DESIGN, DIFFUSION, AND CRYPTANALYSIS OF SYMMETRIC PRIMITIVES

MURILO COUTINHO SILVA

TESE DE DOUTORADO EM TELECOMUNICAÇÕES E REDES DE COMUNICAÇÃO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESIGN, DIFFUSION, AND CRYPTANALYSIS OF SYMMETRIC PRIMITIVES

DESENVOLVIMENTO, DIFUSÃO E CRIPTOANÁLISE DE PRIMITIVAS CRIPTOGRÁFICAS SIMÉTRICAS

MURILO COUTINHO SILVA

ORIENTADOR: RAFAEL T. DE SOUSA JR.
COORIENTADOR: FÁBIO BORGES DE OLIVEIRA

TESE DE DOUTORADO EM
TELECOMUNICAÇÕES E REDES DE
COMUNICAÇÃO

PUBLICAÇÃO: PPGEE.TD-189/22
BRASÍLIA/DF: NOVEMBRO - 2022

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESIGN, DIFFUSION, AND CRYPTANALYSIS OF SYMMETRIC PRIMITIVES

MURILO COUTINHO SILVA

TESE DE DOUTORADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR.

BRASÍLIA, 25 DE NOVEMBRO DE 2022.

FICHA CATALOGRÁFICA

COUTINHO, MURILO; DE SOUSA, RAFAEL; BORGES, FÁBIO
Design, Diffusion, and Cryptanalysis of Symmetric Primitives [Distrito Federal] 2022.
xix, 195p., 210 x 297 mm (ENE/FT/UnB, Doutor, Telecomunicações e Redes de Comunicação, 2022).
Tese de doutorado – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica
1. Criptografia, ARX, Criptoanálise, Difusão, Segurança, ChaCha, Salsa, Speck, AES
3. Forró
I. ENE/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

COUTINHO, M., DE SOUSA, R.T., BORGES, F. (2022). Design, Diffusion, and Cryptanalysis of Symmetric Primitives . Tese de doutorado em Telecomunicações e Redes de Comunicação, Publicação PPGEE.TD-189/22, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 195p.

CESSÃO DE DIREITOS

Murilo Coutinho Silva

Departamento de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

I dedicate this work to my family.

ACKNOWLEDGMENTS

I would like to thank my supervisors for their support during the preparation of this doctoral thesis. Additionally, I would like to thank the committee members for their various contributions that have improved the quality of the work.
I am grateful for the technical and computational support of the Decision-Making Technologies Laboratory - LATITUDE, of the University of Brasilia, which is supported by CNPq - Brazilian National Research Council (Grant 312180/2019-5 PQ-2, and 465741/2014-2 INCT on Cybersecurity), by CAPES - Brazilian Higher Education Personnel Improvement Coordination (Grant 88887.144009/2017-00 PROBRAL), by FAP-DF - Brazilian Federal District Research Support Foundation (Grant 0193.001366/2016 UIoT, and Grant 0193.001365/2016 SSDDC), by the Brazilian Ministry of the Economy (Grant 005/2016 DIPLA, and Grant 083/2016 ENAP), by the Institutional Security Office of the Presidency of Brazil (Grant ABIN 002/2017), by the Administrative Council for Economic Defense (Grant CADE 08700.000047/2019-14), by the General Attorney of the Union (Grant AGU 697.935/2019), by the National Auditing Department of the Brazilian Health System SUS (Grant DENASUS 23106.118410/2020-85), by the General Attorney's Office for the National Treasure (Grant PGFN 23106.148934/2019-67), and by the University of Brasilia (Grant UnB COPEI 7129).

RESUMO

Título: Desenvolvimento, Difusão e Criptoanálise de Primitivas Criptográficas Simétricas
Autor: Murilo Coutinho Silva
Orientador: Rafael T. de Sousa Jr.
Coorientador: Fábio Borges de Oliveira
Programa de Pós-Graduação em Telecomunicações e Redes de Comunicação
Brasília, 25 de novembro de 2022

Nessa tese de doutorado, novas técnicas de criptografia, criptoanálise e de desenvolvimento de algoritmos são estudadas e propostas. Resumidamente, os seguintes resultados são alcançados:

- Uma técnica denominada Análise de Difusão Contínua (CDA) é proposta. Utilizando-se essa técnica é possível se estudar, desenvolver e comparar algoritmos criptográficos. Com CDA é possível generalizar tais algoritmos a partir da transformação dos bits discretos em probabilidades, de tal forma que o algoritmo é generalizado em uma função matemática contínua. A partir disso, propõe-se três novas métricas de difusão a serem utilizadas nesse novo espaço contínuo, a saber: o Fator de Avalanche Contínuo (CAF), a Métrica de Neutralidade Contínua (CNM), e o Fator de Difusão (DF). Além disso, mostra-se que essas métricas de difusão podem ser utilizadas para avaliar e comparar algoritmos criptográficos. Em particular, o Fator de Difusão pode ser usado para comparar a difusão sem a necessidade de se reduzir o número de rodadas dos algoritmos criptográficos, algo inédito até então na área de criptografia.

- Um novo método para avaliar a segurança de algoritmos criptográficos em relação à criptoanálise diferencial, denominado *ColoreD*, é proposto. Com o *ColoreD*, ao invés de se considerar apenas diferenças binárias ("pretas e brancas"), passa a ser possível o uso de diferenças contínuas. Isso é possível a partir do uso das generalizações contínuas que permitem que se considere diferenças menores do que 1 bit. Adicionalmente, com o método *ColoreD*, propõe-se novas ferramentas tais como a Criptoanálise Diferencial Contínua (CDC). Esta ferramenta viabiliza a implementação de ataques de recuperação de chave sem a necessidade de redução do número de rodadas para algoritmos complexos. Para demonstrar a utilidade dessa proposta, utiliza-se o ferramental *ColoreD* para estudar e comparar os algoritmos AES e PRESENT, duas cifras de bloco bastante conhecidas. Tal análise, leva a conclusão de que o algoritmo AES é mais seguro do que o PRESENT quando se considera a criptoanálise diferencial. Em particular, demonstra-se que o algoritmo PRESENT necessitaria de ao menos 37 rounds para atingir a mesma margem de segurança do AES. Finalmente, aplicando-se o CDC, é proposto um ataque capaz de recuperar chave desses algoritmos, a partir do uso de suas generalizações contínuas e de pares de entradas com diferenças bem pequenas.

- Novas técnicas de criptoanálise contra algoritmos do tipo ARX são propostas. Primeiramente, uma nova forma de se gerar aproximações lineares é apresentada. Com tal técnica, demonstra-se ser possível encontrar aproximações lineares mais eficientes em cifras tipo ARX. Com tal técnica, propõe-se as primeiras aproximações lineares explicitamente derivadas para 3 e 4 rounds da cifra de fluxo ChaCha. Como consequência, novos ataques contra o ChaCha são apresentados, sendo possível reduzir a complexidade dos ataques para 2^{51} e 2^{224} bits de complexidade, contra 6 e 7 rodadas da cifra ChaCha, respectivamente. Adicionalmente, propõe-se uma nova técnica denominada Expansões Lineares Bidirecionais (BLE), capaz de aumentar a eficácia de *distinguishers* do tipo linear-diferencial. Usando a BLE, apresenta-se os primeiros *distinguishers* da literatura alcançando 7 e 8 rounds do algoritmo Salsa20 com complexidades de $2^{108.98}$ e $2^{215.62}$, respectivamente. Finalmente, demonstra-se que usando os novos diferenciais obtidos via BLE, é possível melhorar ataques de recuperação de chave do tipo *Probabilistic Neutral Bits* (PNB) contra 7 e 8 rodadas do algoritmo Salsa20, obtendo complexidades de $2^{122.63}$ e $2^{219.56}$, respectivamente.

- Novas cifras de fluxo são propostas. Primeiramente, demonstra-se que é possível aplicar uma alteração bastante simples no algoritmo ChaCha, apenas pela alteração dos parâmetros de rotações na função de quarto de round (QRF), tornando o ChaCha mais seguro contra todos os ataques conhecidos sem perda de performance. De fato, com tais mudanças, deixa de ser possível quebrar 7 rounds do ChaCha, restando apenas ataques contra 6 rounds. Na sequência, a cifra Forró é proposta. Demonstra-se que o algoritmo Forró é capaz de atingir segurança maior do que a do ChaCha mesmo aplicando uma menor quantidade de operações matemáticas. Assim, conclui-se que 5 rounds do Forró é tão seguro quanto 7 rounds do ChaCha e que o algoritmo Forró é mais eficiente quando implementado em diversos tipos de processadores.

Palavras-chave: Criptografia, ARX, Criptoanálise, Difusão, Segurança, ChaCha, Salsa, Speck, AES, PRESENT, Forró, Generalizações contínuas.

ABSTRACT

Title: Design, Diffusion, and Cryptanalysis of Symmetric Primitives
Author: Murilo Coutinho Silva
Supervisor: Rafael T. de Sousa Jr.
Co-Supervisor: Fábio Borges de Oliveira
Graduate Program in Telecommunications and Communication Networks
Brasília, November 25th, 2022

In this PhD thesis, we study and propose new cryptographic techniques and algorithms. The following results are achieved:

- We propose a new technique called Continuous Diffusion Analysis (CDA) that can be used to study, design, and compare of cryptographic algorithms. CDA allows us to generalize cryptographic algorithms by transforming the discrete bits into probabilities such that the algorithm is generalized into a continuous mathematical function. We propose three new metrics to measure the diffusion in this generalized continuous space, namely the Continuous Avalanche Factor, the Continuous Neutrality Measure, and the Diffusion Factor. In addition, we show that these measures can be used to analyze the diffusion of cryptographic algorithms, in particular, the Diffusion Factor can be used to compare the diffusion without the need of reducing the number of rounds or considering a small subset of bits.

- We propose a new framework, named *ColoreD*, to evaluate security against differential cryptanalysis. In the proposed framework, instead of considering only binary (black and white) differences, we allow the use of *Continuous Differences* (*ColoreD*), which is possible using of continuous generalizations of cryptographic algorithms, allowing us to use differences smaller than one bit. *ColoreD* incorporates not only continuous generalization of algorithms, but we also propose new theoretical tools such as the Continuous Differential Cryptanalysis (CDC). This tool provides us with a theoretical framework that allows us to mount key recovery attacks without the need of reducing the number of rounds. To showcase the usefulness of the new framework, we use *ColoreD* to study and compare AES and PRESENT ciphers. This analysis leads to the conclusion that AES is safer than PRESENT when considering differential cryptanalysis, and that PRESENT would need at least 37 rounds to achieve the same security margin of AES. Additionally, applying CDC to both AES and PRESENT we show that is possible to mount a key recovery to both algorithms when considering inputs with very small continuous differences.

- We propose new techniques to improve cryptanalysis against ARX ciphers. First, we present a new way to generate linear approximations, which can be used to find better linear approximations in ARX ciphers. Using this technique, we present the first explicitly derived linear approximations for 3 and 4 rounds of ChaCha and, as a consequence, it enables us to improve the recent attacks against ChaCha. More precisely, we our attacks have complexity of 2^{51} and 2^{224} against 6 and 7 rounds of ChaCha, respectively. Additionally, we propose a technique called Bidirectional Linear Expansions (BLE) to improve the efficacy of differential-linear distinguishers. Using the BLE, we propose the first differential-linear distinguishers ranging 7 and 8 rounds of Salsa20, with time complexities of $2^{108.98}$ and $2^{215.62}$, respectively. Additionally, we show that using the differentials obtained, it is possible to improved Probabilistic Neutral Bits (PNB) key-recovery attacks against 7 and 8 rounds of Salsa20, obtaining time complexities of $2^{122.63}$ and $2^{219.56}$, respectively.

- We propose the design of new stream ciphers. First, we show that a simple modification in the algorithm ChaCha, namely changing the rotation distances in the Quarter Round Function, makes it more secure against all the most effective known attacks without any loss in performance. In fact, we show that with these changes, it is only possible to break up to 6 rounds of ChaCha. Therefore, it would be no longer possible to break 7 rounds of ChaCha with the best-known attacks. Finally, we propose a new stream cipher called Forró. We show that Forró is able to achieve more security than Salsa and ChaCha using fewer arithmetic operations. We show that the security of 5 rounds of Forró is equivalent to 7 rounds of ChaCha and that Forró is faster when implemented in several different processors.

Keywords: Cryptography, ARX, Cryptanalysis, Diffusion, Security, ChaCha, Salsa, Speck, AES, PRESENT, Forró, Continuous Diffusion Analysis, Continuous Differential Cryptanalysis.

SUMMARY

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ARX	Addition Rotation XOR. 56
CDA	Continuous Diffusion Analysis. 56
DPA	Differential Probability of Addition. 56
PNB	Probabilistic Neutral Bits. 56
SAC	Strict Avalanche Criterion. 56
SPN	Substitution Permutation Network. 56

NOTATION

X	A 4×4 state matrix of ChaCha, Salsa, or Forró.
$X^{(0)}$	Initial state matrix of ChaCha, Salsa, or Forró.
$X^{(R)}$	State matrix after application of R round functions.
Z	Output of ChaCha, Salsa, or Forró, $Z = X^{(0)} + X^{(R)}$.
$\Delta X^{(R)}$	XOR difference of $X^{(R)}$ and $X'^{(R)}$. $\Delta X^{(R)} = X^{(R)} \oplus X'^{(R)}$.
$\Delta_{\mathcal{C}} x$	The continuous difference, $\Delta_{\mathcal{C}} x = x \oplus_{\mathcal{C}} x'$.
Δx	XOR difference of x and x'.
$\Delta x_i^{(R)}$	Difference $\Delta x_i^{(R)} = x_i^{(R)} \oplus x'^{(R)}_i$.
$\Delta x_{i,j}^{(R)}$	Difference $\Delta x_{i,j}^{(R)} = x_{i,j}^{(R)} \oplus x'^{(R)}_{i,j}$.
$\Pr(E)$	Probability of occurrence of an event E.
$\mathcal{B}$	The set $\{x \in \mathbb{R} : -1 \leq x \leq 1\}$.
$\mathcal{ID}$	Input difference.
$\mathcal{OD}$	Output difference.
$\oplus_{\mathcal{C}}$	The continuous XOR function, $\oplus_{\mathcal{C}} \overset{\sim}{\leftarrow} \oplus$, such that $a \oplus_{\mathcal{C}} b = -ab$, where $a, b \in \mathcal{B}$.
$\phi : \mathbb{F}_2^n \rightarrow \mathcal{B}^n$	Defined as $\phi(x) = (2x_0 - 1, ..., 2x_{n-1} - 1)$.
$\sigma : \mathcal{B}^n_{\neq 0} \rightarrow \mathbb{F}_2^n$	Defined as $\sigma(y) = \big(\frac{1}{2}(\mathrm{sgn}(y_0) + 1), ..., \frac{1}{2}(\mathrm{sgn}(y_{n-1}) + 1)\big)\big)$.
$\mathrm{sgn}(x)$	The sign function.
$\varepsilon_{(x_1 \oplus ... \oplus x_m)}$	Correlation of event $E = \{\Delta x_1 \oplus ... \oplus \Delta x_m = 0\}$.
$f_{\mathcal{C}} \overset{\sim}{\leftarrow} f$	Means the function $f_{\mathcal{C}}$ is a continuous generalization of the function f.
$x \ggg n$	Rotation of x by n bits to the left.
$x \lll n$	Rotation of x by n bits to the left.
$x \oplus y$	Bitwise XOR of x and y.
$x + y$	Addition of x and y modulo 2^{32}.
$x - y$	Subtraction of x and y modulo 2^{32}.
$x_{i,j}^{(R)}$	The j^{th} bit of i^{th} word of the state matrix $X^{(R)}$.
$x_i^{(R)}$	The i^{th} word of the state matrix $X^{(R)}$ (words arranged in row major).
$x_i^{(R)}[j_0, j_1, ..., j_t]$	The sum $x_{i,j_0}^{(R)} \oplus x_{i,j_1}^{(R)} \oplus \cdots \oplus x_{i,j_t}^{(R)}$.

1 INTRODUCTION

Our world is revolved in information. It was Claude Shannon, an engineer, mathematician and cryptologist, that defined the concept of information. Usually working alone in a small room at Bell Labs, located in an old mud-brick building on Wall Street, Shannon wrote his famous article entitled “A Mathematical Theory of Communication” [2]. In this work, Shannon mathematically defined information and coined the term *bit* as the basic and indivisible unit of information. The bit became like the meter, the liter, the second, a fundamental unit of measurement.

In 1949, just a year after creating Information Theory, Shannon published a work entitled “Communication Theory of Secrecy Systems” [3], in which he brilliantly uses Information Theory to mathematically define cryptography. Of course, before Shannon cryptography already existed. In fact, along with the first written records of human beings, rudimentary forms of cryptography have already begun to emerge. What Shannon did was to turn cryptography into a mathematical science, in particular an information science, one that was studied extensively in the past 4 decades.

In the modern computer, cryptography has found its true home. The information machine is the perfect encryption machine. Thus, cryptography was disseminated to the general public, being available to anyone who owns a computer. Today, cryptography is everywhere, even without knowing it, the vast majority of people use it daily when accessing the internet, sending messages to their friends or family, or carrying out electronic financial transactions.

Cryptography is an important and complex area of knowledge. In this thesis, we contribute to the advancement of the area by studying symmetric encryption algorithms, that uses the same key to encryption and decryption. Symmetric algorithms are typically fast, constructed through simple boolean and arithmetic operations to achieve the desired security.

A symmetric cryptographic algorithm needs to have several properties to be considered secure. Two of the most important of such properties were named *confusion* and *diffusion* by Claude Shannon [3] back in 1949. Shannon defined confusion as the capacity of an algorithm to create a very complex and involved relationship between the key and the ciphertext and diffusion as the property that the redundancy in the statistics of the plaintext is “dissipated” into the statistics of the ciphertext. These are, however, very abstract concepts and since then several metrics were proposed to try to measure the confusion and diffusion of algorithms [1, 4, 5].

Cryptographic Boolean functions [6] contain important properties that can be used, at least in theory, to access the confusion and diffusion of algorithms. In designing a cryptographic algorithm, we often need functions that satisfy requirements such as balance, high nonlinearity, high algebraic degree, and good avalanche characteristics.

Unfortunately, for modern ciphers, it is not possible to actually prove these properties for each output bit whereas the Boolean functions generated cannot be explicitly derived. Thus, these properties are estimated by randomly generating many inputs to the cipher and then changing input bits to see the behavior of the output bits empirically [1, 7]. However, using these kind of metrics is not actually useful to access the security of algorithms because even weak and broken ciphers tend to have good results in such empirical tests.

The absence of mathematical proofs for the security of symmetric algorithms means that cryptographers need to rely on practical evidences. More concretely, it is generally argued that confidence in the security of a particular algorithm only comes through time, after careful examination by several experts through the use of cryptanalytic technique.

Cryptanalysis is the art of breaking cryptography, a set of mathematical tools developed over the years to explore vulnerabilities in cryptographic algorithms. Through the advancement of cryptanalysis the quality of the algorithms have increased, as specialists learn how to develop faster and more secure algorithms.

In this thesis, we aim to contribute to the scientific advancement of cryptography. To do so, we divide this thesis in four parts. Part I is called *preliminaries* and presents an introduction to several of the topics that are explored in subsequent chapters. In particular, we review the cryptographic algorithms that are explored during the thesis and we present a short and simple introduction to cryptography and cryptanalysis.

In Part II, we propose new tools to evaluate the security of cryptographic algorithms. First, we propose a new technique, called Continuous Diffusion Analysis (CDA), which can be used to study the diffusion of cryptographic algorithms. The main idea of CDA is to generalize cryptographic operations allowing to express the individual bits as probabilities, effectively creating a continuous generalization of the algorithm itself. In this way, we can also generalize diffusion metrics and differential cryptanalysis, because we can change the input not only by flipping bits but also by inserting very small biases in a particular continuous bit. Moreover, we propose three new metrics to measure the diffusion in this generalized continuous space, namely the Continuous Avalanche Factor, the Continuous Neutrality Measure, and the Diffusion Factor. In particular, for the best of our knowledge, the Diffusion Factor is the first metric capable of measuring the diffusion of a modern secure cipher without considering a reduction with fewer rounds or a sample subset of input bits.

Part III is dedicated to cryptanalysis. We focus on algorithms designed using only the addition, XOR , and rotation operations (known as ARX ciphers). ARX-based design is a major building block of modern ciphers due to its efficiency in software, security, and simplicity of their design and implementation. ARX ciphers are heavily used in practice. For instance, in 2005, Bernstein proposed the stream cipher Salsa20 [8] as a contender to the eSTREAM [9], the ECRYPT Stream Cipher Project. As outlined by the author, Salsa20 is an ARX type family of algorithms which can be ran with several number of rounds, including the well known Salsa20/12 and Salsa20/8 versions. Latter, in 2008, Bernstein proposed some modifications to Salsa20 in order to provide better diffusion per round and higher resistance to cryptanalysis. These changes originated a new stream cipher, a variant which he called ChaCha [10]. Although Salsa20 was one of the winners of the eSTREAM competition, ChaCha has received much more attention through the years. Nowadays, we see the usage of this cipher in several projects and applications.

ChaCha, along with Poly1305 [11], is in one of the cipher suits of the new TLS 1.3 [12], which has been used by Google on both Chrome and Android. Not only has ChaCha been used in TLS but also in many other protocols such as SSH, Noise and S/MIME 4.0. In addition, the RFC 7634 proposes the use of ChaCha in IKE and IPsec. ChaCha has been used not only for encryption, but also as a pseudo-random number generator in any operating system running Linux kernel 4.8 or newer [13, 14]. Additionally, ChaCha has been used in several applications such as WireGuard (VPN) (see [15] for a huge list of applications, protocols and libraries using ChaCha).

In this thesis, we significantly improve the cryptanalysis against both Salsa and ChaCha. More precisely, we propose a new way to generate linear approximations, which can be used to find better linear approximations in ARX ciphers. Using these ideas we were able to improve differential-linear distinguishers and also key recovery attacks.

