We present the higher-order and higher-order differential-linear biases across different rounds of ChaCha. We propose attacks targeting ChaCha 5, ChaCha 5.5, and ChaCha 6. Our proposed attacks have complexities of $2^{33.21}$, $2^{63.21}$, and $2^{87.21}$ on ChaCha 5, ChaCha 5.5, and ChaCha 6, respectively. This attack framework has the potential to become an essential method for assessing the security of various ciphers.

Chapter 6 analyzed the ChaCha cipher in the context of the boomerang attack. We developed an algorithm to target ChaCha rounds 6 and 7 permutation functions. This attack model utilizes adaptively chosen plaintext and ciphertext to compromise ChaCha. According to our analysis, $2^{4.587}$ and $2^{5.9999}$ adaptively chosen plaintext and ciphertext can distinguish ChaCha 6 and ChaCha 7 from a random permutation, respectively. This represents the first boomerang attack and the most effective attack on reduced rounds of the ChaCha cipher. Differential cryptanalysis in Chapter 4 introduced a key-recovery attack on Salsa 8 and ChaCha 7.25. The attack complexity can be reduced on Salsa 8 by further scrutinizing the Salsa 20 and ChaCha quarter-round functions.

Additionally, the proposed attack could be changed to a differential-linear attack to improve the attack on Salsa 8 and ChaCha 7.25. Furthermore, the proposed attack in Chapter 5 introduced the higher-order differential-linear distinguishing attack on ChaCha 6. However, the attack could be further improved by focusing on linear approximation and applying the key recovery attack based on PNBs. While the introduced boomerang attack on ChaCha in Chapter 6 targeted ChaCha 7 based on the chosen plaintext-ciphertext assumption, the attack could be turned into an amplified boomerang attack to introduce a chosen plaintext attack and make it more practical.

List of Publications

Journals

1. Nasratullah Ghafoori, Atsuko Miyaji, Ryoma Ito and Shotaro Miyashita, PNB based differential cryptanalysis of Salsa 20 and Chacha, IEICE TRANSACTIONS on Information and Systems, vol.E106-D, no.9, pp.1407-1422, 2023.

2. Nasratullah Ghafoori and Atsuko Miyaji, Higher-Order Differential-Linear Cryptanalysis of ChaCha Stream Cipher, IEEE Access, 2024.

International Conferences

1. Nasratullah Ghafoori and Atsuko Miyaji, Differential cryptanalysis of Salsa20 based on comprehensive analysis of PNBs, International Conference on Information Security Practice and Experience (ISPEC), LNCS, vol.13620, Springer, pp.520-536, 2022.

2. Ryo Watanabe, Nasratullah Ghafoori and Atsuko Miyaji, Improved Differential-Linear Cryptanalysis of Reduced Rounds of ChaCha, International Conference on Information Security Applications (WISA), LNCS, vol.14402, Springer, pp.269-281, 2023.

3. Nasratullah Ghafoori and Atsuko Miyaji, The Boomerang Attack on ChaCha Stream Cipher Permutation, International Conference on Computer Communication and the Internet (ICCCI), IEEE, pp.18-23, 2024.

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 *Fast Software Encryption: 15th International Workshop, FSE 2008, February 10-13, 2008, Revised Selected Papers 15*, pages 470–488. Springer, 2008.

[Alk16] Hoda A Alkhzaimi. Cryptanalysis of selected block ciphers. 2016.

[B+08] Daniel J Bernstein et al. ChaCha, a variant of Salsa20. In *Workshop record of SASC*, volume 8, pages 3–5. Citeseer, 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. *Journal of Cryptology*, 35(4):29, 2022.

[BBM20] Stefano Barbero, Emanuele Bellini, and Rusydi Makarim. Rotational analysis of ChaCha permutation. *arXiv preprint arXiv:2008.13406*, 2020.

[BDK05] Eli Biham, Orr Dunkelman, and Nathan Keller. New combined attacks on block ciphers. In *Fast Software Encryption: 12th International Workshop, FSE 2005, February 21-23, 2005, Revised Selected Papers 12*, pages 126–144. Springer, 2005.

[Ber08] Daniel J Bernstein. The Salsa20 family of stream ciphers. In *New stream cipher designs: the eSTREAM finalists*, pages 84–97. Springer, 2008.

[BGG+23] Emanuele Bellini, David Gerault, Juan Grados, Rusydi H Makarim, and Thomas Peyrin. Boosting differential-linear cryptanalysis of ChaCha7 with milp. *IACR Transactions on Symmetric Cryptology*, 2023.