Finally, in Part IV, we use all the tools developed and experience gained to propose new stream ciphers. First, we show that is possible to improve the security of ChaCha by simply changing its rotation distances. Moreover, we propose a fundamentally new design that is more resistant to all known attacks while being faster in constrained devices, resulting in a new stream cipher that we named Forró.

Part I

Preliminaries and Previous Work

2 SYMMETRIC ENCRYPTION

In this thesis, we focus on symmetric cryptography. To understand the concept of symmetric cryptography, it is common to image two entities, named Alice and Bob, that share a secret key k. Alice wants to transmit a message m to Bob over a network while maintaining the secrecy of m in the presence of an eavesdropping adversary. Besides transmitting a message over a network, these same techniques allow Alice to store a file on a disk so that no one else with access to the disk can read the file, but Alice herself can read the file at a later time.

The basic mechanism for encrypting a message using a shared secret key is called a cipher (or encryption scheme). Here, we review a slightly simplified notion of a cipher, which we call a Shannon cipher. A Shannon cipher is a pair $\mathcal{E} = (E, D)$ of functions.

- The function E (the encryption function) takes as input a key k and a message m (also called a plaintext), and produces as output a ciphertext c. That is,

  $$c = E(k, m),$$

  and we say that c is the encryption of m under k.

- The function D (the decryption function) takes as input a key k and a ciphertext c, and produces a message m. That is,

  $$m = D(k, c),$$

  and we say that m is the decryption of c under k.

- We require that decryption inverts encryption; that is, the cipher must satisfy the following correctness property: for all keys k and all messages m, we have

  $$D(k, E(k, m)) = m.$$

More formally, we define $\mathcal{K}$ as the set of all keys (the key space), $\mathcal{M}$ as the set of all messages (the message space), and $\mathcal{C}$ as the set of all ciphertexts (the ciphertext space). With this notation, we can write:

$$E : \mathcal{K} \times \mathcal{M} \to \mathcal{C},$$
$$D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

Also, we shall say that $\mathcal{E}$ is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$.

Therefore, the idea is that Alice and Bob must somehow agree in advance on a key

$k \in \mathcal{K}$. Then, when Alice wants to send a message $m \in \mathcal{M}$ to Bob, she encrypts m under k, obtaining the ciphertext $c = E(k, m) \in \mathcal{C}$, and then sends c to Bob via any communication network. Upon receiving c, Bob decrypts c under k, and the correctness property ensures that $D(k, c)$ is the same as Alice's original message m.

Here, the goal is that $\mathcal{E}$ is secure in the sense that an adversary that obtains the ciphertext c cannot obtain relevant information about m. Next we define the basic notions of security in cryptography.

2.1 DEFINING SECURITY

So far, we defined what is a Shannon cipher. But it remains to understand what is a secure cipher. In the following we define the concept of perfect security:

Definition 2.1 (Perfect security). Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Consider a probabilistic experiment in which the random variable $\mathbf{k}$ is uniformly distributed over $\mathcal{K}$. If for all $m_0, m_1 \in \mathcal{M}$, and all $c \in \mathcal{C}$, we have

$$\Pr\left[E\left(\mathbf{k}, m_0\right) = c\right] = \Pr\left[E\left(\mathbf{k}, m_1\right) = c\right], \tag{2.1}$$

then we say that $\mathcal{E}$ is a perfectly secure Shannon cipher.

Let's try to understand better Definition 2.1. First, notice that we require k to be uniformly distributed. Intuitively this makes sense because if we have keys that appear with higher probability than the attacker can explore this fact by testing such keys first. Additionally, Definition 2.1 says that the encryption of any message has exactly the same probability. If this is true, than given c the adversary will not find any good guess for the message m.

Definition 2.1 makes sense. However, how do we achieve such encryption scheme in practice? Unfortunately, to achieve perfectly security we need keys that at least as big as the message (see [16] for a proof of this claim). This turns out to be unpractical for almost all situations. Therefore, we normally work with a weaker notion of security called **semantic security**. Precisely defining semantic security is not required for this thesis, but it suffices to say that we will no longer require Eq. (2.1) to hold. Instead, we require the probabilities to be extremely close but not necessarily equal.

2.2 BLOCK CIPHER AND STREAM CIPHERS

Achieving semantic security is easier in practice as we no long require the keys to be the same size of the message. Nevertheless, it should be infeasible to the adversary to guess the key k. For that, keys are usually of some fixed length, for example, 32-byte (i.e., 256-bit). In this case, we have a total of 2^{256} keys, a extremely big number. This avoids the possibility of the adversary testing all possible keys (this is call an exhaustion attack).

In practice, we have two types of symmetric ciphers:

- A **stream cipher** expands a fixed sized key into a key-stream with size equal to the message being encrypted. Here, messages, and ciphertexts are all L-bit strings. Also, we have a short, ℓ-bit key k, where ℓ is much smaller than L. The string k is stretched using some efficient, deterministic algorithm G that maps ℓ-bit strings to L-bit strings. For $k \in \{0,1\}^\ell$ and $m, c \in \{0,1\}^L$, encryption and decryption are defined as follows:

$$E(k,m) := G(k) \oplus m \quad \text{and} \quad D(k,c) := G(k) \oplus c \tag{2.2}$$

- A **block cipher** is a cipher $\mathcal{E} = (E, D)$ whose message space and ciphertext space are the same (finite) set $\mathcal{X}$. If the key space of $\mathcal{E}$ is $\mathcal{K}$, we say that $\mathcal{E}$ is a block cipher defined over $(\mathcal{K}, \mathcal{X})$. We call an element $x \in \mathcal{X}$ a data block, and refer to $\mathcal{X}$ as the data block space of $\mathcal{E}$. In practice, block ciphers are used together with the so called **modes of operation** to encrypt messages of arbitrary length.

In the next section we review some important stream ciphers and block ciphers that are used in practice and explored during this thesis.

2.3 ALGORITHMS

In this section, we review some important algorithms that we will be using throughout this thesis.

2.3.1 Salsa

The stream cipher Salsa20 was proposed by Bernstein [8] to the *eSTREAM* competition and consists of a series of ARX (addition, rotation, and XOR) operations on 32-bit words, being highly efficient in software and hardware. Salsa20 operates on a state of 64 bytes, organized as a 4×4 matrix with 32-bit integers, initialized with a 256-bit key $k_0, k_1, ..., k_7$, a 64-bit nonce v_0, v_1 and a 64-bit counter t_0, t_1 (we may also refer to the nonce and counter

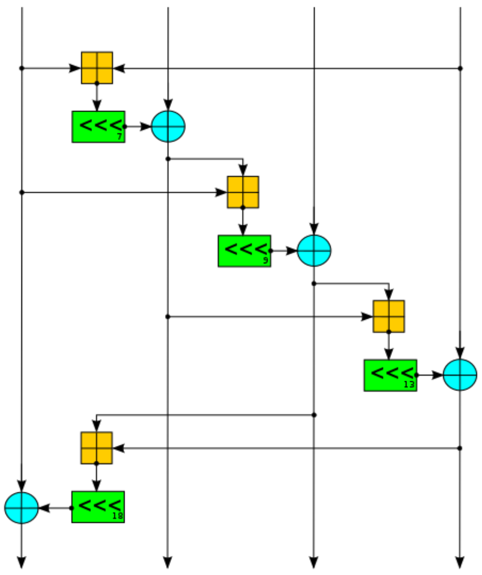


Figure 2.1 – Graphical representation of the QRF of Salsa.

words as IV words), and 4 constants $c_0 = 0\text{x}61707865$, $c_1 = 0\text{x}3320646e$, $c_2 = 0\text{x}79622d32$ and $c_3 = 0\text{x}6b206574$. For Salsa20, we have the following initial state matrix:

$$X^{(0)} = \begin{pmatrix} x_0^{(0)} & x_1^{(0)} & x_2^{(0)} & x_3^{(0)} \\ x_4^{(0)} & x_5^{(0)} & x_6^{(0)} & x_7^{(0)} \\ x_8^{(0)} & x_9^{(0)} & x_{10}^{(0)} & x_{11}^{(0)} \\ x_{12}^{(0)} & x_{13}^{(0)} & x_{14}^{(0)} & x_{15}^{(0)} \end{pmatrix} =$$

$$= \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}. \tag{2.3}$$

The state matrix is modified in each round by a *Quarter Round Function* (QRF), named $QR_{salsa}^{(r)}(a, b, c, d)$, which receives and updates 4 integers in the following way see also Figure 2.1:

$$\begin{aligned} x_b^{(r)} &= x_b^{(r-1)} \oplus ((x_a^{(r-1)} + x_d^{(r-1)}) \lll 7) \\ x_c^{(r)} &= x_c^{(r-1)} \oplus ((x_b^{(r-1)} + x_a^{(r-1)}) \lll 9) \\ x_d^{(r)} &= x_d^{(r-1)} \oplus ((x_c^{(r-1)} + x_b^{(r-1)}) \lll 13) \\ x_a^{(r)} &= x_a^{(r-1)} \oplus ((x_d^{(r-1)} + x_c^{(r-1)}) \lll 18) \end{aligned} \tag{2.4}$$

One round of Salsa20 is defined as 4 applications of the QRF. There is a difference,

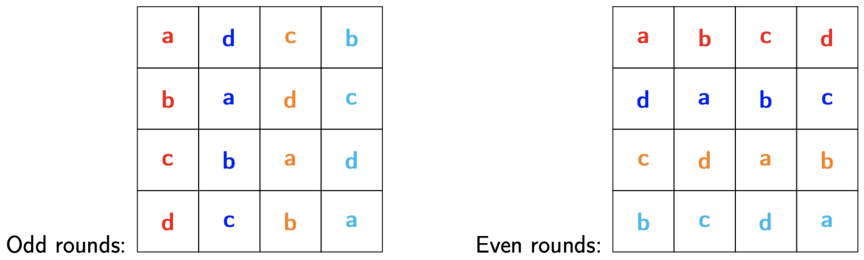


Figure 2.2 – Order of application of the QRF function for Salsa in each round

however, between odd and even rounds. Thus, for odd rounds, when $m \in \{1, 3, 5, 7, ...\}$, $X^{(m)}$ is defined from $X^{(m-1)}$, from

$$\begin{aligned}
\left(x_0^{(m)}, x_4^{(m)}, x_8^{(m)}, x_{12}^{(m)}\right) &= QR_{salsa}\left(x_0^{(m-1)}, x_4^{(m-1)}, x_8^{(m-1)}, x_{12}^{(m-1)}\right)\\
\left(x_5^{(m)}, x_9^{(m)}, x_{13}^{(m)}, x_1^{(m)}\right) &= QR_{salsa}\left(x_5^{(m-1)}, x_9^{(m-1)}, x_{13}^{(m-1)}, x_1^{(m-1)}\right)\\
\left(x_{10}^{(m)}, x_{14}^{(m)}, x_2^{(m)}, x_6^{(m)}\right) &= QR_{salsa}\left(x_{10}^{(m-1)}, x_{14}^{(m-1)}, x_2^{(m-1)}, x_6^{(m-1)}\right)\\
\left(x_{15}^{(m)}, x_3^{(m)}, x_7^{(m)}, x_{11}^{(m)}\right) &= QR_{salsa}\left(x_{15}^{(m-1)}, x_3^{(m-1)}, x_7^{(m-1)}, x_{11}^{(m-1)}\right)
\end{aligned},$$

and for even rounds $m \in \{2, 4, 6, 8, ...\}$ from

$$\begin{aligned}
\left(x_0^{(m)}, x_1^{(m)}, x_2^{(m)}, x_3^{(m)}\right) &= QR_{salsa}\left(x_0^{(m-1)}, x_1^{(m-1)}, x_2^{(m-1)}, x_3^{(m-1)}\right)\\
\left(x_5^{(m)}, x_6^{(m)}, x_7^{(m)}, x_4^{(m)}\right) &= QR_{salsa}\left(x_5^{(m-1)}, x_6^{(m-1)}, x_7^{(m-1)}, x_4^{(m-1)}\right)\\
\left(x_{10}^{(m)}, x_{11}^{(m)}, x_8^{(m)}, x_9^{(m)}\right) &= QR_{salsa}\left(x_{10}^{(m-1)}, x_{11}^{(m-1)}, x_8^{(m-1)}, x_9^{(m-1)}\right)\\
\left(x_{15}^{(m)}, x_{12}^{(m)}, x_{13}^{(m)}, x_{14}^{(m)}\right) &= QR_{salsa}\left(x_{15}^{(m-1)}, x_{12}^{(m-1)}, x_{13}^{(m-1)}, x_{14}^{(m-1)}\right)
\end{aligned}$$

The output of Salsa20/R is then defined as the sum of the initial state with the state obtained after R rounds of operations $Z = X + X^{(R)}$. One should note that is possible to parallelize each application of the QRF on each round and that each round is reversible, hence we can compute $X^{(m-1)}$ from $X^{(m)}$. For more information on the design of Salsa20, we refer to [8].

The cryptanalysis of Salsa20 was introduced by Crowley [17] in 2005. Crowley developed a differential attack against Salsa20/5, namely the 5-round version of Salsa20, and received the 1000 dolar prize offered by Bernstein for the most interesting Salsa20 cryptanalysis in that year. In 2006, Fischer et al. [18] improved the attack against Salsa20/5 and presented their attack against Salsa20/6. Probably the most important cryptanalysis in this regard was proposed by Aumasson et al. at FSE 2008 [19] with the introduction of Probabilistic Neutral Bits (PNBs), showing attacks against Salsa20/7, Salsa20/8. After that, several authors proposed small enhancements on the attack of Aumasson et al. The work by

Shi et al. [20] introduced the concept of Column Chaining Distinguisher (CCD) to achieve some incremental advancements over [19] for both Salsa.

Maitra, Paul and Meier [21] studied an interesting observation regarding round reversal of Salsa, but no significant cryptanalytic improvement could be obtained using this method. Maitra [22] used a technique of Chosen IVs to obtain certain improvements over existing results. Dey and Sarkar [23] showed how to choose values for the PNB to further improve the attack. In a paper presented in FSE 2017, Choudhuri and Maitra [24] significantly improved the attacks by considering the mathematical structure of Salsa in order to find differential characteristics with much higher correlations.

2.3.2 ChaCha

The stream cipher Salsa20 was proposed by Bernstein [8] to the *eSTREAM* competition and later Bernstein proposed ChaCha [10] as an improvement of Salsa20. ChaCha consists of a series of ARX operations on 32-bit words, being highly efficient in software and hardware. Each round of ChaCha has a total of 16 bitwise XOR, 16 addition modulo 2^{32} and 16 constant-distance rotations.

ChaCha operates on a state of 64 bytes, organized as a 4×4 matrix with 32-bit integers, initialized with a 256-bit key $k_0, k_1, ..., k_7$, a 64-bit nonce v_0, v_1 and a 64-bit counter t_0, t_1 (we may also refer to the nonce and counter words as IV words), and 4 constants $c_0 = 0\text{x}61707865$, $c_1 = 0\text{x}3320646e$, $c_2 = 0\text{x}79622d32$ and $c_3 = 0\text{x}6b206574$, the same as in Salsa. For ChaCha, we have the following initial state matrix:

$$X^{(0)} = \begin{pmatrix} x_0^{(0)} & x_1^{(0)} & x_2^{(0)} & x_3^{(0)} \\ x_4^{(0)} & x_5^{(0)} & x_6^{(0)} & x_7^{(0)} \\ x_8^{(0)} & x_9^{(0)} & x_{10}^{(0)} & x_{11}^{(0)} \\ x_{12}^{(0)} & x_{13}^{(0)} & x_{14}^{(0)} & x_{15}^{(0)} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & t_1 & v_0 & v_1 \end{pmatrix}. \tag{2.5}$$

The state matrix is modified in each round by a *Quarter Round Function* (QRF), denoted by $QR\left(x_a^{(r-1)}, x_b^{(r-1)}, x_c^{(r-1)}, x_d^{(r-1)}\right)$, which receives and updates 4 integers in the following way:

$$\begin{array}{rclrcl} x_{a'}^{(r-1)} & = & x_a^{(r-1)} + x_b^{(r-1)}; & x_{d'}^{(r-1)} & = & (x_d^{(r-1)} \oplus x_{a'}^{(r-1)}) \lll 16; \\ x_{c'}^{(r-1)} & = & x_c^{(r-1)} + x_{d'}^{(r-1)}; & x_{b'}^{(r-1)} & = & (x_b^{(r-1)} \oplus x_{c'}^{(r-1)}) \lll 12; \\ x_a^{(r)} & = & x_{a'}^{(r-1)} + x_{b'}^{(r-1)}; & x_d^{(r)} & = & (x_{d'}^{(r-1)} \oplus x_a^{(r)}) \lll 8; \\ x_c^{(r)} & = & x_{c'}^{(r-1)} + x_d^{(r)}; & x_b^{(r)} & = & (x_{b'}^{(r-1)} \oplus x_c^{(r)}) \lll 7; \end{array} \tag{2.6}$$

Compared to Salsa, all integers are updated more often improving the diffusion. Another important change when compared to Salsa is the order of application of the QRF, which is

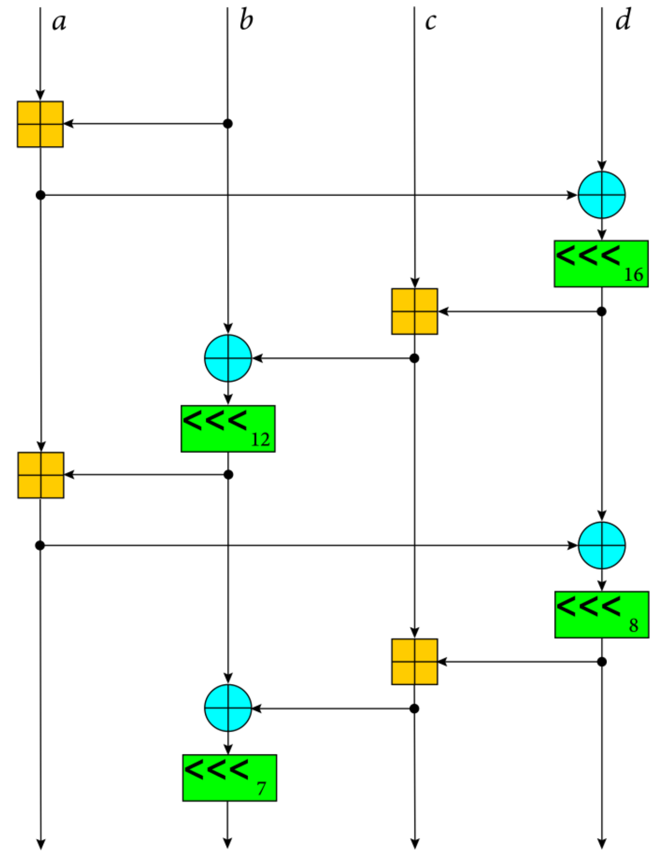


Figure 2.3 – Graphical representation of the QRF of ChaCha.

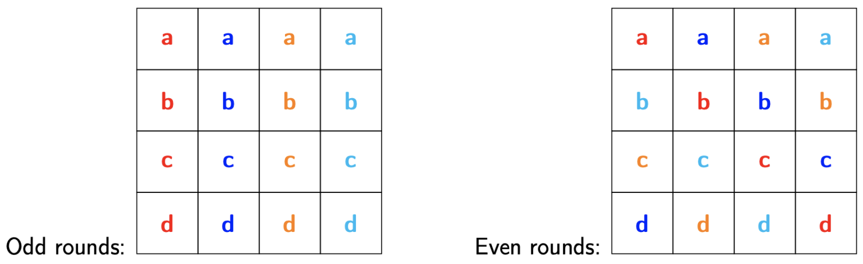


Figure 2.4 – Order of application of the QRF function for ChaCha in each round

now applied first on the columns and then on the diagonals of the state matrix. This change allows extra optimization on certain platforms and, as we demonstrate in this work, also improve the diffusion of ChaCha when compared to Salsa.

One round of ChaCha is defined as 4 applications of the QRF. There is, however, a difference between odd and even rounds. For odd rounds, i.e., $r \in \{1, 3, 5, 7, ...\}$, $X^{(r)}$ is obtained from $X^{(r-1)}$ by applying

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

Contrarily, for even rounds, i.e., $r \in \{2, 4, 6, 8, , ...\}$, $X^{(r)}$ is calculated from $X^{(r-1)}$ by applying

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

The output of ChaCha20/R is then defined as the sum of the initial state with the state after R rounds $Z = X^{(0)} + X^{(R)}$. One should note that it is possible to parallelize each application of the QRF on each round, which are reversible. Hence, we can compute $X^{(r-1)}$ from $X^{(r)}$. For more information on ChaCha, we refer to [10].

The cryptanalysis of ChaCha followed the techniques applied against Salsa. Aumasson et al. at FSE 2008 [19] used PNBs to attack ChaCha20/6 and ChaCha20/7. The techniques of Shi [20], Choudhuri and Maitra [24] also improved attacks against ChaCha.

Recently, several works presented improvements in attack against ChaCha. First, Coutinho and Souza [25] proposed new multi-bit differentials using the mathematical framework of Choudhuri and Maitra. In Crypto 2020, Beierle et al. [26] proposed improvements to the framework of differential-linear cryptanalysis against ARX-based designs and further improved the attacks against ChaCha. At Eurocrypt 2021, Coutinho and Souza [27] developed a new technique to expand linear trails improving the attack against ChaCha even further. However, these new techniques were not used against Salsa. Finally, at Eurocrypt 2022 Dey et al. [28] improved the analysis of the PNB construction and key recovery attacks against ChaCha.

2.3.3 Speck

Speck is a family of lightweight block ciphers publicly released by the National Security Agency (NSA) [29]. Speck has been optimized for performance in software implementations and supports a variety of block and key sizes. A block is always two words, but the words may be 16, 24, 32, 48, or 64 bits in size. The corresponding key is 2, 3, or 4 words.

The round function $F : \mathbb{F}_2^k \times \mathbb{F}_2^{2k} \to \mathbb{F}_2^{2k}$ of Speck is very simple. It takes as input a k-bit subkey K and a cipher state consisting of two k-bit words (L_i, R_i) and produces from this the next round state (L_{i+1}, R_{i+1}) as follows:

$$L_{i+1} = ((L_i \gg \alpha) + R_i) \oplus K$$

$$R_{i+1} = (R_i \ll \beta) \oplus L_{i+1}$$

where α, β are constants specific to each member of the Speck cipher family ($\alpha = 7$, $\beta = 2$ for Speck32/64 and $\alpha = 8$, $\beta = 3$ for other variants).

The round function is applied a fixed number of times (for 22 rounds in the case of Speck32/64) to produce from the plaintext input the ciphertext output. The subkeys for each round are generated from a master key by a non-linear key schedule that uses as its main building block also this round function. Some details of the key schedule differ between different versions of Speck due to the different number of words in the master key. For more information about Speck, we refer to [29].

In this paper we will not focus on cryptanalysis of Speck.

2.3.4 AES

AES is a variant of the Rijndael block cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits,

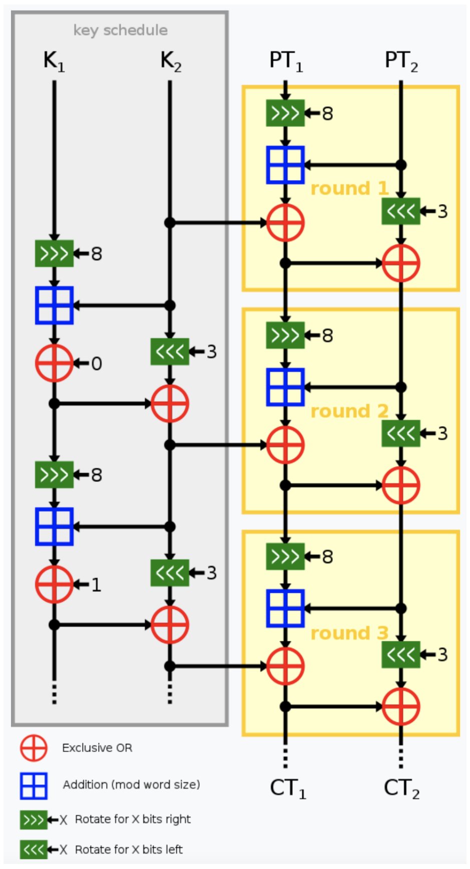


Figure 2.5 – Diagram of the Speck cipher.

but three different key lengths: 128, 192 and 256 bits. AES has been adopted by the U.S. government.

AES is a substitution-permutation network and was established as a standard by the (NIST) in 2001. AES has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. AES operates on a 4×4 state matrix as follows

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}.$$

The number of rounds of AES varies depending on the size of the key, being 10, 12 or 14 rounds for key sizes of 128, 192, or 256 bits, respectively. Each round consists of several processing steps, using different key streams defined by the key schedule. As a high-level description of the algorithm, we say that each round has 4 steps:

1. AddRoundKey: each byte of the state is combined with a byte of the round key using bitwise XOR.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 2.1 – AES S-box. The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value 9a is converted into b8.

2. SubBytes: a non-linear substitution step, where each byte is replaced with another according to the S-box presented in Table 2.1. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over $GF(2^8)$, known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points and also any opposite fixed points.

3. ShiftRows: a transposition step, where the last three rows of the state are shifted cyclically a certain number of steps. More precisely, each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. The importance of this step is to avoid the columns being encrypted independently, in which case AES would degenerate into four independent block ciphers.

4. MixColumns: a linear mixing operation that operates on the columns of the state, combining the four bytes in each column, where each input byte affects all four output bytes. During this operation, each column is transformed by a multiplication against a fixed matrix, creating diffusion in the algorithm. All operations are handled in $GF(2^8)$.

The key schedule of AES is quite simple and can be represented by Figure 2.6. Note that the key schedule is invertible. For more information on AES, we refer to [30].

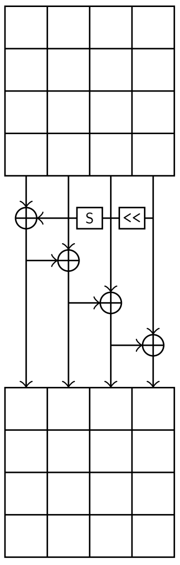


Figure 2.6 – The key schedule for AES-128. The bytes of a 4×4 matrix are updated using a S-box, XOR, and rotation.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	b	9	0	A	D	3	E	F	8	4	7	1	2

Table 2.2 – PRESENT S-box.

2.3.5 PRESENT

PRESENT is a block cipher introduced by Bogdanov et al. [31] in 2007. It has a block size of 64 bits, and the key size can be 80 bits or 128 bits. The non-linear layer is based on a single 4-bit S-box (see Table 2.2), which was designed with hardware optimizations in mind. PRESENT is intended to be used in situations where low-power consumption and high chip efficiency is desired. PRESENT is designed as classical Substitution Permutation Network (SPN), alternating an add round key, a S-Box, and a permutation layer, as in Figure 2.7. The key schedule is also very simple, using the S-Box and a round counter to derive the round keys, as in Figure 2.8. Note that knowing the full state the key schedule is invertible.

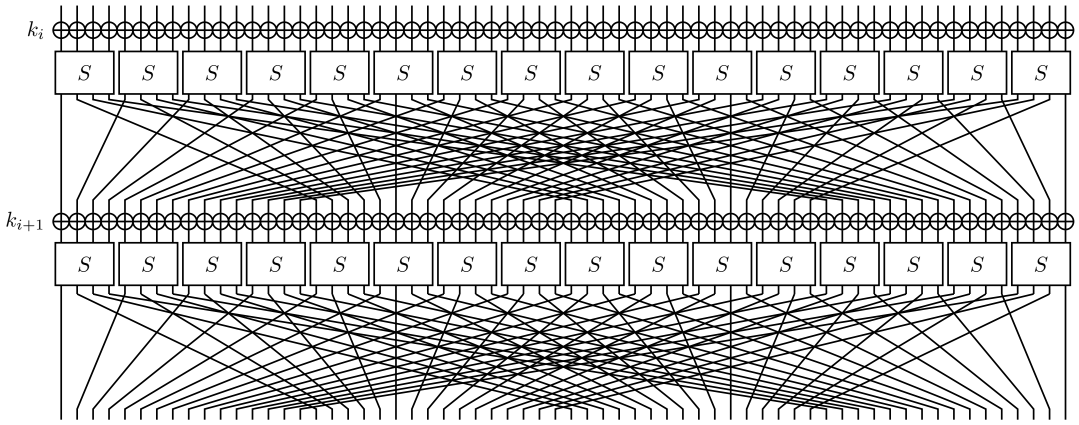


Figure 2.7 – The block cipher PRESENT.

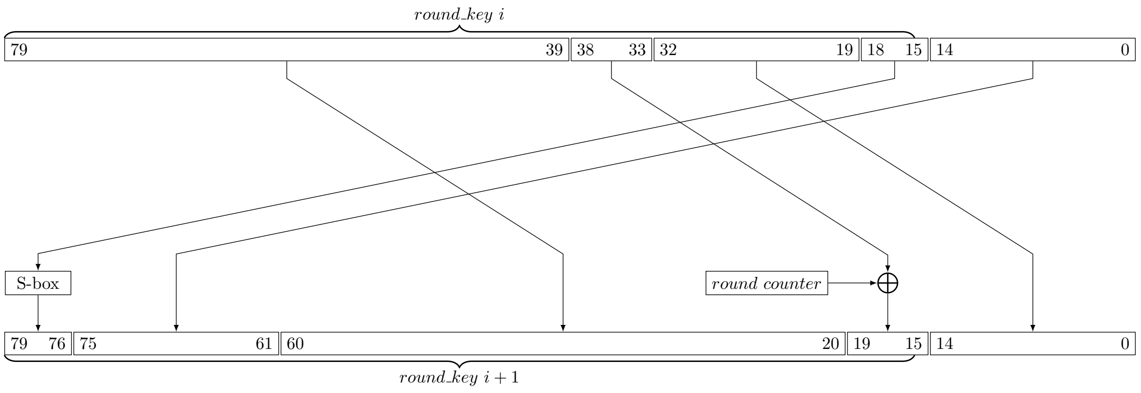


Figure 2.8 – Key schedule of PRESENT.

3 CRYPTANALYSIS

In this chapter, we review some important cryptanalysis techniques that are used in this thesis. More precisely, in Section 3.1, we review Differential Cryptanalysis, and in Section 3.3, we review Differential-Linear Cryptanalysis. In Section 3.4, we review the technique of Probabilistic Neutral Bits (PNBs).

3.1 DIFFERENTIAL CRYPTANALYSIS

In differential cryptanalysis [32], the attacker is interested in identifying and exploiting non-uniformity in occurrences of plaintext and ciphertext differences. Let x and y be two elements of a set $\mathcal{A}$, then we define the *difference* between x and y as $\Delta(x, y) = x \otimes y^{-1}$, where y^{-1} is the inverse of y with respect to some operation $\otimes$ on the set $\mathcal{A}$. Now given a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, we define the *differential* as the pair (δ, Δ) of an input difference $\delta \in \mathbb{F}_2^n$ and an output difference $\Delta \in \mathbb{F}_2^m$, i.e.,

$$F(x) \otimes F^{-1}(x \otimes \delta^{-1}) = \Delta.$$

Since δ and Δ have a finite number of possible values, we can define the *difference distribution table* of F as the matrix $A = \{A_{i,j}\}$, where

$$A_{i,j} = |\{x \in \mathbb{F}_2^n : F(x) \otimes F^{-1}(x \otimes i^{-1}) = j\}|.$$

In the specific case where we use XOR as the difference, we have

$$A_{i,j} = |\{x \in \mathbb{F}_2^n : F(x) \oplus F(x \oplus i) = j\}|. \tag{3.1}$$

From the difference distribution table, we can compute the probability of the differential (δ, Δ) as

$$\Pr[\delta \xrightarrow{F} \Delta] = \frac{A_{\delta,\Delta}}{2^{-n}}.$$

In general, we wish to find differentials with high probability to mount statistical attacks. To finding these differentials is often easier to divide the cipher in multiple components or steps $(c_0, c_1, ..., c_s)$ and consider the so called *differential characteristics*.

DEFINITION 3.1 An s-round **differential characteristic** is a series of differences, notated as an $(s+1)-$tuple $(\alpha_0, ..., \alpha_s)$, where α_i is the anticipated value of Δc_i for the characteristic, i.e., the difference between the values of the partially encrypted messages where α_0 is the chosen value of the message difference $\Delta m = \Delta c_0$.

The probability of an s-round characteristic can be computed from the conditional probability that $\Delta c_i = \alpha_i$ is the observed difference after i rounds given that $\Delta c_{i-1} = \alpha_{i-1}$ is the difference after $i-1$ rounds for $1, ..., s$. More formally, the probability of a characteristic is given by

$$\Pr_{\mathcal{M},\mathcal{K}}\left(\Delta c_s = \alpha_s, \Delta c_{s-1} = \alpha_{s-1}, \ldots, \Delta c_1 = \alpha_1 \mid \Delta m = \alpha_0\right),$$

where the probability is taken over all choices of the plaintext and the key. This probability can be hard to calculate. However, for certain ciphers it can be calculated from the probabilities of single-round characteristics. To do so, we note that a sequence of stochastic variables $v_0, v_1, \ldots, v_r$ is a Markov chain if $\Pr\left(v_{i+1} = \beta_{i+1} \mid v_i = \beta_i, \ldots, v_0 = \beta_0\right) = \Pr\left(v_{i+1} = \beta_{i+1} \mid v_i = \beta_i\right)$ for $0 \leq i < r-1$. If $\Pr\left(v_{i+1} = \beta \mid v_i = \alpha\right)$ is independent of i for all α and β, then a Markov chain is called *homogeneous*.

Thus, we speak of a Markov chain if the probabilities of the individual variables are independent of each other and of a homogeneous chain if the probability distribution is the same for all variables. This leads to the following definition:

DEFINITION 3.2 (Markov Cipher) An iterated cipher is called a *Markov cipher* if there is an operation $\otimes$ defining Δ such that $\Pr\left(\Delta c_1 = \beta \mid \Delta c_0 = \alpha, c_0 = \gamma\right)$ is independent of γ for all α and β when the round key k is chosen uniformly at random.

Thus, for a Markov cipher with independent round keys, the probability of an s-round characteristic is the product of the probabilities of the s one-round characteristics or, more formally,

$$\Pr\left(\Delta c_s = \alpha_s, \Delta c_{s-1} = \alpha_{s-1}, \ldots, \Delta c_1 = \alpha_1 \mid \Delta m = \alpha_0\right) = \prod_{i=1}^{s} \Pr\left(\Delta c_i = \alpha_i \mid \Delta c_{i-1} = \alpha_{i-1}\right).$$

Several ciphers, such as DES, PRESENT, and AES with independent subkeys are Markov ciphers when the notion of difference is the XOR operation. For more information about differential cryptanalysis, we refer to [32, 33, 34].

3.2 LINEAR CRYPTANALYSIS

3.2.1 The basics

After differential cryptanalysis, linear cryptanalysis provides the most important general technique for analyzing symmetric primitives. The development of linear cryptanalysis and its application to DES is due to Matsui [35]. However, linear cryptanalytic techniques had been used earlier in an attack on FEAL-4 by Tardy-Corfdir and Gilbert [36]. Interestingly, apart from a few enhancements linear cryptanalysis has evolved little since the early systematic treatment of Matsui. As an attack, it seems to be intrinsically less versatile than differential cryptanalysis.

Linear cryptanalysis is a known plaintext attack in which the attacker exploits probabilistic linear relations between bits of the plaintext m, the ciphertext c and the key k. Mathematically, we can write that

$$(m \cdot \alpha) \oplus (c \cdot \beta) = (k \cdot \gamma), \tag{3.2}$$

where α, β, and γ denote linear *masks* applied through the dot product $\cdot$. In other words, the masks represent the selection of particular bits of the plaintext, ciphertext, or key.

Generally, Eq. (3.2) represents and approximation, in the sense that the equality occurs with a probability ρ. It s useful, however, to work with the correlation given by $\varepsilon = 2\rho - 1$ when discussing cryptanalysis.

Let us consider how to find relations for individual rounds in an iterated cipher. It is clear that for linear and affine functions linear relations can be found that hold with probability 1 or 0. The important step in linear cryptanalysis is to establish such relations through the non-linear components of a cipher. For SPNs, such as AES and PRESENT, it is possible to compute so-called linear approximation tables for the different non-linear components. For the AES and PRESENT these are the S-boxes, and such linear approximation tables contain all possible input and output masks together with the corresponding probability that the parity of input approximation is preserved in the output approximation. Such a table for a function f can be presented as a matrix $A = \{A_{\alpha,\beta}\}$, where

$$A_{\alpha,\beta} = |\{x \in \mathrm{Dom}(f) : (x \cdot \alpha) = (f(x) \cdot \beta)\}|. \ |$$

We have that the bit $x \cdot \alpha$ equals the bit $f(x) \cdot \beta$ with probability

$$p_{\alpha,\beta} = \frac{A_{\alpha,\beta}}{|\,\mathrm{Dom}(h)|},$$

and the correlation the correlation c is given by $\varepsilon = 2p_{\alpha,\beta} - 1$.

It is possible to have ciphers where the range or the domain of the non-linear function is too large to allow a complete linear approximation table to be constructed (for example in ARX ciphers). In such circumstances, a cryptanalyst would likely restrict himself to searching over some useful mask values or some other techniques.

The first stages in a linear cryptanalytic attack are used to establish linear relations for individual components. These are then joined to form approximations to a round which, in turn, is joined to give approximations over more rounds. Suppose c_i denotes the partially encrypted text after i rounds of encryption for $i = 1$ to r for an r-round block cipher. Further suppose that the per-round relations

$$(c_{i-1} \cdot \alpha_{i-1}) \oplus (c_i \cdot \alpha_i) = (k_i \cdot \gamma_i)$$

hold with probability p_i, where the probability is taken over all possible inputs to the round and for fixed values of the subkeys k_i. By combining these relations with exclusive-or, one obtains an expression over s rounds,

$$(m \cdot \alpha_0) \oplus (c_s \cdot \alpha_s) = \sum_{i=1}^{s} (k_i \cdot \gamma_i)$$

that holds with some probability p when averaged over all plaintexts. The typical approach to estimating p is to use what is often called the piling-up lemma, a technique described by Tardy-Corfdir and Gilbert [36] and Matsui [35].

Let Z_i, for $1 \le i \le m$, be m independent random variables, which take the values $\{0, 1\}$, and further suppose that each random variable takes the value 0 with correlation ε_i. Thus, we have that

$$\Pr(Z_1 \oplus Z_2 \oplus ... \oplus Z_m = 0) = \frac{1}{2}\left(1 + \prod_{i=1}^{m} \varepsilon_i\right). \tag{3.3}$$

This goes some way to explaining the attractiveness of using correlations instead of probabilities since correlations can simply be multiplied together.

DEFINITION 3.3 An s-round **linear characteristic** is a series of masks defined as an $(s+1)$-tuple $(\alpha_0, \alpha_1, ..., \alpha_{s-1}, \alpha_s)$ with probability ρ, where α_i is the mask used in the i-th round. The characteristic predicts for $i = 1, ..., s$ that the sum of certain bits of the input to the i-th round will equal to the sum of certain bits in the outputs of the i-th round.

Most of the linear attacks reported in the literature make use of iterative characteristics. For an iterated block cipher, an s-round iterative linear characteristic is an $(s+1)$-tuple $(\alpha_0, \alpha_1, ..., \alpha_{s-1}, \alpha_s)$. Their usefulness can easily be seen when we note that an s-round iterative linear characteristic can always be used to build a t-round characteristic for any integer $t \ge s$.

For a Markov, cipher we can be sure that if the cipher has independent round keys, then the correlation of a multiple-round linear approximation, when taken over all values of the round keys, can be calculated from the one-round correlations using the piling-up lemma. The probability of a linear characteristic is calculated as an average over all keys. For more information about linear cryptanalysis, we refer to [34].

3.2.2 Mathematical Framework

Behind linear cryptanalysis, there is a very rich theoretical framework that can be explored. In particular, the use of the Fast Fourier transform (FFT) is especially useful, as there are extremely fast algorithms that use the FFT to compute the correlations of linear relationships [37, 38]. In this section, we introduce this mathematical formalism that will be used in the following sections.

3.2.2.1 Boolean Functions

The theory that we explore is mainly based on the theory of Boolean functions [6, 38].

DEFINITION 3.4 (Boolean function). A *Boolean function* of n variables is a function of $\mathbb{F}_2^n$ in $\mathbb{F}_2$. The representation in real values of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is given by $\hat{f} : \mathbb{F}_2^n \to \mathbb{R}$, where $\hat{f}(x) = (-1)^{f(x)}$.

Next, we formally define the concept of correlation, which we define informally in the previous section.

DEFINITION 3.5 (Correlation). Let $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ be Boolean functions. The *correlation* between f and g is defined by

$$c(f(x), g(x)) = 2\Pr(f(x) = g(x)) - 1. \tag{3.4}$$

Note that the correlation $c(f, g)$ is a value in the range $[-1, 1]$ and that $c(f, g) = c(g, f)$. Another way to define correlation is as follows:

$$c(f(x), g(x)) = 2^{-n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+g(x)} = 2^{-n} \sum_{x \in \mathbb{F}_2^n} \hat{f}(x)\hat{g}(x). \tag{3.5}$$

To see that the two definitions are equivalent, let us define

$$l = \#\{x \in \mathbb{F}_2^n; f(x) + g(x) = 0\},$$

$$m = \#\{x \in \mathbb{F}_2^n; f(x) + g(x) = 1\}.$$

Therefore $l + m = 2^n$. Furthermore, we have $f(x) = g(x)$ if and only if $f(x) + g(x) = 0$. We can then use Eq. (3.5) to obtain

$$c(f(x), g(x)) = 2^{-n}(l - m) = 2^{-n}(2l - 2^n) = \frac{2l}{2^n} - 1 = 2\Pr(f(x) = g(x)) - 1,$$

which is exactly the expression (3.4).

3.2.2.2 Linear approximations

Let $w = (w_{n-1}, ..., w_0)$, $x = (x_{n-1}, ..., x_0)$, we define $w \cdot x = x_{n-1}w_{n-1} + ... + x_0w_0$ as the inner product of x and w. We define the concept of linear approximation and efficiency.

DEFINITION 3.6 (Linear Approximation) Let $h : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function. Denote by $u \overset{h}{\leftarrow} w$ the *linear approximation* of the function h such that

$$u \cdot h(x) = w \cdot x.$$

DEFINITION 3.7 (Efficiency) The *efficiency* of the linear approximation $u \overset{h}{\leftarrow} w$ is given by

$$\mathcal{C}(u \overset{h}{\leftarrow} w) = c(u \cdot h(x), w \cdot x),$$

where c is the correlation presented in Definition 3.5.

3.2.2.3 Fourier Transform

Define the function δ such that $\delta(0) = 1$ and $\delta(x) = 0$, if $x \neq 0$. We define the characteristic function and the Fourier transform.

DEFINITION 3.8 (Characteristic Function) Let $h : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function. So its *characteristic function* is given by

$$\chi_h(x, y) = \delta(h(x) \oplus y),$$

where $x \in \mathbb{F}_2^n$ and $y \in \mathbb{F}_2^m$.

Definition 3.9 (Fourier transform) We define the *Fourier transform* of the function $h : \mathbb{F}_2^n \to \mathbb{F}_2^m$ as

$$\mathcal{F}(\chi_h)(u,w) = 2^{-mn} \sum_{x,y} \chi_h(x,y)(-1)^{u \cdot y + w \cdot x}$$

Lemma 3.1 It holds that

$$\mathcal{F}(\chi_h)(u,w) = 2^{-m}\mathcal{C}(u \overset{h}{\leftarrow} w).$$

proof.

By Definition 3.8, it is easy to see that

$$\chi_h(x,y) = 1 \iff h(x) = y,$$

so by Definition 3.9, it follows that

$$2^{-mn} \sum_{x,y} \chi_h(x,y)(-1)^{u \cdot y + w \cdot x} = 2^{-mn} \sum_{x} (-1)^{u \cdot h(x) + w \cdot x}.$$

The proof is concluded by verifying that the right side of the equation is equal to $2^{-m}\mathcal{C}(u \overset{h}{\leftarrow} w)$ from Definition 3.7 and Eq. (3.5). ■

The Lemma 3.1 is extremely important because it shows that we can compute the correlation presented by a linear approximation from the FFT. The great advantage of this approach is performance, since the FFT is recognized as an efficient algorithm.