[Bir11] Alex Biryukov. *Chosen Plaintext Attack*, pages 205–206. Springer US, Boston, MA, 2011.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4:3–72, 1991.

[BV14] Alex Biryukov and Vesselin Velichkov. Automatic search for differential trails in arx ciphers. In *Topics in Cryptology–CT-RSA 2014: The Cryptographer's Track at the RSA Conference 2014, February 25-28, 2014. Proceedings*, pages 227–250. Springer, 2014.

[BVLC16] Alex Biryukov, Vesselin Velichkov, and Yann Le Corre. Automatic search for the best trails in arx: application to block cipher speck. In *Fast Software Encryption: 23rd International Conference, FSE 2016, March 20-23, 2016, Revised Selected Papers 23*, pages 289–310. Springer, 2016.

[CM16] Arka Rai Choudhuri and Subhamoy Maitra. Significantly improved multi-bit differentials for reduced round Salsa and ChaCha. *IACR Transactions on Symmetric Cryptology*, pages 261–287, 2016.

[CN20] Murilo Coutinho and TC Souza Neto. New multi-bit differentials to improve attacks against ChaCha. *Cryptology ePrint Archive*, 2020.

[CPV+23a] Murilo Coutinho, Iago Passos, Juan C Grados Vásquez, Santanu Sarkar, Fábio LL de Mendoncomca, Rafael T de Sousa Jr, and Fábio Borges. Latin dances reloaded: improved cryptanalysis against Salsa and ChaCha, and the proposal of forró. *Journal of Cryptology*, 36(3):18, 2023.

[CPV+23b] Murilo Coutinho, Iago Passos, Juan C Grados Vásquez, Santanu Sarkar, Fábio LL de Mendoncomca, Rafael T de Sousa Jr, and Fábio Borges. Latin dances reloaded: Improved cryptanalysis against Salsa and ChaCha, and the proposal of forró. *Journal of Cryptology*, 36(3):18, 2023.

[CSN21] Murilo Coutinho and Tertuliano C Souza Neto. Improved linear approximations to arx ciphers and attacks against ChaCha. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, October 17–21, 2021, Proceedings, Part I 40*, pages 711–740. Springer, 2021.

[DDSM22] Sabyasachi Dey, Chandan Dey, Santanu Sarkar, and Willi Meier. Revisiting cryptanalysis on ChaCha from crypto 2020 and eurocrypt 2021. *IEEE Transactions on Information Theory*, 68(9):6114–6133, 2022.

[DGM23] Sabyasachi Dey, Hirendra Kumar Garai, and Subhamoy Maitra. Cryptanalysis of reduced round ChaCha-new attack and deeper analysis. *Cryptology ePrint Archive*, 2023.

[DGSS22] Sabyasachi Dey, Hirendra Kumar Garai, Santanu Sarkar, and Nitin Kumar Sharma. Revamped differential-linear cryptanalysis on reduced round

ChaCha. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 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 Transactions on Information Theory*, 2023.

[DK15] Hans Delfs and Helmut Knebl. *Symmetric-Key Cryptography*, pages 11–48. Springer Berlin Heidelberg, 2015.

[DKRS20] Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. The retracing boomerang attack. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 280–309. Springer, 2020.

[DL11] Ming Duan and Xuejia Lai. Higher order differential cryptanalysis framework and its applications. In *International Conference on Information Science and Technology*, pages 291–297. IEEE, 2011.

[DS17] Sabyasachi Dey and Santanu Sarkar. Improved analysis for reduced round Salsa and Chacha. *Discrete Applied Mathematics*, 227:58–69, 2017.

[DS23] Chandan Dey and Santanu Sarkar. A new distinguishing attack on reduced round ChaCha permutation. *Scientific Reports*, 13(1):13958, 2023.

[eP] The eSTREAM Project. http://www.ecrypt.eu.org/stream.

[HPTY23] Kai Hu, Thomas Peyrin, Quan Quan Tan, and Trevor Yap. Revisiting higher-order differential-linear attacks from an algebraic perspective. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 405–435. Springer, 2023.

[HW19] Mingjiang Huang and Liming Wang. Automatic tool for searching for differential characteristics in arx ciphers and applications. In *International Conference on Cryptology in India*, pages 115–138. Springer, 2019.

[KB96] Lars R Knudsen and Thomas A Berson. Truncated differentials of safer. In *Fast Software Encryption: Third International Workshop, February 21–23 1996 Proceedings 3*, pages 15–26. Springer, 1996.