3.3 DIFFERENTIAL-LINEAR CRYPTANALYSIS

In this section, we describe the technique of Differential-Linear cryptanalysis. Let E be a cipher and suppose we can write $E = E_2 \circ E_1$, where E_1 and E_2 are sub ciphers, covering m and l rounds of the main cipher, respectively. We can apply an input difference $\mathcal{ID}$ $\Delta X^{(0)}$ in the sub cipher E_1 obtaining an output difference $\mathcal{OD}$ $\Delta X^{(m)}$ (see the left side of Fig. 3.1). The next step is to apply Linear Cryptanalysis to the second sub cipher E_2. Using masks Γ_m and Γ_{out}, we attempt to find good linear approximations covering the remaining l rounds of the cipher E. Applying this technique, we can construct a differential-linear distinguisher covering all $m + l$ rounds of the cipher E. This is the main idea in Langford and Hellman's classical approach [39].

Sometimes, however, it can be useful to divide the cipher E into three other ciphers, i.e., $E = E_3 \circ E_2 \circ E_1$. In this scenario, we can explore properties of the cipher in the first part E_1 and then apply a differential linear attack where we divide the differential part of the attack in two (see the right side of Fig. 3.1). Here, the $\mathcal{OD}$ from the sub cipher E_1 after r rounds, namely $\Delta X^{(r)}$, is the $\mathcal{ID}$ for the sub cipher E_2, which produces an output difference $\Delta X^{(m)}$. For more information in this regard, see [26].

It is important to understand how to compute the complexity of a differential-linear attack. We denote the differential of the state matrix as $\Delta X^{(r)} = X^{(r)} \oplus X'^{(r)}$ and the differential of individual words as $\Delta x_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$. Let $x_{i,j}^{(r)}$ denote the j-th bit of the i-th word of the state matrix after r rounds and let $\mathcal{J}$ be a set of bits. Also, let σ and σ' be linear combinations of bits in the set $\mathcal{J}$

$$\sigma = \left(\bigoplus_{(i,j)\in\mathcal{J}} x_{i,j}^{(r)} \right), \quad \sigma' = \left(\bigoplus_{(i,j)\in\mathcal{J}} x_{i,j}'^{(r)} \right).$$

Thus,

$$\Delta\sigma = \left(\bigoplus_{(i,j)\in\mathcal{J}} \Delta x_{i,j}^{(r)} \right)$$

is the linear combination of the differentials. We can write

$$\Pr\left[\Delta\sigma = 0 | \Delta X^{(0)}\right] = \frac{1}{2}(1 + \varepsilon_d), \tag{3.6}$$

where ε_d is the differential correlation.

Using linear cryptanalysis, it is possible to go further and find new relations between the initial state matrix and the state matrix after $R > r$ rounds. To do so, let $\mathcal{L}$ denote another set of bits and define

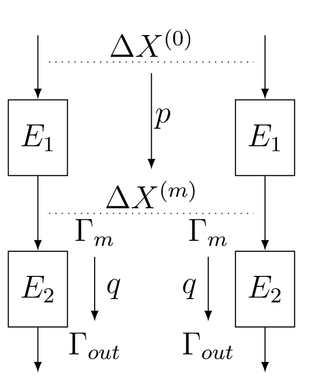
$$\rho = \left(\bigoplus_{(i,j)\in\mathcal{L}} x_{i,j}^{(R)} \right), \quad \rho' = \left(\bigoplus_{(i,j)\in\mathcal{L}} x_{i,j}'^{(R)} \right).$$

Thus, as before,

$$\Delta\rho = \left(\bigoplus_{(i,j)\in\mathcal{L}} \Delta x_{i,j}^{(R)} \right).$$

We can define $\Pr[\sigma = \rho] = \frac{1}{2}(1 + \varepsilon_L)$, where ε_L is the linear correlation. We want to find γ such that $\Pr\left[\Delta\rho = 0 | \Delta X^{(0)}\right] = \frac{1}{2}(1 + \gamma)$.

To compute γ, we write (to simplify the notation, we make the conditional to $\Delta X^{(0)}$



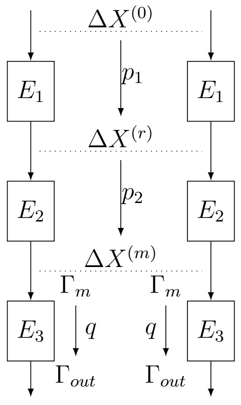


Figure 3.1 – A classical differential-linear distinguisher (on the left) and a differential-linear distinguisher with experimental evaluation of the correlation p_2 (on the right). E denotes a cipher that may be divided into sub-ciphers $E = E_2 \circ E_1$, or $E = E_3 \circ E_2 \circ E_1$. In the differential part we may apply an input difference $\mathbb{ID}$ $\Delta X^{(0)}$ in the sub cipher E_1 obtaining an output difference $\mathbb{OD}$ $\Delta X^{(m)}$ after m rounds. The next step is to apply Linear Cryptanalysis using masks Γ_m and Γ_{out}. Applying this technique we can construct a differential-linear distinguisher of the cipher E. One way to improve attacks is to explore properties of the cipher in the first part E_1 (on the right), and then apply a differential linear attack where we divide the differential part of the attack in two.

implicit):

$$\begin{aligned}\Pr[\Delta\sigma = \Delta\rho] &= \Pr[\sigma = \rho] \cdot \Pr\left[\sigma' = \rho'\right] + \Pr[\sigma = \bar{\rho}] \cdot \Pr\left[\sigma' = \overline{\rho'}\right] \\ &= \frac{1}{2}\left(1 + \varepsilon_L^2\right).\end{aligned}$$

Thus,

$$\begin{aligned}\Pr[\Delta\rho = 0] &= \Pr[\Delta\sigma = 0] \cdot \Pr[\Delta\sigma = \Delta\rho] + \Pr[\Delta\sigma = 1] \cdot \Pr[\Delta\sigma = \overline{\Delta\rho}] \\ &= \frac{1}{2}\left(1 + \varepsilon_d \cdot \varepsilon_L^2\right).\end{aligned}$$

Therefore, the differential-linear correlation is given by $\gamma = \varepsilon_d \cdot \varepsilon_L^2$, which defines a distinguisher with complexity $\mathcal{O}\left(\frac{1}{\varepsilon_d^2 \varepsilon_L^4}\right)$. For further information on differential-linear cryptanalysis, we refer to [40].

3.4 PROBABILISTIC NEUTRAL BITS

This section reviews the attack of Aumasson [19]. The attack first identifies good choices of truncated differentials, then it uses probabilistic backwards computation with the notion of Probabilistic Neutral Bits (PNB), and, finally, it estimates the complexity of the attack.

If $\Delta x_i^{(R)}$ is the difference for the i_{th} word of state matrix $X^{(R)}$, thus $\Delta x_i^{(R)} = x_i^{(R)} \oplus x'^{(R)}_i$; and if $\Delta x_{i,j}^{(R)}$ is the difference for the j_{th} bit of the i_{th} word, then $\Delta x_{i,j}^{(R)} = x_{i,j}^{(R)} \oplus x'^{(R)}_{i,j}$. In [19], the input difference $\mathcal{ID}$ is defined for a single-bit difference $\Delta x_{i,j}^{0} = 1$ and consider a single-bit output difference $\mathcal{OD}$ after r rounds $\Delta x_{p,q}^{(r)}$, such differential is denoted $(\Delta x_{p,q}^{(r)} | \Delta x_{i,j}^{0})$. For a fixed key, the correlation ε_d of the $\mathcal{OD}$ is defined by $\Pr_{v,t}(\Delta x_{p,q}^{(r)} = 1 | \Delta x_{i,j}^{0}) = \frac{1}{2}(1 + \varepsilon_d)$, where the probability holds over all nonces v and counters t. Furthermore, considering the key as a random variable, we denote the median value of ε_d by $\varepsilon_d^{\star}$. Hence, for half of the keys, this differential have a correlation of at least $\varepsilon_d^{\star}$.

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

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

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

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

If we denote the correlation of g by ε, i.e., $\Pr(g(\bar{k}, v, t, Z, Z') = 1) = \frac{1}{2}(1+\varepsilon)$, and $\varepsilon^{\star}$ the median correlation of g over all keys, we can approximate ε by $\varepsilon_d \varepsilon_a$. The problem that remains is how to efficiently find such a function g. In [19], this is done by first identifying key bits that have little influence on the result of $f(k, v, t, Z, Z')$, these are called *probabilistic neutral bits* (PNBs). This is done by defining the *neutrality measure* $\gamma_{i,j}$ of a key bit $k_{i,j}$.

After computing $\gamma_{i,j}$ (see [19] for a method of estimation), for all $i = (0, 1, ..., 7)$ and $j = (0, 1, ..., 31)$, we can define the set of significant key bits as $\Psi = \{(i, j) : \gamma_{i,j} \leq \gamma\}$ where γ is a threshold value, and then define our approximation g as $g(k_{\Psi}, v, t, Z, Z') = f(k^{*}, v, t, Z, Z')$ where k_{Ψ} is defined as the subkey with key bits in the set Ψ and k^{*} is computed from k_{Ψ} by setting $k_{i,j} = 0$ for all $(i, j) \notin \Psi$. Thus, the attack can be evaluated with the following steps:

1. Compute a good differential for r rounds $(\Delta x_{p,q}^{(r)} | \Delta x_{i,j}^{0})$ by estimating the correlation ε_d for all single-bit $\mathcal{ID}$ with several random combinations of keys, nonces, and counters.

2. Empirically estimate the neutrality measure $\gamma_{r,s}$ for each key bit $k_{r,s}$.

3. Construct the function g by setting all key bit such that $\gamma_{r,s} > \gamma$ to zero and estimate the median correlation $\varepsilon^{\star}$ by empirically measuring correlation of g using many randomly

chosen keys, nonces, and counters.

4. Estimate the data and time complexity of the attack.

We refer to [19] for further information about the estimation of the data and time complexity of the attack and for further details on the described technique.

3.5 CRYPTANALYSIS OF ARX CIPHERS

In this thesis, we improve results of cryptanalysis against ARX ciphers. To do so, first, we need to understand how to apply differential and linear cryptanalysis when we are faced against the addition operation.

3.5.1 Differential cryptanalysis of Addition

Differential cryptanalysis emerged with the intention of being applied to block ciphers that have an SPN-like structure, as can be seen in the previous sections. However, there are several cryptographic algorithms that use only add, rotate, and XOR (ARX) operations and do not necessarily constitute an SPN. Therefore, it is important for the cryptanalyst to know in depth the way in which differential cryptanalysis can be applied in ARX-like algorithms.

In general, to analyze ARX-type ciphers, one can use the differential by the XOR operation or by the subtraction operation, that is, we could define the differential as

$$\Delta X = X \oplus X'$$

or as

$$\Delta X = X - X'.$$

Despite its possibilities for the ARX case, the present thesis continues using the differential in relation to the XOR operation.

In the following sections, we present how to compute the differential in an ARX cipher, starting with the simplest operations, XOR and rotation. Afterwards, we describe how to calculate the differential for the addition operation, which consists of the nonlinear element when considers the differential by the XOR operation. We present algorithms and the theory heavily based on the work of Lipmaa [41], which is recommended for the reader who wants more details on what is presented in this thesis.

3.5.1.1 Preliminary notions

In an ARX algorithm, when working with the differences based on the XOR operation, we need to know how to compute and update differences for each internal operation. Doing this for the XOR and rotation operations is trivial, but the situation gets more complex in the case of the addition operation. The following lemmas show the reason for this fact.

LEMMA 3.2 Let $Z = X \oplus Y$ and $Z' = X' \oplus Y'$, where $X, Y, Z \in \mathbb{F}_2^n$. In this case, we have that $\Delta Z = \Delta X \oplus \Delta Y$.

proof.

Follows directly from:

$$\Delta Z = Z \oplus Z' = X \oplus Y \oplus X' \oplus Y' = \Delta X \oplus \Delta Y.$$

■

LEMMA 3.3 Let $Z = X \ggg R$ and $Z' = X' \ggg R$, where $X, X', Z, Z' \in \mathbb{F}_2^n$. Therefore, we have that $\Delta Z = \Delta X \ggg R$.

proof.

We have $\Delta Z = (X \ggg R) \oplus (X' \ggg R)$. Since the rotation do not flip any bit, we get $\Delta Z = (X \oplus X') \ggg R = \Delta X \ggg R$. ■

LEMMA 3.4 Let $Z = X + Y$ e $Z' = X' + Y'$, where $X, X', Y, Y', Z, Z' \in \mathbb{F}_2^n$. In general, we have that $\Delta Z \neq \Delta X + \Delta Y$.

proof.

We have that

$$\Delta Z = Z \oplus Z' = (X + Y) \oplus (X' + Y'),$$

and

$$\Delta X + \Delta Y = (X \oplus Y) + (X' \oplus Y').$$

However, in general, we have that

$$(X + Y) \oplus (X' + Y') \neq (X \oplus Y) + (X' \oplus Y').$$

3.5.1.2 Analyzing the addition operation on each bit

The addition operation is quite simple when thought of in integers, but when viewed in terms of bits and Boolean functions, it can present some surprises. Consider the sum $X+Y=Z$, where $X,Y,Z \in \mathbb{F}_2^4$, that is, have 4 bits and denote $X = x_3||x_2||x_1||x_0$, where x_0 denotes the least significant bit and x_3 the most significant bit. We can then represent the sum in the same way that we learned to add decimal numbers in elementary school, but here, we will work on bits:

$$\begin{array}{cc} & c_3\ c_2\ c_1\ c_0 \\ + & x_3\ x_2\ x_1\ x_0 \\ & y_3\ y_2\ y_1\ y_0 \\ \hline & z_3\ z_2\ z_1\ z_0 \end{array}$$

where c_3, c_2, c_1 e c_0 denote the *carry* bits. By definition, $c_0 = 0$. Using basic addition rules, we get that $c_1 = 1$ if $x_0 = y_0 = 1$ and $c_1 = 0$ otherwise. Also, $c_2 = 1$ if at least two bits between x_1, y_1 and c_1 are equal to 1. Generally, it holds that:

$$c_{i+1} = \text{MAJ}(c_i, x_i, y_i) = c_i x_i \oplus c_i y_i \oplus x_i y_i. \tag{3.7}$$

It is also not difficult to note (it is left as an exercise for the reader) that the following relation is valid

$$z_i = x_i \oplus y_i \oplus c_i. \tag{3.8}$$

3.5.1.3 Addition Differential Properties

Following the notation of [41], let $\alpha = \Delta X$, $\beta = \Delta Y$ and $\gamma = \Delta Z$. Also, let $\pi = \Delta C = C \oplus C'$, where C denotes the *carry* of the sum $X+Y$ and C' denotes the *carry* of the sum $X' + Y'$. We can represent the two sums as follows:

$$\begin{array}{cc|cc} & c_3\ c_2\ c_1\ c_0 & & c'_3\ c'_2\ c'_1\ c'_0 \\ + & x_3\ x_2\ x_1\ x_0 & + & x'_3\ x'_2\ x'_1\ x'_0 \\ & y_3\ y_2\ y_1\ y_0 & & y'_3\ y'_2\ y'_1\ y'_0 \\ \cline{2-2}\cline{4-4} & z_3\ z_2\ z_1\ z_0 & & z'_3\ z'_2\ z'_1\ z'_0 \end{array}$$

such that $\alpha_i = x_i \oplus x'_i$, $\beta_i = y_i \oplus y'_i$, $\gamma_i = z_i \oplus z'_i$ and $\pi_i = c_i \oplus c'_i$.

Now, let us check some basic properties.

Lemma 3.5 It holds that $\gamma_i = \alpha_i \oplus \beta_i \oplus \pi_i$.

proof.

It follows directly by the application of the differential rule for the XOR operation presented in the Lemma 3.2, and by Eq. (3.8). ■

Lemma 3.6 It holds that $\pi_0 = 0$.

proof.

By definition, we have that, $c_0 = c'_0 = 0$, therefore $\pi_0 = c_0 \oplus c'_0 = 0$. ■

Lemma 3.7 It holds that $\alpha_0 \oplus \beta_0 \oplus \gamma_0 = 0$.

proof.

From Lemma 3.6, we have that $\pi_0 = 0$, therefore, the result follows directly from Lemma 3.5. ■

Lemma 3.8 If $\alpha_i = \beta_i = \gamma_i = 0$, then $\pi_{i+1} = 0$.

proof.

From Lemma 3.5, we have that $\pi_i = 0$. Since $\alpha_i = 0$, we have that $x_i = x'_i$, however, we do not know if $x_i = x'_i = 0$ or if $x_i = x'_i = 1$. In the same manner, as $\beta_i = 0$, $\gamma_i = 0$, and $\pi_i = 0$, we have that $y_i = y'_i$, $z_i = z'_i$ and $c_i = c'_i$, respectively, but we do not know the exact value. Therefore, we have to assume all the existing possibilities. For example, if $x_i = x'_i = 0$, $y_i = y'_i = 0$ and $c_i = c'_i = 0$ we have

$$c_{i+1} = \text{MAJ}(x_i, y_i, c_i) = \text{MAJ}(0, 0, 0) = 0$$

and

$$c'_{i+1} = \text{MAJ}(x'_i, y'_i, c'_i) = \text{MAJ}(0, 0, 0) = 0,$$

therefore, $\pi_{i+1} = c_{i+1} \oplus c'_{i+1} = 0$. Performing this procedure for all possibilities, we arrive at the table below, which shows that for any combination $\pi_{i+1} = 0$.

α_i	β_i	γ_i	π_i	x_i	x'_i	y_i	y'_i	c_i	c'_i	c_{i+1}	c'_{i+1}	π_{i+1}
0	0	0	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	1	1	0	0	0
				0	0	1	1	0	0	0	0	0
				0	0	1	1	1	1	0	0	0
				1	1	0	0	0	0	0	0	0
				1	1	0	0	1	1	0	0	0
				1	1	1	1	0	0	0	0	0
				1	1	1	1	1	1	0	0	0

■

Lemma 3.9 If $\alpha_i = \beta_i = \gamma_i = 1$, then $\pi_{i+1} = 1$.

proof.

From Lemma 3.5, we have that $\pi_i = 1$. Since $\alpha_i = 1$, we have that $x_i \neq x'_i$, however, we do not know if $x_i = 0$ and $x'_i = 1$, or if $x_i = 1$ and $x'_i = 0$. In the same manner, as $\beta_i = 1$, $\gamma_i = 1$ and $\pi_i = 1$, we have that $y_i \neq y'_i$, $z_i \neq z'_i$ and $c_i \neq c'_i$, respectively, but we do not know the exact value. Therefore, we have to assume all the existing possibilities. For example, if $x_i = 1$, $x'_i = 0$, $y_i = 0$, $y'_i = 1$, $c_i = 0$, and $c'_i = 1$, we have that

$$c_{i+1} = \text{MAJ}(x_i, y_i, c_i) = \text{MAJ}(1, 0, 0) = 0$$

and

$$c'_{i+1} = \text{MAJ}(x'_i, y'_i, c'_i) = \text{MAJ}(0, 1, 1) = 1,$$

therefore, $\pi_{i+1} = c_{i+1} \oplus c'_{i+1} = 1$. Performing this procedure for all possibilities, we arrive at the table below, which shows that for any combination $\pi_{i+1} = 1$.

α_i	β_i	γ_i	π_i	x_i	x'_i	y_i	y'_i	c_i	c'_i	c_{i+1}	c'_{i+1}	π_{i+1}
1	1	1	1	1	0	1	0	1	0	1	0	1
				1	0	1	0	0	1	1	0	1
				1	0	0	1	1	0	1	0	1
				1	0	0	1	0	1	0	1	1
				0	1	1	0	1	0	1	0	1
				0	1	1	0	0	1	0	1	1
				0	1	0	1	1	0	0	1	1
				0	1	0	1	0	1	0	1	1

■

LEMMA 3.10 If $\alpha_i \neq \beta_i$ or $\alpha_i \neq \gamma_i$, then $\Pr\{\pi_{i+1} = 1\} = 0.5$.

proof.

In this case, we have that between α_i, β_i, and γ_i there is at least a value of 0 and a value of 1. For example, one possibility is that $(\alpha_i, \beta_i, \gamma_i) = (0, 1, 1)$. Computing all the possibilities for this case, we arrive at the table below in which it is noted that $\Pr\{\pi_{i+1} = 1\} = 0.5$:

α_i	β_i	γ_i	π_i	x_i	x'_i	y_i	y'_i	c_i	c'_i	c_{i+1}	c'_{i+1}	π_{i+1}
0	1	1	0	1	1	1	0	1	1	1	1	0
				1	1	1	0	0	0	1	0	1
				1	1	0	1	1	1	1	1	0
				1	1	0	1	0	0	0	1	1
				0	0	1	0	1	1	1	0	1
				0	0	1	0	0	0	0	0	0
				0	0	0	1	1	1	0	1	1
				0	0	0	1	0	0	0	0	0

By the symmetry of the MAJ function, the cases where $(\alpha_i, \beta_i, \gamma_i) = (1, 0, 1)$ or $(\alpha_i, \beta_i, \gamma_i) = (1, 1, 0)$ are analogous to the previous one. Another possibility would be when $(\alpha_i, \beta_i, \gamma_i) = (0, 0, 1)$, computing all the possibilities for this case, we arrive at the table below in which we can see that $\Pr\{\pi_{i+1} = 1\} = 0.5$:

α_i	β_i	γ_i	π_i	x_i	x'_i	y_i	y'_i	c_i	c'_i	c_{i+1}	c'_{i+1}	π_{i+1}
0	0	1	1	1	1	1	1	0	1	1	1	0
				1	1	1	1	1	0	1	1	0
				1	1	0	0	0	1	0	1	1
				1	1	0	0	1	0	1	0	1
				0	0	1	1	0	1	0	1	1
				0	0	1	1	1	0	1	0	1
				0	0	0	0	0	1	0	0	0
				0	0	0	0	1	0	0	0	0

By the symmetry of the MAJ function, $\Pr\{\pi_{i+1} = 1\} = 0.5$. ■

LEMMA 3.11 Given α, β, γ, and defining $\alpha_{-1} = \beta_{-1} = \gamma_{-1} = 0$. If i such that $\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} \neq \alpha_i \oplus \beta_i \oplus \gamma_i$, then it is an impossible differential.

proof.

Suppose for some index i, we have

$$\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} \neq \alpha_i \oplus \beta_i \oplus \gamma_i.$$

As we have that $\gamma_i = \alpha_i \oplus \beta_i \oplus \pi_i$ (by Lema 3.5), then we can rewrite the previous expression as

$$\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} \neq \pi_i.$$

Also by the equation of the Lemma 3.5, we can calculate π_{i-1}, in fact we have that $\pi_{i-1} = \alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}$. Now, we have two options:

1. If $\pi_{i-1} = \alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} = 0$, then $\pi_i = 0$ (by Lemma 3.8). Thus,

   $$\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} = \pi_i = 0,$$

   which contradicts the initial assumption.

2. If $\pi_{i-1} = \alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} = 1$, then $\pi_i = 1$ (by Lemma 3.9). Thus,

   $$\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} = \pi_i = 1,$$

   which contradicts the initial assumption.

Therefore, we have an impossible differential. ■

3.5.1.4 Differential Probability of Addition

In this section, we will evaluate the *Differential Probability of Addition* (DPA). Let $\alpha = \Delta X$, $\beta = \Delta Y$ and $\gamma = \Delta Z$, we define

$$\mathrm{DPA}(\alpha, \beta, \gamma) = \Pr_{x,y}\{(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma | \alpha, \beta\}. \tag{3.9}$$

In other words, $\mathrm{DPA}(\alpha, \beta, \gamma)$ defines the probability of getting an output difference γ given the input differences α and β, out of the entire space of possibilities for x and y.

The calculation of this probability is easy to understand given the knowledge of the properties presented in the previous section. Let us do this with an example. Assume the 4-bit differentials $\alpha = 1000$ and $\beta = 1010$. Let us represent these differentials in the traditional view for sum that we used earlier:

$$\begin{array}{c|cccc}
\pi & & & & \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\cline{2-5}
\gamma & & & &
\end{array}$$

By Lemma 3.6, we know that $\pi_0 = 0$, therefore, we have:

$$\begin{array}{c|cccc}
\pi & & & & 0 \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\cline{2-5}
\gamma & & & &
\end{array}$$

Since $\pi_0 = \alpha_0 = \beta_0 = 0$, we get $\gamma_0 = 0$ (by Lemma 3.5) and $\pi_1 = 0$ (by Lemma 3.8). Thus, we have

$$\begin{array}{c|cccc}
\pi & & & 0 & 0 \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\cline{2-5}
\gamma & & & & 0
\end{array}$$

Since $\pi_1 = \alpha_1 = 0$ and $\beta_1 = 1$, we have that $\gamma_1 = 1$ (by Lemma 3.5), therefore, we have:

$$\begin{array}{c|cccc}
\pi & & & 0 & 0 \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\cline{2-5}
\gamma & & & 1 & 0
\end{array}$$

Now, as $\alpha_1 \neq \beta_1$, we have an undetermined result for π_2, since $\Pr\{\pi_2 = 1\} = 0.5$ (by the Lemma 3.10). Hence, we divide the solution into two possibilities with probability 0.5, already calculating the result of γ_2:

$$\begin{array}{c}
\text{Probability} = \frac{1}{2} \\
\begin{array}{c|cccc}
\pi & & 0 & 0 & 0 \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\cline{2-5}
\gamma & & 0 & 1 & 0
\end{array}
\end{array}
\quad \vdots \quad
\begin{array}{c}
\text{Probability} = \frac{1}{2} \\
\begin{array}{c|cccc}
\pi & & 1 & 0 & 0 \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\cline{2-5}
\gamma & & 1 & 1 & 0
\end{array}
\end{array}$$

On the left hand, we have that $\pi_2 = \alpha_2 = \beta_2 = \gamma_2 = 0$, which implies that $\pi_3 = 0$ (by Lemma 3.8), so we have:

Probability = $\frac{1}{2}$

$$\begin{array}{ccccc} \pi & 0 & 0 & 0 & 0 \\ \alpha & 1 & 0 & 0 & 0 \\ \beta & 1 & 0 & 1 & 0 \\ \hline \gamma & 0 & 0 & 1 & 0 \end{array}$$

Probability = $\frac{1}{2}$

$$\begin{array}{ccccc} \pi & & 1 & 0 & 0 \\ \alpha & 1 & 0 & 0 & 0 \\ \beta & 1 & 0 & 1 & 0 \\ \hline \gamma & & 1 & 1 & 0 \end{array}$$

On the right hand, as $\alpha_2 \neq \gamma_2$, we have an undetermined result for π_3, since $\Pr\{\pi_3 = 1\} = 0.5$ (by Lemma 3.10). Hence, we split the solution into two possibilities with resulting probability 0.25, getting as the result:

Probability = $\frac{1}{2}$

$$\begin{array}{ccccc} \pi & 0 & 0 & 0 & 0 \\ \alpha & 1 & 0 & 0 & 0 \\ \beta & 1 & 0 & 1 & 0 \\ \hline \gamma & 0 & 0 & 1 & 0 \end{array}$$

Probability = $\frac{1}{4}$

$$\begin{array}{ccccc} \pi & 0 & 1 & 0 & 0 \\ \alpha & 1 & 0 & 0 & 0 \\ \beta & 1 & 0 & 1 & 0 \\ \hline \gamma & 0 & 1 & 1 & 0 \end{array}$$

Probability = $\frac{1}{4}$

$$\begin{array}{ccccc} \pi & 1 & 1 & 0 & 0 \\ \alpha & 1 & 0 & 0 & 0 \\ \beta & 1 & 0 & 1 & 0 \\ \hline \gamma & 1 & 1 & 1 & 0 \end{array}$$

Therefore, we can summarize the probability distribution as

$$
\begin{aligned}
\mathrm{DPA}(1000, 1010, 0000) &= 0 \\
\mathrm{DPA}(1000, 1010, 0001) &= 0 \\
\mathrm{DPA}(1000, 1010, 0010) &= 0.5 \\
\mathrm{DPA}(1000, 1010, 0011) &= 0 \\
\mathrm{DPA}(1000, 1010, 0100) &= 0 \\
\mathrm{DPA}(1000, 1010, 0101) &= 0 \\
\mathrm{DPA}(1000, 1010, 0110) &= 0.25 \\
\mathrm{DPA}(1000, 1010, 0111) &= 0 \\
\mathrm{DPA}(1000, 1010, 1000) &= 0 \\
\mathrm{DPA}(1000, 1010, 1001) &= 0 \\
\mathrm{DPA}(1000, 1010, 1010) &= 0 \\
\mathrm{DPA}(1000, 1010, 1011) &= 0 \\
\mathrm{DPA}(1000, 1010, 1100) &= 0 \\
\mathrm{DPA}(1000, 1010, 1101) &= 0 \\
\mathrm{DPA}(1000, 1010, 1110) &= 0.25 \\
\mathrm{DPA}(1000, 1010, 1111) &= 0
\end{aligned}
$$

Similarly, it is possible to construct an algorithm that computes $\mathrm{DPA}(\alpha, \beta, \gamma)$ from α, β, γ. For example, let $\alpha = 1000$, $\beta = 1010$, and $\gamma = 1110$. In this case, we have that $\pi = 1100$ (by Lemma 3.5). As $\gamma_0 = \alpha_0 = \beta_0 = 0$, we have $\pi_1 = 0$ with probability 1. Next, as $\alpha_1 \neq \beta_1$, we have a transition with probability 0.5, and so on, from as follows:

$$
\begin{array}{c ccccccc}
 & & \overset{\frac{1}{2}}{\curvearrowleft} & & \overset{\frac{1}{2}}{\curvearrowleft} & & \overset{1}{\curvearrowleft} & \\
\pi & 1 & & 1 & & 0 & & 0 \\
\alpha & 1 & & 0 & & 0 & & 0 \\
\beta & 1 & & 0 & & 1 & & 0 \\
\hline
\gamma & 1 & & 1 & & 1 & & 0
\end{array}
$$

To compute $\mathrm{DPA}(1000, 1010, 1110)$, we simply multiply the probabilities. So

$$\mathrm{DPA}(\alpha, \beta, \gamma) = 1 \times 0.5 \times 0.5 = 0.25.$$

One more example, let $\alpha = 1000$, $\beta = 1010$ and $\gamma = 0000$. In this case, we have $\pi = 0010$ (by Lemma 3.5). However, the value of π_1 is inconsistent, which implies an impossible differential, and therefore, the probability is 0:

$$\begin{tikzcd}[row sep=tiny, column sep=small]
\pi & 0 & 0 \arrow[l, bend right, "1"'] & 1 \arrow[l, bend right, "\frac{1}{2}"'] & 0 \arrow[l, bend right, "0"'] \\
\alpha & 1 & 0 & 0 & 0 \\
\beta & 1 & 0 & 1 & 0 \\
\gamma & 0 & 0 & 0 & 0
\end{tikzcd}$$

Therefore, we get

$$\mathrm{DPA}(1000, 1010, 000) = 1 \times 0.5 \times 0 = 0.$$

3.5.1.5 Maximizing the probability from α and β

In differential cryptanalysis, it is common for the cryptanalyst to want to obtain a differential path with high probability. For this, it is useful to have the knowledge of how to choose an output difference that has maximum probability given two inputs. Thus, we want to find the value for γ that maximizes the DPA, that is:

$$\max_{\gamma} \mathrm{DPA}(\alpha, \beta, \gamma).$$

In the previous section, we saw how to calculate probabilities for $\mathrm{DPA}(\alpha, \beta, \gamma)$. In the presented method, we observed only three types of possible transitions during the probability calculation:

1. Impossible transitions whose probability is 0.

2. Deterministic transitions whose probability is 1.

3. Probabilistic transitions with probability 0.5.

Now, if we want to maximize the probability, we need to maximize the number of deterministic transitions. We can do that by choosing convenient values during the probabilistic transitions. To understand this dynamic, consider $\alpha = 10001110$ and $\beta = 11000111$. Remembering that $\pi_0 = 0$, we have:

$$\begin{array}{c|cccccccc}
\pi & & & & & & & & 0 \\
\alpha & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\beta & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\hline
\gamma & & & & & & & &
\end{array}$$

Naturally, $\gamma_0 = 1$, and since $\alpha_0 \neq \beta_0$ then the value of π_1 is undetermined. In that way, we can choose any value for π_1. However, what would be the value that maximizes the probability? To answer this question, we will test both hypotheses. Suppose we choose the value $\pi_1 = 0$. In this case, we have:

$$\begin{array}{c|cccccccc}
 & & & & & & & \frac{1}{2} & \frac{1}{2} \\
\pi & & & & & & & 0 & 0 \\
\alpha & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\beta & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\hline
\gamma & & & & & & & 0 & 1
\end{array}$$

this choice implies two consecutive transitions of 0.5, accumulating the probability of the partial differential at 0.25. In the other case, if we choose $\pi_1 = 1$, we get:

$$\begin{array}{c|cccccccc}
 & & & & & & & 1 & \frac{1}{2} \\
\pi & & & & & & & 1 & 0 \\
\alpha & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\beta & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\hline
\gamma & & & & & & & 1 & 1
\end{array}$$

in this case, as $\gamma_1 = \alpha_1 = \beta_1$ we have a transition of probability 0.5 and another of probability 1, accumulating the probability of the partial differential in 0.5, greater than the previous one. Therefore, the maximization algorithm (Algorithm 1) forces the transition of probability equal to 1 as many times as possible. To prove that this is really the maximum, it would be necessary to prove that the step-by-step maximization results in the global maximization, as it could be the case that deciding not to optimize a step would result in a better result in the future, but this proof is left as an exercise for the interested reader.

Algorithm 1 *MaxGamma*: Find γ that maximizes the DPA

```
procedure INPUT: DIFFERENCES $\alpha$ AND $\beta$ AND THEIR LENGTH $n$ IN BITS.
    $\pi_0 = 0$
    for $i \in \{0, ..., n-2\}$ do
        $\gamma_i = \alpha_i \oplus \beta_i \oplus \pi_i$
        if $\alpha_i = \beta_i = \gamma_i$ then
            $\pi_{i+1} = \alpha_i$
        else
            if $\alpha_{i+1} = \beta_{i+1}$ then
                $\pi_{i+1} = \alpha_{i+1}$
            else
                $\pi_{i+1} =$ generates 0 or 1, randomly.
    $\gamma_{n-1} = \alpha_{n-1} \oplus \beta_{n-1} \oplus \pi_{n-1}$
    return $\gamma$
```

Continuing with this procedure:

$$\begin{array}{c|cccccccc}
 & & & & & & 1 & 1 & \frac{1}{2} \\
\pi & & & & & 1 & 1 & 1 & 0 \\
\alpha & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\beta & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\hline
\gamma & & & & & 0 & 1 & 1 & 1
\end{array}$$

Next, we choose the zero bit since the next values of α and β are equal to zero, and so on:

$$\begin{array}{c|cccccccc}
 & & \frac{1}{2} & 1 & 1 & \frac{1}{2} & 1 & 1 & \frac{1}{2} \\
\pi & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\alpha & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\beta & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\hline
\gamma & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}$$

Therefore,

$$\mathrm{DPA}(10001110, 11000111, 11000111) = \frac{1}{8},$$

where this is the highest possible probability for the differences α and β.

3.5.1.6 Maximizing the probability from α

In the previous section, we found γ that maximizes the DPA given the input differences α and β. Another possibility is to seek to maximize the probability starting only from α, that is, to find β such that the subsequent search for a γ by the Algorithm 1 generates the highest possible probability. The logic here is very similar to the previous one: we want to maximize deterministic transitions and minimize probabilistic transitions, so let us do another example, if $\alpha = 00110110$, we have:

$$\begin{array}{c|cccccccc}
\pi & & & & & & & & 0 \\
\alpha & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\beta & & & & & & & & \\
\hline
\gamma & & & & & & & &
\end{array}$$

if we set $\beta_0 = 0$, then we have $\gamma_0 = 0$ and the transition will be deterministic. If we choose $\beta_0 = 1$ the transition will be with probability 0.5, so we set $\beta_0 = 0$:

$$\begin{array}{c|cccccccc}
\pi & & & & & & & 0 & 0 \\
\alpha & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\beta & & & & & & & & 0 \\
\hline
\gamma & & & & & & & & 0
\end{array}$$

now, as $\alpha_1 \neq \pi_1$, no matter what we choose for β_1, the transition probability will be equal to 0.5. Suppose $\beta_1 = 0$:

$$
\begin{array}{c|cccccccc}
 & & & & & & & \tfrac{1}{2} & 1 \\
\pi & & & & & & & 0 & 0 \\
\alpha & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\beta & & & & & & & 0 & 0 \\
\hline
\gamma & & & & & & & 1 & 0
\end{array}
$$

Next, we can choose the value for π_2 and for β_2, which we will both choose equal to α_2 since this will imply a deterministic transition:

$$
\begin{array}{c|cccccccc}
 & & & & & & 1 & \tfrac{1}{2} & 1 \\
\pi & & & & & 1 & 1 & 0 & 0 \\
\alpha & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\beta & & & & & & 1 & 0 & 0 \\
\hline
\gamma & & & & & & 1 & 1 & 0
\end{array}
$$

now, as $\alpha_3 \neq \pi_3$, we can choose any value for β_3, suppose it is 1, we repeat these steps until the end, which results in the Algorithm 2:

$$
\begin{array}{c|cccccccc}
 & & \tfrac{1}{2} & 1 & 1 & \tfrac{1}{2} & 1 & \tfrac{1}{2} & 1 \\
\pi & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
\alpha & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\beta & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
\hline
\gamma & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

Therefore,

$$\text{DPA}(00110110, 00111100, 01110110) = \frac{1}{8},$$

and this is the highest possible probability given the input difference α.

3.5.2 Linear cryptanalysis of Addition

In this section, we will study the linear approximations for the modular addition. For this, we will use several results presented in Section 3.2.2, following and seeking to simplify the methodology presented in [37, 42].

As usual, we use the symbol $\oplus$ to denote the sum in $\mathbb{F}_2^n$, $\boxplus$ to denote the sum of integers modulo 2^n, and $+$ denotes the sum in $\mathbb{F}_2$ used to simplify the notation of Boolean equations.

Algorithm 2 *MaxBeta*: Find the difference β that maximizes the DPA

```
1: procedure INPUT: DIFFERENCE $\alpha$ AND YOUR LENGTH IN BITS $n$.
2:     $\pi_0 = 0$
3:     $\alpha_n = 0$
4:     for $i \in \{0, ..., n-1\}$ do
5:         if $\pi_i = \alpha_i$ then
6:             $\beta_i = \alpha_i$
7:         else
8:             $\beta_i =$ generates 0 or 1, randomly.
9:         if $\alpha_i = \pi_i = \beta_i$ then
10:            $\pi_{i+1} = \alpha_i$
11:        else
12:            $\pi_{i+1} = \alpha_{i+1}$
```

Let $carry : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the carry function of the sum modulo 2^n, set to

$$carry(x, y) = x \oplus y \oplus (x \boxplus y).$$

3.5.2.1 Linear approximation for addition and subtraction as a function of the carry

We define the functions $cls, clc, clsub, clcarrysub : (\mathbb{F}_2^n)^3 \to [-1, 1]$ for the linear correlation of the sum, of the *carry* function, of the subtraction and the *carry* of the subtraction, respectively:

$$\begin{array}{lcl} cls(u, v, w) & = & \mathcal{C}(u \overset{\boxplus}{\leftarrow} v, w), \\ clc(u, v, w) & = & \mathcal{C}(u \xleftarrow{carry} v, w), \\ clsub(u, v, w) & = & \mathcal{C}(u \overset{\boxminus}{\leftarrow} v, w) \\ clcarrysub(u, v, w) & = & \mathcal{C}(u \xleftarrow{carrysub} v, w). \end{array}$$

Hence, these four functions are closely related. In fact, you can use the linear approximation to the carry of the sum to obtain a linear approximation to the sum, subtraction, or carry of the subtraction. In fact, let $z = x - y$. By Definition 3.6, we have that

$$\begin{gathered} u \overset{\boxplus}{\leftarrow} v, w \Rightarrow \\ u \cdot (x \boxplus y) = v \cdot x + w \cdot y \Rightarrow \\ u \cdot (x \oplus y \oplus carry(x, y)) = v \cdot x + w \cdot y \Rightarrow \\ u \cdot carry(x, y) = (v + u) \cdot x + (w + u) \cdot y \Rightarrow \\ u \xleftarrow{carry} v + u, w + u \end{gathered}$$

$$
\begin{gathered}
u \overset{\boxminus}{\leftarrow} v, w \Rightarrow \\
u \cdot (x \boxminus y) = v \cdot x + w \cdot y \Rightarrow \\
u \cdot z = v \cdot (z + y) + w \cdot y \Rightarrow \\
v \cdot (z \boxplus y) = u \cdot z + w \cdot y \Rightarrow \\
v \overset{\boxplus}{\leftarrow} u, w \\
v \xleftarrow{carry} u + v, w + v
\end{gathered}
$$

$$
\begin{gathered}
u \xleftarrow{carrysub} v, w \Rightarrow \\
u \cdot (carrysub(x, y)) = v \cdot x + w \cdot y \Rightarrow \\
u \cdot (x \oplus y \oplus (x \boxminus y)) = v \cdot x + w \cdot y \Rightarrow \\
u \cdot (x \boxminus y) = (v + u) \cdot x + (w + u) \cdot y \Rightarrow \\
u \overset{\boxminus}{\leftarrow} v + u, w + u \\
v + u \xleftarrow{carry} v, w
\end{gathered}
$$

The same can be done for subtraction. Hence, we conclude that

$$cls(u, v, w) = clc(u, v + u, w + u), \tag{3.10}$$

$$clsub(u, v, w) = clc(v, v + u, w + v), \tag{3.11}$$

and

$$clcarrysub(u, v, w) = clc(u + v, v, w). \tag{3.12}$$

Therefore, it is possible to work only with the linear approximation for the *carry* and use the results to compute approximations for the addition and subtraction.

3.5.2.2 Linear approximation to the carry function

Let $c_i = carry(x, y)_i$, that is, the i-th bit of *carry*. It is known that *carry* bits can be calculated recursively. In fact, $c_{i+1} = 1$ if at least two values among x_i, y_i and c_i are equal to 1. Testing all combinations for x_i, y_i and c_i, we can show that by setting $c_0 = 0$, or $\hat{c}_0 = (-1)^0 = 1$, we can calculate $\hat{c}_i$ from the recursive equation

$$\hat{c}_{i+1} = \frac{1}{2}((-1)^{x_i} + (-1)^{y_i} + (-1)^{c_i} - (-1)^{x_i+y_i+c_i}) = \frac{1}{2}(\hat{x}_i + \hat{y}_i + \hat{c}_i - \hat{x}_i\hat{y}_i\hat{c}_i). \tag{3.13}$$

As a result of this recursive construction, it becomes evident that the c_i bit is independent of either x_j or y_j, if $j \geq i$. Independence implies zero correlation. Therefore, the following proposition is valid:

Proposition 3.1 Let u_i be the i-th bit of the vector u. Let k be the largest integer such that $u_k = 1$, then $clc(u, v, w) = 0$ whenever there is $v_i = 1$ or $w_i = 1$ such that $i \geq k$.

To calculate other possibilities for the clc function, let us first consider that u is such that $u_0 = 1$ and $u_j = 0$, for all $j \neq 0$. In this case, we have

$$\hat{C}_0(v, w) = clc(u, v, w) = \mathcal{C}(u \xleftarrow{carry} v, w) = c(c_0, v.x + w.y) =$$

$$= 2^{-2n} \sum_{x,y} \hat{c}_0(-1)^{v.x+w.y} = 2^{-2n} \sum_{x,y} (-1)^{v.x+w.y} = \delta(v, w), \tag{3.14}$$

where $\delta(0, 0) = 1$ and $\delta(v, w) = 0$ if $v \neq 0$ or $w \neq 0$.