[KHL+04] Jongsung Kim, Seokhie Hong, Sangjin Lee, Junghwan Song, and Hyungjin Yang. Truncated differential attacks on 8-round crypton. In *Information Security and Cryptology-ICISC 2003: 6th International Conference, November 27-28, 2003. Revised Papers 6*, pages 446–456. Springer, 2004.

[Knu95] Lars R Knudsen. Truncated and higher order differentials. In *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings 2*, pages 196–211. Springer, 1995.

[Knu98] Lars Knudsen. Deal-a 128-bit block cipher. *complexity*, 258(2):216, 1998.

[KR97] Lars R Knudsen and Vincent Rijmen. Truncated differentials of idea. *Department of Electrical Engineering, ESAT–COSIC Technical Report 97*, 1, 1997.

[KR11] Lars R Knudsen and Matthew Robshaw. *The block cipher companion*. Springer Science & Business Media, 2011.

[KT22] Andreas B Kidmose and Tyge Tiessen. Formal analysis of boomerang probabilities. *IACR Transactions on Symmetric Cryptology*, 2022(1):88–109, 2022.

[Lai94] Xuejia Lai. Higher order derivatives and differential cryptanalysis. *Communications and Cryptography: Two Sides of One Tapestry*, pages 227–233, 1994.

[Leu16] Gaëtan Leurent. Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 344–371. Springer, 2016.

[LH94] Susan K Langford and Martin E Hellman. Differential-linear cryptanalysis. In *Advances in Cryptology—CRYPTO' 94: 14th Annual International Cryptology Conference, August 21–25, 1994 Proceedings 14*, pages 17–25. Springer, 1994.

[LMM91] Xuejia Lai, James L Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology—EUROCRYPT' 91: Workshop on the Theory and Application of Cryptographic Techniques, April 8–11, 1991 Proceedings 10*, pages 17–38. Springer, 1991.

[LWD04] Helger Lipmaa, Johan Wallén, and Philippe Dumas. On the additive differential probability of exclusive-or. In *Fast Software Encryption: 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers 11*, pages 317–331. Springer, 2004.

[Mai16] Subhamoy Maitra. Chosen iv cryptanalysis on reduced round ChaCha and Salsa. *Discrete Applied Mathematics*, 208:88–97, 2016.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 386–397. Springer, 1993.

[MIM22] Shotaro Miyashita, Ryoma Ito, and Atsuko Miyaji. Pnb-focused differential cryptanalysis of ChaCha stream cipher. In *Australasian Conference on Information Security and Privacy*, pages 46–66. Springer, 2022.

[MP13] Nicky Mouha and Bart Preneel. A proof that the arx cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2013:328, 2013.

[MS01] Itsik Mantin and Adi Shamir. A practical attack on broadcast rc4. In *International workshop on fast software encryption*, pages 152–164. Springer, 2001.

[MZ06] Serge Mister and Robert Zuccherato. An attack on cfb mode encryption as used by openpgp. In *Selected Areas in Cryptography: 12th International Workshop, SAC 2005, August 11-12, 2005, Revised Selected Papers 12*, pages 82–94. Springer, 2006.

[NSLL22] Zhongfeng Niu, Siwei Sun, Yunwen Liu, and Chao Li. Rotational differential-linear distinguishers of arx ciphers with arbitrary output linear masks. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2022.

[S+99] Data Encryption Standard et al. Data encryption standard. *Federal Information Processing Standards Publication*, 112, 1999.

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[SHY16] Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of arx block ciphers with application to speck and lea. In *Australasian Conference on Information Security and Privacy*, pages 379–394. Springer, 2016.

[SZFW12] Zhenqing Shi, Bin Zhang, Dengguo Feng, and Wenling Wu. Improved key recovery attacks on reduced-round Salsa20 and ChaCha. In *International Conference on Information Security and Cryptology*, pages 337–351. Springer, 2012.

[Wag99] David Wagner. The boomerang attack. In *International Workshop on Fast Software Encryption*, pages 156–170. Springer, 1999.

[WLHL23] Shichang Wang, Meicheng Liu, Shiqi Hou, and Dongdai Lin. Moving a step of ChaCha in syncopated rhythm. In *Annual International Cryptology Conference*, pages 273–304. Springer, 2023.

[ZCL10] Bo Zhu, Kefei Chen, and Xuejia Lai. Bitwise higher order differential cryptanalysis. In *Trusted Systems: First International Conference, INTRUST 2009, Beijing, China, December 17-19, 2009. Revised Selected Papers 1*, pages 250–262. Springer, 2010.