Now, if $u_{i+1} = 1$, and $u_j = 0$, for all $i \geq 0$ and $j \neq i + 1$, we might use Eq. (3.13) to get

$$\hat{C}_{i+1}(v, w) = c(c_{i+1}, v.x + w.y) = 2^{-2n} \sum_{x,y} \hat{c}_{i+1}(-1)^{v.x+w.y} =$$

$$= 2^{-2n} \sum_{x,y} \frac{1}{2}((-1)^{x_i+v.x+wy} + (-1)^{y_i+v.x+wy} + \hat{c}_i(-1)^{v.x+wy} - \hat{c}_i(-1)^{x_i+y_i+v.x+wy}) =$$

$$= \frac{1}{2}(\delta(v + e_i, w) + \delta(v, w + e_i) + \hat{C}_i(v, w) - \hat{C}_i(v + e_i, w + e_i)), \tag{3.15}$$

where $e_i \in \mathbb{F}_2^n$ is the canonical vector.

Lemma 3.12 In the expression

$$\delta(v + e_i, w) + \delta(v, w + e_i) + \hat{C}_i(v, w) - \hat{C}_i(v + e_i, w + e_i)$$

at most one of the four terms is non-zero.

proof.

The proof is simple and can be obtained by the interested reader considering the definition of the function δ and the Proposition 3.1. ∎

Now, we are ready to prove the following lemma:

Lemma 3.13 The function $clc(u, v, w)$ is given recursively as follows. First, $clc(0, v, w) = clc(e_0, v, w) = \delta(v, w)$. Second, if $u \notin \{0, e_0\}$, let k be the largest value such that $u_k = 1$ and let $i \geq k$. Thus,

$$clc(u + e_{i+1}, v, w) = \frac{1}{2}\begin{cases} clc(u, v\bar{e}_i, w\bar{e}_i) & \text{se } v_i \neq w_i \text{ and} \\ (-1)^{v_i} clc(u + e_i, v\bar{e}_i, w\bar{e}_i) & \text{otherwise.} \end{cases}$$

proof.

Let $\hat{F}(v, w) = clc(u, v, w)$. Using Eq. (3.15) and the properties of the convolution, we can write

$$clc(u + e_i, v, w) = (\hat{C}_{i+1} * \hat{F})(v, w) =$$

$$= \frac{1}{2}(\hat{F}(v + e_i, w) + \hat{F}(v, w + e_i) + (\hat{C}_i * \hat{F})(v, w) - (\hat{C}_i * \hat{F})(v + e_i, w + e_i)), \quad (3.16)$$

and by the Lemma 3.12, we know that at most one of these terms is nonzero. For the first term to be different from zero, we have that $(v_i, w_i) = (1, 0)$, in this case the expression (3.16) simplifies to

$$clc(u + e_i, v, w) = \frac{1}{2}(clc(u, v + e_i, w)).$$

Likewise, for the second term to be non-zero, we have that $(v_i, w_i) = (0, 1)$, in this case the expression (3.16) simplifies to

$$clc(u + e_i, v, w) = \frac{1}{2}(clc(u, v, w + e_i)).$$

We can join the two expressions into one by writing

$$clc(u + e_i, v, w) = \frac{1}{2}(clc(u, v\bar{e}_i, w\bar{e}_i)),$$

since $v_i \neq w_i$.

For the third term to be different from zero, we must have that $(v_i, w_i) = (0, 0)$, in this case the expression (3.16) simplifies to

$$clc(u + e_i, v, w) = \frac{1}{2}(clc(u + e_i, v, w)).$$

Likewise, for the fourth term to be non-zero, we have that $(v_i, w_i) = (1, 1)$, in this case the expression (3.16) simplifies to

$$clc(u + e_i, v, w) = -\frac{1}{2}(clc(u + e_i, v + e_i, w + e_i)).$$

We can join the two expressions into one by writing

$$clc(u+e_i, v, w) = (-1)^{v_i} clc(u+e_i, v\bar{e}_i, w\bar{e}_i),$$

since $v_i = w_i$. ■

The Lemma 3.13 gives us a recursive way of computing the correlation for any linear approximation to the *carry*. However, there is a more elegant expression by which it is possible to calculate the same value directly. For this, consider the linear approximation $u \xleftarrow{carry} v, w$, as a word composed of octals $x = x_{n-1}...x_0$, where $x_= 4u_i + 2v_i + w_i$. So, we define the following theorem:

THEOREM 3.1 (Linear representation of clc). Let x be the octal word derived from the linear approximation $u \xleftarrow{carry} v, w$. The function $clc(u, v, w)$ can be written as follows

$$clc(u, v, w) = clc(x) = LA_{x_{n-1}}...A_{x_0}C,$$

where $L = (1, 0)$, $C = (1, 1)^t$ and the matrices $A_0, ..., A_7$ are given by

$$A_0 = \tfrac{1}{2}\begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix} \quad A_1 = \tfrac{1}{2}\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad A_2 = \tfrac{1}{2}\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad A_3 = \tfrac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}$$

$$A_4 = \tfrac{1}{2}\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \quad A_5 = \tfrac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad A_6 = \tfrac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad A_7 = \tfrac{1}{2}\begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}$$

proof.

The proof of this theorem can be found in [37]. ■

EXAMPLE 3.1 Consider $u = 10100$, $v = 01110$ e $w = 01000$, in this case we have that $x_0 = 000 = 0$, $x_1 = 010 = 2$, $x_2 = 110 = 6$, $x_3 = 011 = 3$, and $x_4 = 100 = 4$. Therefore, $clc(u, v, w) = LA_4A_3A_6A_2A_0C = -1/8$.

3.5.2.3 Intuitive Representation

It is possible to further simplify the result presented in Theorem 3.1. For that, set $e_0 = \begin{pmatrix} 1 & 0 \end{pmatrix}$ and $e_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}$. Then, it is easy to check that $e_0A_0 = e_0$, $e_0A_4 = e_1$, $e_0A_i = 0$ for $i \neq \{0.4\}$, $e_1A_0 = e_1A_5 = e_1A_6 = \frac{1}{2}e_1$, $e_1A_1 = e_1A_2 = e_1A_4 = \frac{1}{2}e_0$, $e_1A_3 = -\frac{1}{2}e_1$ and $e_1A_7 = -\frac{1}{2}e_0$. Since $L = e_0$ it follows by induction that the only possible values for

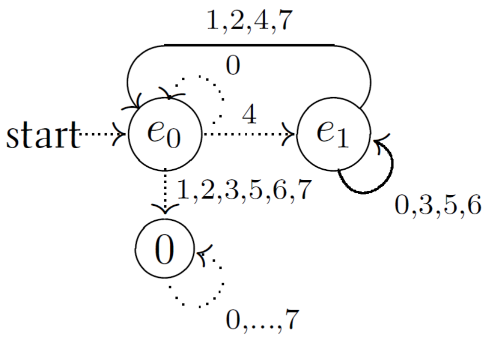


Figure 3.2 – Transition Graph for Theorem 3.1.

$clc(w) = LA_{|w|-1}...A_0C$ are of the form 0, $\pm 2^{-k}e_0$ or $\pm 2^{-k}e_1$, for some $0 \leq k < |w|$.

In fact, we can represent the Theorem 3.1 by the transition graph of Figure 3.2. From these results, it can be seen that clc is equal to zero if there is a transition $1, 2, 3, 5, 6, 7$ starting from e_0. Otherwise, k is given by the number of transitions starting from e_1 and the sign is given by the number of transitions 3 or 7 starting from e_1. For the case of Example 3.1, we have that:

EXAMPLE 3.2 Consider Example 3.1. In this case, we have the following state transitions:

$$e_0 \xrightarrow{4} e_1 \xrightarrow{3} e_1 \xrightarrow{6} e_1 \xrightarrow{2} e_0 \xrightarrow{0} e_0.$$

As there are 3 transitions from e_1, we have $k = 3$. Also, since there is only one transition 3 from e_1, we have a negative sign. Therefore,

$$clc(43620) = -2^{-3}.$$

3.5.2.4 Computing correlations

In this section, we show how to calculate all possible linear approximations such that the correlation is specific, equal to $\pm 2^{-k}$ for some k. For that, let us first define a notation to represent sequences of transitions in the Figure 3.2, which depicts the Transition Graph.

First, a sequence of octals defines the transition sequence, starting at state e_0. For example, 43620 defines the sequence

$$e_0 \xrightarrow{4} e_1 \xrightarrow{3} e_1 \xrightarrow{6} e_1 \xrightarrow{2} e_0 \xrightarrow{0} e_0.$$

If there is more than one transition possibility at a given point, we use the notation $(x_0 + x_1 + ... + x_t)$ to represent the t transition possibilities. For example, $43(5 + 6)20$ defines the sequence

$$e_0 \xrightarrow{4} e_1 \xrightarrow{3} e_1 \xrightarrow{6} e_1 \xrightarrow{2} e_0 \xrightarrow{0} e_0$$

or the sequence

$$e_0 \xrightarrow{4} e_1 \xrightarrow{3} e_1 \xrightarrow{5} e_1 \xrightarrow{2} e_0 \xrightarrow{0} e_0.$$

We also define x^* as a sequence of any length of x transitions. For example, 436^*20 defines the sequence

$$e_0 \xrightarrow{4} e_1 \xrightarrow{3} e_1 \xrightarrow{6} e_1 \xrightarrow{6} ... \xrightarrow{6} e_1 \xrightarrow{2} e_0 \xrightarrow{0} e_0.$$

Finally, we can mix up the notations. For example, $43(5+6)^*20$ denotes the set of sequences that repeats transitions 5 and 6 in any order, an arbitrary number of times, e.g., the following sequence belongs to this set

$$e_0 \xrightarrow{4} e_1 \xrightarrow{3} e_1 \xrightarrow{6} e_1 \xrightarrow{6} e_1 \xrightarrow{5} e_1 \xrightarrow{5} e_1 \xrightarrow{6} e_1 \xrightarrow{2} e_0 \xrightarrow{0} e_0.$$

Using this notation, we can verify, from Figure 3.2, that $clc(w) = 0$ if and only if w is of the form

$$w = (0 + 4(0+3+5+6)^*(1+2+4+7))^*(1+2+3+5+6+7)\Sigma^*,$$

where $\Sigma = (0+1+2+3+4+5+6+7)$.

Let $S^0(n,k)$ and $S^1(n,k)$ be the sets

$$S^0(n,k) = \{w : |w| = n, e_0 A_{w_{n-1}} ... A_{w_0} = \pm 2^{-k} e_0\},$$

$$S^1(n,k) = \{w : |w| = n, e_0 A_{w_{n-1}} ... A_{w_0} = \pm 2^{-k} e_1\}.$$

Therefore, $S^0(n,k) + S^1(n,k)$ determines the set of all sequences of transitions such that $clc(w) = \pm 2^{-k}$. Furthermore, it is easy to see that the following recursive relations hold:

$$S^0(n,k) = S^0(n-1,k)0 + S^1(n-1,k-1)(1+2+4+7),$$

$$S^1(n,k) = S^0(n-1,k)(4) + S^1(n-1,k-1)(0+3+5+6).$$

With these results, we can define the following theorem:

THEOREM 3.2 For any non-empty octal word w, the correlation $clc(w) \in \{0\} \cup \{\pm 2^{-k} : k \in \{0,1,...,|w|-1\}\}$. The set of words of length $n > 0$ with correlations such that

$clc(w) \neq 0$ is given by $S^0(n,k) + S^1(n,k)$, where S^0 and S^1 are defined recursively as follows. First, set $S^0(1,0) = 0$ and $S^1(1,0) = 4$, and $S^0(n,k) = S^1(n,k) = \emptyset$ if $k < 0$ or $k \geq n$. For the other values of k, the recursive relationship

$$S^0(n,k) = S^0(n-1,k)0 + S^1(n-1,k-1)(1+2+4+7),$$

$$S^1(n,k) = S^0(n-1,k)(4) + S^1(n-1,k-1)(0+3+5+6).$$

For every word $w \in S^0(n,k) + S^1(n,k)$, $clc(w) = 2^{-k}$ if and only if w contains an even number of transitions of type $(3,7)$ and $clc(w) = -2^{-k}$, otherwise.

3.5.2.5 Use in Practice

In this section, we present the best linear approximations for *carry* bits (some of these approximations are used later to improve cryptanalysis against ChaCha and Salsa). For this, we use Theorem 3.2. To obtain the best possible correlations, we need small values for k.

PROPOSITION 3.2 The following transition sequences apply:

$$\begin{array}{rcl}
S^0(1,0) &=& 0 \\
S^1(1,0) &=& 4 \\
S^0(n,0) &=& 0^* \\
S^1(n,0) &=& 0^*4 \\
S^0(n,1) &=& 0^*4(1+2+4+7)0^* \\
S^1(n,1) &=& [0^*4(1+2+4+7)0^*4] + [0^*4(0+3+5+6)] \\
\\
S^0(n,2) &=& [0^*4(1+2+4+7)0^*4(1+2+4+7)0^*]+ \\
 && [0^*4(0+3+5+6)(1+2+4+7)0^*] \\
\\
S^1(n,2) &=& [0^*4(1+2+4+7)0^*4(1+2+4+7)0^*4]+ \\
 && [0^*4(0+3+5+6)(1+2+4+7)0^*4]+ \\
 && [0^*4(1+2+4+7)0^*4(0+3+5+6)]+ \\
 && [0^*4(0+3+5+6)(0+3+5+6)]
\end{array}$$

proof.

The proof is left as an exercise for the reader. Just recursively apply the equations of the Theorem 3.2. ■

Proposition 3.2 tells us all possible sequences of transitions for $k = 0, 1, 2$, which corresponds to biases of magnitude $1, 0.5, 0.25$, respectively. From these results, we can build the linear approximations for the *carry* that have greater magnitude.

($\mathbf{clc(w) = 1}$).

To obtain linear equations such that $clc(w) = 1$, we need that $k = 0$. In this case, we use $S^0(n,0)$ and $S^1(n,0)$, presented in Proposition 3.2. We have

$$S^0(n,0) = 0^* \Rightarrow \begin{array}{cccc} u & 0 & \dots & 0 \\ v & 0 & \dots & 0 \\ w & 0 & \dots & 0 \end{array},$$

so $\Pr(0 = 0) = 1$. Another possibility is given by

$$S^1(n,0) = 0^*4 \Rightarrow \begin{array}{ccccc} u & 0 & \dots & 0 & 1 \\ v & 0 & \dots & 0 & 0 \\ w & 0 & \dots & 0 & 0 \end{array},$$

therefore, $\Pr(c_0 = 0) = 1$. Both trivial equations.

($\mathbf{clc(w) = 1/2}$).

To obtain linear equations such that $clc(w) = 1/2$, we need that $k = 1$. In this case, we use $S^0(n, 1)$ and $S^1(n, 1)$, presented in Proposition 3.2. We have

$$S^0(n,1) = 0^*4(1{+}2{+}4{+}7)0^* \Rightarrow \begin{array}{c} u \\ v \\ w \end{array} \quad \begin{array}{cccc} 0 & ... & 0 & 1 \\ 0 & ... & 0 & 0 \\ 0 & ... & 0 & 0 \end{array} \begin{pmatrix} 0 & & 0 & & 1 & & 1 \\ 0 & + & 1 & + & 0 & + & 1 \\ 1 & & 0 & & 0 & & 1 \end{pmatrix} \begin{array}{ccc} 0 & ... & 0 \\ 0 & ... & 0 \\ 0 & ... & 0 \end{array},$$

therefore, we get the following relations

$$\begin{aligned} \Pr(c_i = x_{i-1}) &= 0.75 \\ \Pr(c_i = y_{i-1}) &= 0.75 \\ \Pr(c_i = c_{i-1}) &= 0.75 \\ \Pr(c_i = c_{i-1} \oplus x_{i-1} \oplus y_{i-1}) &= 0.25. \end{aligned}$$

Another possibility is given by $S^1(n, 1) = 0^*4(1+2+4+7)0^*4$, however, this case does not generate anything new since the same previous equations are generated , adding c_0 to the expressions, but $c_0 = 0$. Finally, we have the case where

$$S^1(n,1) = 0^*4(0+3+5+6) \Rightarrow \begin{array}{c} u \\ v \\ w \end{array} \quad \begin{array}{cccc} 0 & ... & 0 & 1 \\ 0 & ... & 0 & 0 \\ 0 & ... & 0 & 0 \end{array} \begin{pmatrix} 0 & & 0 & & 1 & & 1 \\ 0 & + & 1 & + & 0 & + & 1 \\ 0 & & 1 & & 1 & & 0 \end{pmatrix},$$

therefore, we get the following relations

$$\begin{aligned} \Pr(c_1 = 0) &= 0.75 \\ \Pr(c_1 = x_0 \oplus y_0) &= 0.25 \\ \Pr(c_1 = c_0 \oplus x_0) &= 0.75 \\ \Pr(c_1 = c_0 \oplus y_0) &= 0.75. \end{aligned}$$

($\mathbf{clc(w) = 1/4}$).

To obtain linear equations such that $clc(w) = 1/4$, we need that $k = 2$. In this case, we use $S^0(n, 2)$ and $S^1(n, 2)$, presented in Proposition 3.2. We have 6 different cases, the accounts are similar to those presented in the previous section. We present all correlations in Tables 3.1-3.5.

(cls).

To work with approximations for the sum, we use the relation of Eq. 3.10, by which it is clear that it is enough to replace c_i by $s_i \oplus x_i \oplus y_i$, in any one of of the equations presented

Equation	Probability	Condition
$0 = 0$	1	none
$c_0 = 0$	1	none
$c_i = x_{i-1}$	0.75	$i > 0$
$c_i = y_{i-1}$	0.75	$i > 0$
$c_i = c_{i-1}$	0.75	$i > 0$
$c_i = x_{i-1} \oplus y_{i-1} \oplus c_{i-1}$	0.25	$i > 0$
$c_1 = 0$	0.75	none
$c_1 = x_0 \oplus y_0$	0.25	none
$c_1 = x_0$	0.75	none
$c_1 = y_0$	0.75	none

Table 3.1 – Equations generated from $S^0(n, 0)$, $S^1(n, 0)$, $S^0(n, 1)$ e $S^1(n, 1)$.

in Tables 3.1-3.5, to obtain the desired result.

(clsub).

To work with approximations for subtraction, we use the relation of Eq. 3.11. Here the substitution is a little more complex, but it can be done by replacing y_i by y'_i, x_i for m_i and c_i for $m_i \oplus x'_i \oplus y'_i$, in the equations presented in the Tables 3.1-3.5.

(clcarrysub).

To work with approximations for subtraction, we use the relation of Eq. 3.12. The equations can be obtained by replacing x_i for $x_i \oplus \pi_i$ and c_i for π_i, in the equations presented in Tables 3.1-3.5.

Equation	Probability
$c_i \oplus x_{i-1} \oplus c_j \oplus x_{j-1} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_j \oplus y_{j-1} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_j \oplus c_{j-1} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_j \oplus x_{j-1} \oplus y_{j-1} \oplus c_{j-1} = 0$	0.325
$c_i \oplus y_{i-1} \oplus c_j \oplus x_{j-1} = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_j \oplus y_{j-1} = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_j \oplus c_{j-1} = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_j \oplus x_{j-1} \oplus y_{j-1} \oplus c_{j-1} = 0$	0.325
$c_i \oplus c_{i-1} \oplus c_j \oplus x_{j-1} = 0$	0.625
$c_i \oplus c_{i-1} \oplus c_j \oplus y_{j-1} = 0$	0.625
$c_i \oplus c_{i-1} \oplus c_j \oplus c_{j-1} = 0$	0.625
$c_i \oplus c_{i-1} \oplus c_j \oplus x_{j-1} \oplus y_{j-1} \oplus c_{j-1} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_j \oplus x_{j-1} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_j \oplus y_{j-1} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_j \oplus c_{j-1} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_j \oplus x_{j-1} \oplus y_{j-1} \oplus c_{j-1} = 0$	0.625

Table 3.2 – Equations generated from $S^0(n, 2) = 0^*4(1 + 2 + 4 + 7)0^*4(1 + 2 + 4 + 7)0^*$ or $S^1(n, 2) = 0^*4(1 + 2 + 4 + 7)0^*4(1 + 2 + 4 + 7)0^*4$, subject to $i - j > 2, j > 0$.

Equation	Probability
$c_i \oplus x_{i-2} = 0$	0.625
$c_i \oplus y_{i-2} = 0$	0.625
$c_i \oplus c_{i-2} = 0$	0.625
$c_i \oplus x_{i-2} \oplus y_{i-2} \oplus c_{i-2} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus x_{i-2} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus y_{i-2} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-2} = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus x_{i-2} \oplus y_{i-2} \oplus c_{i-2} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_{i-1} \oplus x_{i-2} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_{i-1} \oplus y_{i-2} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_{i-1} \oplus c_{i-2} = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_{i-1} \oplus x_{i-2} \oplus y_{i-2} \oplus c_{i-2} = 0$	0.325
$c_i \oplus y_{i-1} \oplus c_{i-1} \oplus x_{i-2} = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_{i-1} \oplus y_{i-2} = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_{i-1} \oplus c_{i-2} = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_{i-1} \oplus x_{i-2} \oplus y_{i-2} \oplus c_{i-2} = 0$	0.325

Table 3.3 – Equations generated from $S^0(n, 2) = 0^*4(0 + 3 + 5 + 6)(1 + 2 + 4 + 7)0^*$ or $S^1(n, 2) = 0^*4(0 + 3 + 5 + 6)(1 + 2 + 4 + 7)0^*4$, subject to $i > 1$.

Equation	Probability
$c_i \oplus x_{i-1} \oplus c_1 = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_1 \oplus x_0 \oplus y_0$=0	0.325
$c_i \oplus x_{i-1} \oplus c_1 \oplus x_0 = 0$	0.625
$c_i \oplus x_{i-1} \oplus c_1 \oplus y_0 = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_1 = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_1 \oplus x_0 \oplus y_0 = 0$	0.325
$c_i \oplus y_{i-1} \oplus c_1 \oplus x_0 = 0$	0.625
$c_i \oplus y_{i-1} \oplus c_1 \oplus y_0 = 0$	0.625
$c_i \oplus c_{i-1} \oplus c_1 = 0$	0.625
$c_i \oplus c_{i-1} \oplus c_1 \oplus x_0 \oplus y_0 = 0$	0.325
$c_i \oplus c_{i-1} \oplus c_1 \oplus x_0 = 0$	0.625
$c_i \oplus c_{i-1} \oplus c_1 \oplus y_0 = 0$	0.625
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_1 = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_1 \oplus x_0 \oplus y_0 = 0$	0.625
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_1 \oplus x_0 = 0$	0.325
$c_i \oplus x_{i-1} \oplus y_{i-1} \oplus c_{i-1} \oplus c_1 \oplus y_0 = 0$	0.325

Table 3.4 – Equations generated from $S^1(n, 2) = 0$*$4(1 + 2 + 4 + 7)0$*$4(0 + 3 + 5 + 6)$, subject to $i > 2$.

Equation	Probability
$c_2 = 0$	0.625
$c_2 \oplus x_0 \oplus y_0 = 0$	0.325
$c_2 \oplus x_0 = 0$	0.625
$c_2 \oplus y_0 = 0$	0.625
$c_2 \oplus x_1 \oplus y_1 = 0$	0.325
$c_2 \oplus x_1 \oplus y_1 \oplus x_0 \oplus y_0 = 0$	0.625
$c_2 \oplus x_1 \oplus y_1 \oplus x_0 = 0$	0.325
$c_2 \oplus x_1 \oplus y_1 \oplus y_0 = 0$	0.325
$c_2 \oplus x_1 \oplus c_1 = 0$	0.625
$c_2 \oplus x_1 \oplus c_1 \oplus x_0 \oplus y_0 = 0$	0.325
$c_2 \oplus x_1 \oplus c_1 \oplus x_0 = 0$	0.625
$c_2 \oplus x_1 \oplus c_1 \oplus y_0 = 0$	0.625
$c_2 \oplus y_1 \oplus c_1 = 0$	0.625
$c_2 \oplus y_1 \oplus c_1 \oplus x_0 \oplus y_0 = 0$	0.325
$c_2 \oplus y_1 \oplus c_1 \oplus x_0 = 0$	0.625
$c_2 \oplus y_1 \oplus c_1 \oplus y_0 = 0$	0.625

Table 3.5 – Equations generated from $S^1(n, 2) = 0$*$4(0 + 3 + 5 + 6)(0 + 3 + 5 + 6)$, subject to $i > 2$.

Part II

New Tools to Evaluate Diffusion in Symmetric primitives

4 CONTINUOUS DIFFUSION ANALYSIS

4.1 INTRODUCTION

A symmetric cryptographic algorithm needs to have several properties to be considered secure. Two of the most important of such properties were named *confusion* and *diffusion* by Claude Shannon [3] back in 1949. Shannon defined confusion as the capacity of an algorithm to create a very complex and involved relationship between the key and the ciphertext and diffusion as the property that the redundancy in the statistics of the plaintext is “dissipated” into the statistics of the ciphertext. These are, however, very abstract concepts and since then several metrics were proposed to try to measure the confusion and diffusion of algorithms [1, 4, 5].

Cryptographic Boolean functions [6] contain important properties that can be used, at least in theory, to access the confusion and diffusion of algorithms. In designing a cryptographic algorithm, we often need functions that satisfy requirements such as balance, high nonlinearity, high algebraic degree, and good avalanche characteristics.

In particular, the *Strict Avalanche Criterion* (SAC) proposed by Webster and Tavares [4] is an important tool for studying the diffusion of algorithms. A Boolean function is said to satisfy the SAC if complementing a single bit in its input results, then the output of the function being complemented with a probability of a half. After its proposal, several authors further generalized the SAC, Forrié proposed the SAC of order k [43], Preneel *et al.* proposed the propagation criterion [5], among others [44], [45].

Unfortunately, for modern ciphers, it is not possible to actually prove these properties for each output bit whereas the Boolean functions generated cannot be explicitly derived. Thus, these properties are estimated by randomly generating many inputs to the cipher and then changing input bits to see the behavior of the output bits empirically [1, 7]. However, using this kind of metrics is not actually useful to access the security of algorithms because even weak and broken ciphers tend to have good results in such empirical tests.

In this chapter, we propose a new technique, called Continuous Diffusion Analysis (CDA), which can be used to study the diffusion of cryptographic algorithms. The main idea of CDA is to generalize cryptographic operations allowing to express the individual bits as probabilities, effectively creating a continuous generalization of the algorithm itself. In this way, we can also generalize the SAC, because we can change the input not only by flipping bits but also by inserting very small correlations in a particular continuous bit. Moreover, we propose three new metrics to measure the diffusion in this generalized continuous space, namely the Continuous Avalanche Factor, the Continuous Neutrality Measure, and the Diffusion Factor. In particular, for the best of our knowledge, the Diffusion Factor is the first metric capable of measuring the diffusion of a modern secure cipher without considering a reduction with fewer rounds or a sample subset of input bits.

This chapter is organized as follows. In Section 4.2, we provide some basic definitions and results of previous works. In Section 4.3, we present the mathematical background of CDA and how to generalize cryptographic algorithms using continuous operations. Additionally, in Section 4.3, we propose three new diffusion metrics, namely the Continuous Avalanche Factor, the Continuous Neutrality Measure, and the Diffusion Factor. In Section 4.4, we present a case study of the proposed techniques and metrics by comparing the algorithms Salsa, ChaCha, AES, and Speck.

4.2 MEASURING THE AVALANCHE EFFECT

The avalanche effect is a desirable cryptographic property in which a small change in the input should result in big changes in the output. More precisely, for a given transformation to exhibit the avalanche effect, an average of one half of the output bits should change whenever a single input bit is flipped [1].

DEFINITION 4.1 (Hamming Distance) Let $x \in \mathbb{F}_2^n$ and let x_i denote the i-th bit of x. The hamming weight of x is defined as

$$w_H(x) = |\{i : x_i = 1\}|,$$

also let $y \in \mathbb{F}_2^n$ then the hamming distance between X and Y is defined as

$$d_H(x, y) = w_H(x \oplus y).$$

DEFINITION 4.2 (Avalanche Factor (AF)) For a function $f : M \to M$ defined on some finite metric space (M, d), the avalanche factor of f with respect to d and an input distance of $\delta \in \mathbb{R}$ is

$$\mathcal{AF}(f, d|\delta) = \frac{\sum_{d(x,y)=\delta} d(f(x), f(y))}{|\{(x, y)|d(x, y) = \delta\}|} \times \frac{|M|^2}{\sum_{x,y} d(x, y)}.$$

If $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $d(x, y) = d_H(x, y)$, then one can show that

$$\mathcal{AF}(f, d_H|\delta) = \frac{2}{n} \frac{\sum_{d(x,y)=\delta} d(f(x), f(y))}{|\{(x, y)|d(x, y) = \delta\}|}.$$

4.3 CONTINUOUS DIFFUSION ANALYSIS

The main idea behind CDA is to consider each bit in a cryptographic algorithm as a real number. Also, we consider sets of bits, like integers, as arrays of real numbers. In this section, we show how to generalize bitwise operations like XOR to continuous functions. To do that, we use the laws of probabilities and derive some basic equations that allow us to generalize several cryptographic algorithms. From that, we define metrics to measure the diffusion in this generalized continuous space.

4.3.1 Motivation

The classical techniques for measuring diffusion have a critical limitation that they are limited in the number of rounds. This happens because at a certain number of rounds the output of the algorithm becomes random enough that becomes impossible to detect any kind of improvement. Here, we want to find a way to overcome this limitation to be able to compare different algorithms with an arbitrary number of rounds.

4.3.2 Continuous Generalizations

In this Section, we show how to generalize cryptographic operations as continuous functions. Our goal is to achieve a generalization of common cryptographic boolean functions in a way that we can work in a continuous space instead of binary boolean states. To do so, we like to achieve the following:

1. Define the set $\mathcal{B}$ where the generalized bits live.
2. Define a function $\phi : \mathbb{F}_2^n \to \mathcal{B}^n$ to transition from binary states to continuous states.
3. Define a function $\delta : \mathcal{B}^m \to \mathbb{F}_2^m$ to transition from continuous states to binary states.
4. Mathematically define how the generalization should behave. In particular, define the property that given binary states as input of the generalization should return only binary states and the result should be exactly the same as of the original function.

To do so, we first define what we mean by a continuous generalization of a function.

Definition 4.3 (Continuous Generalization of a function) Let $\mathcal{B} = \{x \in \mathbb{R} : -1 \leq x \leq 1\}$ and $\phi : \mathbb{F}_2^n \to \mathcal{B}^n$ be a function defined as $\phi(x) = (2x_0 - 1, ..., 2x_{n-1} - 1)$ where $x \in \mathbb{F}_2^n$ and x_i is the i-th bit of x. Also, let $\delta : \mathcal{B}^m \to \mathbb{F}_2^m$ be a function defined as

$$\delta(y) = \left(\frac{1}{2}(\text{sgn}(y_0) + 1), ..., \frac{1}{2}(\text{sgn}(y_{m-1}) + 1))\right)$$

where $y \in \mathcal{B}^m$, y_i denote the i-th element of y, and $\text{sgn}(.)$ is the sign function. Hence, we call $f_\mathcal{C} : \mathcal{B}^n \to \mathcal{B}^m$ a Continuous Generalization of the function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, denoted by $f_\mathcal{C} \overset{\sim}{\leftarrow} f$, if for all $x \in \mathbb{F}_2^n$, it holds that

$$(y_0, ..., y_{m-1}) = f_\mathcal{C}(\phi(x)),$$

such that $y_i = \pm 1$, for all i, and

$$f(x) = \delta(f_\mathcal{C}(\phi(x))).$$

Next, we define three important Lemmas.

Lemma 4.1 Let $x \in \mathbb{F}_2^n$ and $y \in \mathcal{B}^n$, and define $\text{ID} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as the identity function $\text{ID}(x) = x$ and $\text{ID}_\mathcal{C} : \mathcal{B}^n \to \mathcal{B}^n$ as $\text{ID}_\mathcal{C}(y) = y$. Then, it holds that $\text{ID}_\mathcal{C} \overset{\sim}{\leftarrow} \text{ID}$.

proof.

First, $\text{ID}_\mathcal{C}(\phi(x)) = \text{ID}_\mathcal{C}(2x_0 - 1, ..., 2x_{n-1} - 1) = (2x_0 - 1, ..., 2x_{n-1} - 1)$. Thus, we have $2x_i - 1 = \pm 1$, for all i. Also, whereas $\delta(\phi(1)) = 1$ and $\delta(\phi(0)) = 0$, we have that $\delta(\text{ID}_\mathcal{C}(\phi(x))) = \delta(\phi(x)) = x$. Therefore, $\text{ID}_\mathcal{C} \overset{\sim}{\leftarrow} \text{ID}$. ■

Lemma 4.2 Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $f_\mathcal{C} : \mathcal{B}^n \to \mathcal{B}^m$, $g : \mathbb{F}_2^m \to \mathbb{F}_2^s$, and $g_\mathcal{C} : \mathcal{B}^m \to \mathcal{B}^s$, such that $f_\mathcal{C} \overset{\sim}{\leftarrow} f$ and $g_\mathcal{C} \overset{\sim}{\leftarrow} g$, then it holds that

$$g_\mathcal{C} \circ f_\mathcal{C} \overset{\sim}{\leftarrow} g \circ f.$$

proof.

It is easy to see that if we have all possible $y \in \mathbb{F}_2^m$, then by the definition of ϕ, it follows that $\phi(y)$ generates all possible arrays of size m composed of ± 1 elements. From this observation and by the definition of $g_\mathcal{C} \overset{\sim}{\leftarrow} g$, we have that $g_\mathcal{C}(\phi(y))$ produces arrays of length s with ± 1 elements whenever it receives as input any array of length m with only ± 1 elements. Since $f_C \overset{\sim}{\leftarrow} f$, by definition $f_C(\phi(x))$ only outputs arrays of size m with ± 1 elements, it holds that $(z_0, ..., z_{s-1}) = g_C(f_C(\phi(x)))$, such that $z_i = \pm 1$, for any $x \in \mathbb{F}_2^n$. It remains to show that $g(f(x)) = \delta(g_C(f_C(\phi(x))))$. Using the fact that $f(x) = \delta(f_C(\phi(x)))$ and $g(x) = \delta(g_C(\phi(x)))$, we can rewrite the left side of the equation as

$$g(f(x)) = g(\delta(f_C(\phi(x)))) = \delta(g_C(\phi(\delta(f_C(\phi(x)))))).$$

The proof concludes by using the fact that $\phi(\delta(x)) = x$ whenever $x = \pm 1$. ■

Lemma 4.3 Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $g : \mathbb{F}_2^{n+s} \to \mathbb{F}_2^{m+s}$ defined as $g(x, y) = (ID(x), f(y))$, where $x \in \mathbb{F}_2^s$ and $y \in \mathbb{F}_2^n$. Also, let $f_C : \mathcal{B}^n \to \mathcal{B}^m$ and $g_C : \mathcal{B}^{n+s} \to \mathcal{B}^{m+s}$ defined as $g_C(\tilde{x}, \tilde{y}) = (ID_C(\tilde{x}), f_C(\tilde{y}))$, where $\tilde{x} \in \mathcal{B}^s$ and $\tilde{y} \in \mathcal{B}^n$. Hence, $g_C \overset{\sim}{\leftarrow} g$ if and only if $f_C \overset{\sim}{\leftarrow} f$.

proof.

First, suppose that $f_C \overset{\sim}{\leftarrow} f$, then we have that $g_C(\phi(x), \phi(y)) = (ID_C(\phi(x)), f_C(\phi(y)))$, which by definition, only produce ± 1 elements. Additionally, $\delta(g_C(\phi(x), \phi(y))) = \delta(ID_C(\phi(x)), f_C(\phi(y))) = (x, \delta(f_C(\phi(y)))) = (x, f(y))$, therefore, $g_C \overset{\sim}{\leftarrow} g$. Conversely, suppose that $g_C \overset{\sim}{\leftarrow} g$, then it must be the case that $g_C(\phi(x), \phi(y))$ only output arrays with ± 1 elements. However, $g_C(\phi(x), \phi(y)) = (ID_C(\phi(x)), f_C(\phi(y)))$, thus, $f_C(\phi(y))$ also must output arrays with ± 1 elements. Additionally, $g(x, f(y)) = \delta(g_C(ID_C(\phi(x)), \phi(y))) = \delta(\phi(x), f_C(\phi(y))) = (x, \delta(f_C(\phi(y))))$, thus $\delta(f_C(\phi(y))) = f(y)$ and $f_C \overset{\sim}{\leftarrow} f$.
■

The majority of symmetric cryptographic algorithms only use operations like addition, rotation, Boolean operations (XOR, AND, OR), bitwise permutations, and S-boxes. Some examples are ARX and AND-RX algorithms like Salsa [8], ChaCha [10], Speck [29], Sparx [46], Blake [47], SHA-1, SHA-2 [48], and SPNs like AES [30]. Any of these algorithms can be viewed as a composition of these basic functions, therefore, Lemma 4.2 guarantees that replacing theses operations by their continuous counterparts will generate a continuous generalization of the algorithm itself.

To generalize cryptographic operations, we use probability theory and define each bit as a probability. Thus, we imagine that the bits are not only zeros and ones but are some kind of superposition of both states. More precisely, let $\Pr(E)$ be the probability of occurrence of an event E and $b \in \mathbb{F}_2$ be a bit, then we can write

$$\Pr(b = 1) = p = \frac{1}{2}(1 + \varepsilon). \tag{4.1}$$

We refer to p and ε as the probability and the correlation of the bit b, respectively. We also refer to ε as the *continuous deviation* of the bit b, we prefer this term since the results of the continuous generalizations should not be understood as a probability. In fact, the proposed technique assumes independence in situations where this hypothesis does not hold, therefore, what we get as result is not close at all to the real probability. However, as we shall demonstrate the proposed metrics remains useful as a diffusion measuring tool. We say that a bit is *discrete* when it acts as a classical bit, i.e., when either $\Pr(b = 1) = 1$ or $\Pr(b = 1) = 0$, note that in these cases, we have that $\varepsilon = \pm 1$. Conversely, we say that the bit is *continuous* when $-1 < \varepsilon < 1$. Using probability theory, all values we can get for any continuous bit are always between 0 and 1. Generally, we will represent continuous bits as the continuous deviation because $\varepsilon \in \mathcal{B}$, which will fit better with Definition 4.3.

For a moment, consider the XOR operation $b_3 = b_1 \oplus b_2$. We know that b_3 is equal to 1 either when $(b_1 = 0) \wedge (b_2 = 1)$ or when $(b_1 = 1) \wedge (b_2 = 0)$, therefore, if we consider $b_i \sim \text{Bernoulli}(p_i)$ being b_1 and b_2 independent random variables, we get that $p_3 = p_1(1 - p_2) + p_2(1 - p_1)$. We can also write this relationship in terms of the continuous deviation, obtaining $\varepsilon_3 = -\varepsilon_1\varepsilon_2$. Thus, we could define the function $x \oplus_{\mathcal{C}} y = -xy$, where $x, y \in \mathcal{B}$. One can verify that $\phi(x) \oplus_{\mathcal{C}} \phi(y) = \pm 1$ and that $x \oplus y = \delta(\phi(x) \oplus_{\mathcal{C}} \phi(y))$, thus $\oplus_{\mathcal{C}} \overset{\sim}{\leftarrow} \oplus$. More generally, we can work with an arbitrary Boolean function.

LEMMA 4.4 Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. If $X \in \mathbb{F}_2^n$ and $X = (X_0, ..., X_{n-1})$ such that $X_i \sim \text{Bernoulli}(p_i)$ are independent random variables, and denote $\varepsilon_i = 2p_i - 1$, then it holds that

$$\Pr(f(X) = 1) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} f(x) \prod_{i=0}^{n-1} (1 - (-1)^{x_i} \varepsilon_i).$$

proof.

From the law of total probability, we have that

$$\Pr(f(X) = 1) = \sum_{x \in \mathbb{F}_2^n} \Pr(f(x) = 1 | X = x) \Pr(X = x)$$

$$= \sum_{x \in \mathbb{F}_2^n} f(x) \Pr(X = x).$$

Since X_i are independent Bernoulli random variables it follows that

$$\Pr(X = x) = \Pr((X_{n-1} = x_{n-1}) \cap ... \cap (X_0 = x_0))$$

$$= \prod_{i=0}^{n-1} \Pr(X_i = x_i) = \prod_{i=0}^{n-1} p_i^{x_i}(1-p_i)^{1-x_i},$$

therefore, we have

$$\Pr(f(X)=1) = \sum_{x\in\mathbb{F}_2^n} f(x) \prod_{i=0}^{n-1} p_i^{x_i}(1-p_i)^{1-x_i}$$

$$= \frac{1}{2^n} \sum_{x\in\mathbb{F}_2^n} f(x) \prod_{i=0}^{n-1} (1-(-1)^{x_i}\varepsilon_i).$$

■

THEOREM 4.1 Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function and $f_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}$ be defined as

$$f_{\mathcal{C}}(x) = \left[\frac{1}{2^{n-1}} \sum_{y \in \mathbb{F}_2^n} f(y) \prod_{i=0}^{n-1} (1 - (-1)^{y_i} x_i) \right] - 1,$$

where $x = (x_0, ..., x_{n-1})$, then $f_{\mathcal{C}} \overset{\sim}{\leftarrow} f$.

proof.

First, by the definition of ϕ, we can write

$$f_{\mathcal{C}}(\phi(x))$$

$$= \left[\frac{1}{2^{n-1}} \sum_{y \in \mathbb{F}_2^n} f(y) \prod_{i=0}^{n-1} (1 - (-1)^{y_i} (2x_i - 1)) \right] - 1.$$

If we define $\varepsilon_i = 2x_i - 1$ and consider Lemma 4.4, we have that $f_{\mathcal{C}}(\phi(x)) = 2\Pr(f(Y) = 1) - 1$, when $Y = (Y_0, ..., Y_{n-1})$ and $Y_i \sim \text{Bernoulli}(x_i)$. Like in the proof of Lemma 4.4, we can use the law of total probability and write

$$\Pr(f(Y) = 1) = \sum_{y \in \mathbb{F}_2^n} f(y) \Pr(Y = y),$$

but since x_i is either zero or one, then $\Pr(Y = y)$ is one when $y = x$ and zero otherwise. Therefore, $\Pr(f(Y) = 1) = f(x)$. Since $f(x)$ is a Boolean function, it can only output zero or one, thus, $f_{\mathcal{C}}(\phi(x)) = \pm 1$. Finally, we get

$$\delta(f_{\mathcal{C}}(\phi(x))) = \delta(2f(x) - 1)$$

$$= (\text{sgn}(2f(x) - 1) + 1)/2 = f(x).$$

■

We are now able to define several continuous generalizations.

DEFINITION 4.4 (Continuous generalization of XOR) Let $x, y \in \mathcal{B}$, then we define the continuous XOR function as

$$x \oplus_{\mathcal{C}} y = -xy.$$

Definition 4.5 (Continuous generalization of AND) Let $x, y \in \mathcal{B}$, then we define the continuous AND function as

$$x \wedge_{\mathcal{C}} y = \frac{xy + x + y - 1}{2}.$$

Definition 4.6 (Continuous generalization of OR) Let $x, y \in \mathcal{B}$, then we define the continuous OR function as

$$x \vee_{\mathcal{C}} y = \frac{x + y + 1 - xy}{2}.$$

Definition 4.7 (Continuous generalization of NOT) Let $x \in \mathcal{B}$, then we define the continuous NOT function as

$$\neg_{\mathcal{C}} x = -x.$$

Definition 4.8 (Continuous generalization of MAJ) Let $\mathrm{MAJ}(a, b, c) = ab \oplus ac \oplus bc$ with $a, b, c \in \mathbb{F}_2$, and let $x, y, z \in \mathcal{B}$, then we define the continuous MAJ function as

$$\mathrm{MAJ}_{\mathcal{C}}(x, y, z) = \frac{1}{2}\left(x + y + z - xyz\right).$$

As a consequence of Theorem 4.1, we can write the following corollary.

Corollary 4.1 It holds that $\oplus_{\mathcal{C}} \overset{\sim}{\leftarrow} \oplus$, $\wedge_{\mathcal{C}} \overset{\sim}{\leftarrow} \wedge$, $\vee_{\mathcal{C}} \overset{\sim}{\leftarrow} \vee$, $\neg_{\mathcal{C}} \overset{\sim}{\leftarrow} \neg$, and $\mathrm{MAJ}_{\mathcal{C}} \overset{\sim}{\leftarrow} \mathrm{MAJ}$.

If we consider X and Y as n-bit words, whereas the XOR, AND, OR, and NOT are computed bit by bit, then we can use continuous operations from Definitions 4.4-4.7 for each bit individually, under the assumption of independence between each bit pair x_i and y_i.

Definition 4.9 (Continuous generalization of Shift and Rotation) Let $x = (x_0, ..., x_{n-1}) \in \mathcal{B}^n$ and $r \in \mathbb{Z}$, such that $0 \leq r \leq n-1$, then we define the continuous shift to the left, continuous shift to the right, continuous rotation to the left and continuous rotation to the right, respectively, as

$$(x_0, ..., x_{n-1}) \ll_C r = (-1, ..., -1, x_0, ..., x_{n-1-r})$$

$$(x_0, ..., x_{n-1}) \gg_C r = (x_r, ..., x_{n-1}, -1, ..., -1)$$

$$(x_0, ..., x_{n-1}) \lll_C r = (x_{n-r}, ..., x_{n-1}, x_0, ..., x_{n-1-r})$$

$$(x_0, ..., x_{n-1}) \ggg_C r = (x_r, ..., x_{n-1}, x_0, ..., x_{r-1}).$$

PROPOSITION 4.1 It holds that $\ll_{\mathcal{C}} \overset{\sim}{\leftarrow} \ll$, $\gg_{\mathcal{C}} \overset{\sim}{\leftarrow} \gg$, $\lll_{\mathcal{C}} \overset{\sim}{\leftarrow} \lll$, and $\ggg_{\mathcal{C}} \overset{\sim}{\leftarrow} \ggg$.

proof.

Let $x \in \mathbb{F}_2^n$, then $\phi(x) \ll_{\mathcal{C}} r$ only outputs ± 1 values. Additionally, we have that

$$\delta(\phi(x) \ll_{\mathcal{C}} r)$$

$$= (\delta(-1), ..., \delta(-1), \delta(\phi(x_0)), ..., \delta(\phi(x_{n-r-1}))$$

$$= (0, ..., 0, x_0, ..., x_{n-r-1}) = x \ll r,$$

whereas $\delta(\phi(y)) = y$ whether y is zero or one. Thus, $\ll_{\mathcal{C}} \overset{\sim}{\leftarrow} \ll$. The proof for the remaining operations is analogous. ■

Next, we define the generalize addition modulo 2^n.

DEFINITION 4.10 (Continuous generalization of addition modulo 2^n) Let $x, y \in \mathcal{B}^n$, then we define the continuous addition modulo 2^n function as

$$x \boxplus_{\mathcal{C}} y = (z_0, ..., z_{n-1}),$$

where z_i is given recursively as follow

$$\begin{array}{rcl} c_0 & = & -1, \\ z_i & = & x_i \oplus_{\mathcal{C}} y_i \oplus_{\mathcal{C}} c_i, \\ c_{i+1} & = & \mathrm{MAJ}_{\mathcal{C}}(x_i, y_i, c_i). \end{array}$$

PROPOSITION 4.2 It holds that $\boxplus_{\mathcal{C}} \overset{\sim}{\leftarrow} \boxplus$.

proof.

Let $x, y, z, c \in \mathbb{F}_2^n$, $\tilde{x}, \tilde{y}, \tilde{c} \in \mathcal{B}^n$. It is well-known that the addition between x and y can be seen as $z = x \boxplus y = x \oplus y \oplus c$, where c denote the carry bits, also that z_i can be computed recursively as

$$\begin{aligned} c_0 &= 0, \\ z_i &= x_i \oplus y_i \oplus c_i, \\ c_{i+1} &= \mathrm{MAJ}(x_i, y_i, c_i). \end{aligned}$$

Clearly, the equations from Definition 4.10 are continuous generalizations of the equations above because they apply Definitions 4.4 and 4.8 directly. The only question that remains is if the recursion on c_i destroys the properties imposed by Definition 4.3. To see why this is not the case, define the function $g_i(x, y, \theta) = (x, y, \mathrm{MAJ}(x_i, y_i, \theta))$, for $i = 0, 1, ..., n-1$, then we can write

$$g_i \circ g_{i-1} \circ \cdots \circ g_0(x, y, 0) = (x, y, \mathrm{MAJ}(x_i, y_i, c_i))$$

$$= (x, y, c_{i+i}).$$

Now, using Lemma 4.3, we get that $g_{\mathcal{C}_i} \overset{\sim}{\leftarrow} g_i$, where $g_{\mathcal{C}_i}(\tilde{x}, \tilde{y}, \theta) = (\tilde{x}, \tilde{y}, \mathrm{MAJ}_{\mathcal{C}}(\tilde{x}_i, \tilde{y}_i, \theta))$, also, using Lemma 4.2, we get that

$$g_{\mathcal{C}_i} \circ g_{\mathcal{C}_{i-1}} \circ \cdots \circ g_{\mathcal{C}_0}(x, y, 0) \overset{\sim}{\leftarrow} g_i \circ g_{i-1} \circ \cdots \circ g_0(x, y, 0),$$

therefore, $\mathrm{MAJ}_{\mathcal{C}}(\tilde{x}_i, \tilde{y}_i, \tilde{c}_i) \overset{\sim}{\leftarrow} \mathrm{MAJ}(x_i, y_i, c_i)$, for all $i = 0, 1, ..., n-1$. ■

DEFINITION 4.11 (Continuous generalization of bitwise permutations) Let $x = (x_0, ..., x_{n-1}) \in \mathcal{B}^n$ and π be a permutation of n elements. Then we define the continuous bitwise permutation as

$$P_{\mathcal{C}}^{\pi}(x) = P_{\mathcal{C}}^{\pi}(x_0, ..., x_{n-1}) = (x_{\pi(0)}, x_{\pi(1)}, ..., x_{\pi(n-1)}).$$

PROPOSITION 4.3 It holds that $P_{\mathcal{C}}^{\pi} \overset{\sim}{\leftarrow} P^{\pi}$, where

$$P^{\pi}(x_0, ..., x_{n-1}) = (x_{\pi(0)}, x_{\pi(1)}, ..., x_{\pi(n-1)})$$

with $x \in \mathbb{F}_2^n$, and $P_{\mathcal{C}}^{\pi}$ is given by Definition 4.11.

proof.

Given $x \in \mathbb{F}_2^n$, then we have

$$P_{\mathcal{C}}^{\pi}(\phi(x)) = (2x_{\pi(0)} - 1, ..., 2x_{\pi(n-1)} - 1),$$

which is an array of ± 1 elements. Also,

$$\delta(P_{\mathcal{C}}^{\pi}(\phi(x))) = (\delta(\phi(x_{\pi(0)})), ..., \delta(\phi(x_{\pi(n-1)}))).$$

Since $\delta(\phi(y)) = y$ when y is zero or one, then we have $\delta(P_{\mathcal{C}}^{\pi}(\phi(x))) = (x_{\pi(0)}, ..., x_{\pi(n-1)}) = P^{\pi}(x)$. Therefore, $P_{\mathcal{C}}^{\pi} \overset{\sim}{\leftarrow} P^{\pi}$. ■

Definition 4.12 (Continuous generalization of S-boxes) Let $x = (x_0, x_1, ..., x_{n-1}) \in \mathcal{B}^n$ and $f_{i_{\mathcal{C}}} : \mathcal{B}^n \to \mathcal{B}$ for $i = (0, 1, ..., n-1)$, then we define a continuous S-box $S_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ as

$$S_{\mathcal{C}}(x) = (f_{0_{\mathcal{C}}}(x), f_{1_{\mathcal{C}}}(x), ..., f_{m-1_{\mathcal{C}}}(x)).$$

Proposition 4.4 For any S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is possible to define a function $S_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ of the form

$$S_{\mathcal{C}}(x) = (f_{0_{\mathcal{C}}}(x), f_{1_{\mathcal{C}}}(x), ..., f_{m-1_{\mathcal{C}}}(x)),$$

where $f_{i_{\mathcal{C}}} : \mathcal{B}^n \to \mathcal{B}$ for $i = (0, 1, ..., m-1)$, such that $S_{\mathcal{C}} \overset{\sim}{\leftarrow} S$.

proof.

It is a well-known fact that any S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, can be represented by m Boolean functions [49, 50]. Thus, we can write $S(x) = f_0(x)||...||f_{m-1}(x)$. Using Theorem 4.1, we can define $f_{i_{\mathcal{C}}} : \mathcal{B}^n \to \mathcal{B}$, for $i = (0, 1, ..., m-1)$, such that $f_{i_{\mathcal{C}}} \overset{\sim}{\leftarrow} f_i$, and write the function $S_{\mathcal{C}}(x) = (f_{0_{\mathcal{C}}}(x), f_{1_{\mathcal{C}}}(x), ..., f_{m-1_{\mathcal{C}}}(x))$. Now suppose that $S_{\mathcal{C}}$ is not a continuous generalization of S. Then, there must exist x such that $S_{\mathcal{C}}(\phi(x))$ is not an array of ± 1 elements nor that $S(x) \neq \delta(S_{\mathcal{C}}(\phi(x)))$. Since we can write $S_{\mathcal{C}}(\phi(x)) = (f_{0_{\mathcal{C}}}(\phi(x)), ..., f_{m-1_{\mathcal{C}}}(\phi(x)))$, then it must exist at least one $x \in \mathbb{F}_2^n$ such that we have $f_{i_{\mathcal{C}}}(\phi(x)) \neq \pm 1$ or $f_i(x) \neq \delta(f_{i_{\mathcal{C}}}(\phi(x)))$ for some i, which are both contradictions. Therefore, $S_{\mathcal{C}} \overset{\sim}{\leftarrow} S$. ■

4.3.3 Continuous Diffusion Metrics

At this point, the reader might be asking himself what the utility is in creating continuous generalizations of cryptographic algorithms. As shown in the next pages, these generalizations can be very useful in measuring and comparing the diffusion of these algorithms, we call this technique Continuous Diffusion Analysis. To this goal, in this section, we propose new metrics for quantifying the avalanche effect and diffusion of cryptographic algorithms. First, we define the signed distance.

DEFINITION 4.13 (Signed Distance) Let $x, y \in \mathcal{B}^n$. Hence, we define the signed distance as

$$d_S(x, y) = \sum_{i=0}^{n-1} |\operatorname{sgn}(x_i) - \operatorname{sgn}(y_i)|,$$

where $\operatorname{sgn}(x)$ is the sign function.

The Avalanche Factor, presented in Definition 4.2, was defined in [1] considering f as a discrete function. However, as we showed in Section 4.3.2, we can create continuous generalizations of these functions. Thus, we propose the *continuous avalanche factor*.

DEFINITION 4.14 (Continuous Avalanche Factor (CAF)) Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $f_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ such that $f_{\mathcal{C}} \overset{\sim}{\leftarrow} f$. Let $x, y \in \mathcal{B}^n$, such that x_i, y_i are independent and identically distributed random variables. Hence, we define the CAF as

$$\begin{aligned} &\mathcal{CAF}(f_{\mathcal{C}}, d_S, d_E|\lambda) \\ &= \frac{\mathbb{E}_{x,y}[d_S(f_{\mathcal{C}}(x), f_{\mathcal{C}}(y))|d_E(x,y) \leqslant \lambda]}{\mathbb{E}_{x,y}[d_S(x,y)]}, \end{aligned}$$

where d_E denote the Euclidean distance and $\lambda \in \mathbb{R}$ is a threshold value.

When x_i and y_i are uniformly distributed, i.e., $x_i, y_i \sim U(-1, 1)$, and $n = m$, one can show that

$$\begin{aligned} &\mathbb{E}_{x,y}[d_S(x,y)] \\ &= \frac{1}{4} \sum_{i=0}^{n-1} \int_{-1}^{1} \int_{-1}^{1} |\operatorname{sgn}(x_i) - \operatorname{sgn}(y_i)| dx_i dy_i = n, \end{aligned}$$

thus, we get

$$\begin{aligned} &\mathcal{CAF}(f_{\mathcal{C}}, d_S, d_E|\lambda) \\ &= \frac{1}{n} \mathbb{E}_{x,y}[d_S(f_{\mathcal{C}}(x), f_{\mathcal{C}}(y))|d_E(x,y) \leqslant \lambda]. \end{aligned}$$

We define two additional metrics, but before that we present two more functions. An important observation is that we have created the proposed continuous generalizations by performing probability operations under the assumption of independence. However, in practice, a cryptographic algorithm mixes all input bits. Therefore, this assumption will not hold after a certain point. Because of that, the result we get by the end is not a probability, and the continuous bits converge to zero exponentially as we progress through the continuous algorithm. To deal with these small numbers, we define the magnitude of a real number by the function

$$Mag(x) = \log_2(|\log_{10}(|x|)| + 1). \tag{4.2}$$

Additionally, we define the function

$$\psi(x, \beta, j) = \begin{cases} \phi(x_i), & i \neq j \\ \beta\phi(x_i), & i = j. \end{cases} \tag{4.3}$$

The continuous generalizations can also be useful to measure the influence of a certain bit to the output of a function. To do that, we propose the following metric.

Definition 4.15 (Continuous Neutrality Measure (CNM)) Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $f_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ such that $f_{\mathcal{C}} \overset{\sim}{\leftarrow} f$, and define $\mathcal{I} \subset \{0, 1, 2, ..., m-1\}$ as a subset of output bits indexes and $\beta \sim U(0, 1)$, a uniform random variable. We measure the influence of an input bit j to an output set of bits $\mathcal{I}$ by the Continuous Neutrality Measure given by

$$\Psi(j, \mathcal{I}, f|f_{\mathcal{C}})$$

$$= \frac{1}{|\mathcal{I}|2^n}\mathbb{E}_\beta\left[\sum_{x \in \mathbb{F}_2^n}\sum_{i \in \mathcal{I}} Mag(f_{\mathcal{C}_i}(\psi(x, \beta, j)))\right],$$

where $f_{\mathcal{C}_i}(.)$ denotes the i-th element of $f_{\mathcal{C}_i}(.)$.

From the CNM, we can compute the average influence of every input bit, which will result in a new metric capable of measuring the overall diffusion of an algorithm, we call this the Diffusion Factor of f.

Definition 4.16 (Diffusion Factor (DF)) Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $f_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ such that $f_{\mathcal{C}} \overset{\sim}{\leftarrow} f$ and $\mathcal{I} = \{0, 1, 2, ..., m-1\}$, the set of all output bit indexes, then we define the

Diffusion Factor of f given the continuous generalization $f_{\mathcal{C}}$ as

$$\mathcal{DF}(f|f_{\mathcal{C}}) = \frac{1}{n}\sum_{j=0}^{n-1}\Psi(j,\mathcal{I},f|f_{\mathcal{C}}).$$

4.4 CASE STUDY: DIFFUSION ANALYSIS OF SALSA, CHACHA, AES, AND SPECK

In this section, we analyze the algorithms Salsa, ChaCha, AES, and Speck with CDA. For all these algorithms, we used the version with 256-bit keys. We divided this section in two parts: first, we show how to use the proposed techniques to compare similar algorithms and how to use it to study small changes in the design. These ideas can be useful to design new algorithms in the future. To this goal, we focused only on the algorithms Salsa and ChaCha because they are very similar ciphers, and we expect to get better diffusion for ChaCha. Second, we apply the proposed metrics to compare different algorithms, including the algorithms Speck, and AES.

When comparing Salsa and ChaCha, from a specification perspective, there are two major changes when comparing Salsa to ChaCha. First, the change of the QRF from QR_{salsa} to QR_{chacha}. Second, the pattern of operation: in Salsa, the function QR_{salsa} is applied first on each column and then on each line of the state matrix, using the element on the diagonal as the first input to the function. Instead, in ChaCha, the function QR_{chacha} is applied first on each column and then on each diagonal but using the element of the first line as the first input to the function. To test whether these changes individually increase the diffusion, we define $Salsa_{\mathcal{C}} \overset{\sim}{\leftarrow} Salsa$ and $ChaCha_{\mathcal{C}} \overset{\sim}{\leftarrow} ChaCha$, as the continuous generalizations of Salsa and ChaCha, respectively, implemented using Definitions 4.4, 4.9 and 4.10. Additionally, we define two modified algorithms:

1. $Salsa\_pt$: consists in the Salsa algorithm but using ChaCha's pattern of operations, in other words, applying QR_{salsa} first on each column and then on each diagonal. We also define $Salsa\_pt_{\mathcal{C}}$ as its continuous generalization.

2. $Salsa\_qr$: consists in the Salsa algorithm but using QR_{chacha} instead of QR_{salsa}. We also define $Salsa\_qr_{\mathcal{C}}$ as its continuous generalization.

4.4.1 Exploratory and Graphical Analysis of ChaCha and Salsa

As usual, we represent Salsa's and ChaCha's states as 4×4 matrices, as in Eq. (2.3). However, as we analyze for every bit, we also represent a 32-bit integer as a 4×8 matrix, leading to the following representation

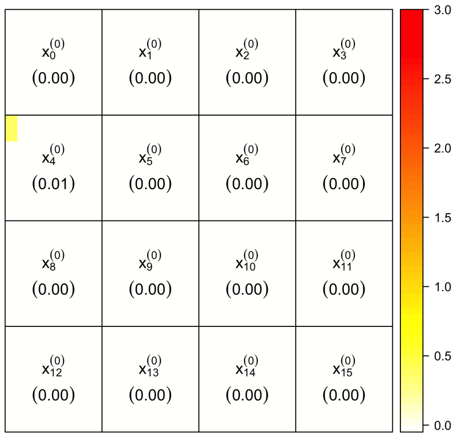
$$X^{(r)} = \begin{pmatrix} x_0^{(r)} & x_1^{(r)} & x_2^{(r)} & x_3^{(r)} \\ x_4^{(r)} & x_5^{(r)} & x_6^{(r)} & x_7^{(r)} \\ x_8^{(r)} & x_9^{(r)} & x_{10}^{(r)} & x_{11}^{(r)} \\ x_{12}^{(r)} & x_{13}^{(r)} & x_{14}^{(r)} & x_{15}^{(r)} \end{pmatrix}, \tag{4.4}$$

where

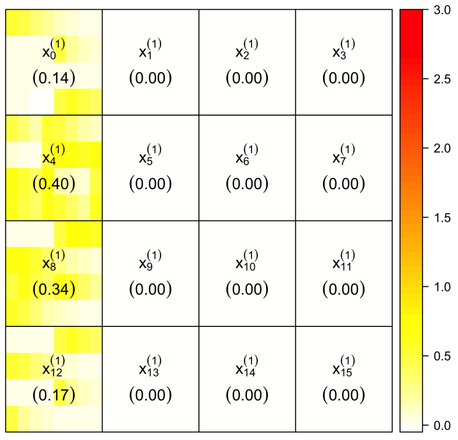
$$x_i^{(r)} = \begin{pmatrix} x_{i,0}^{(r)} & x_{i,1}^{(r)} & \dots & x_{i,7}^{(r)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i,24}^{(r)} & x_{i,25}^{(r)} & \dots & x_{i,31}^{(r)} \end{pmatrix}.$$

We start our analysis with a simple example, using the CNM of Definition 4.15 we measured the influence of the input bit $j = 129$ to each output bit, individually, i.e., we computed $\Psi(129, 0, ChaCha|ChaCha_{\mathcal{C}}), \dots, \Psi(129, 511, ChaCha|ChaCha_{\mathcal{C}})$. We used $j = 129$ because this leads to $x_{4,0}^{(0)}$, which is the first bit of the key, but any other bit might be used. The results are presented in Figure 4.1. As expected, $x_{4,0}^{(0)}$ does not have any influence on columns $2, 3$, and 4 after one round, since in the first round QR_{chacha} is applied on each column independently. Nevertheless, we can also note some bits in the first column that are either very little affected or not affected at all. For the second round, many more bits are affected by $x_{4,0}^{(0)}$, however, there are some regions that are still not very affected. For example, the bits of integer $x_1^{(2)}$, because we have $\Psi(129, \mathcal{I}, ChaCha|ChaCha_{\mathcal{C}}) = 0.39$, where $\mathcal{I} = \{32, 33, ..., 63\}$.

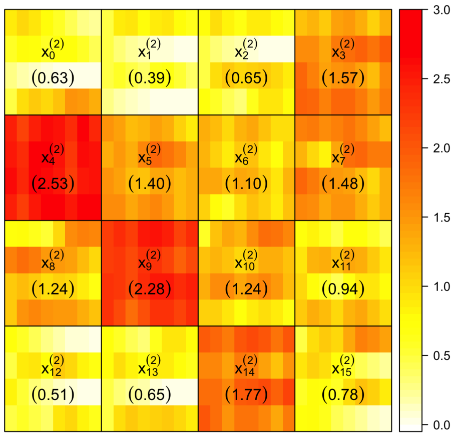
Certainly, we could repeat this experiment for each input bit to understand the diffusion behavior of the algorithm. To observe an average behavior of the diffusion of the algorithm, we can compute the Diffusion Factor. To get a visual representation, we estimated the DF for each output bit individually, i.e., we computed $\mathcal{DF}(f_i|f_{\mathcal{C}_i})$ where f_i denotes the i-th bit of f, Fig. 4.2 presents the results. We used $r = 3$ because we get a better visual representation in this case. These results seem to confirm Bernstein's claims that ChaCha has better diffusion than Salsa. In fact, each change in the algorithm, represented by $Salsa\_pt$ and $Salsa\_qr$, improves the DF individually. Also, it is interesting to note the patterns: Salsa has higher magnitude on the diagonals and ChaCha has higher magnitude on the second line, which are a consequence of the order of the operation.



(a) Initial matrix.

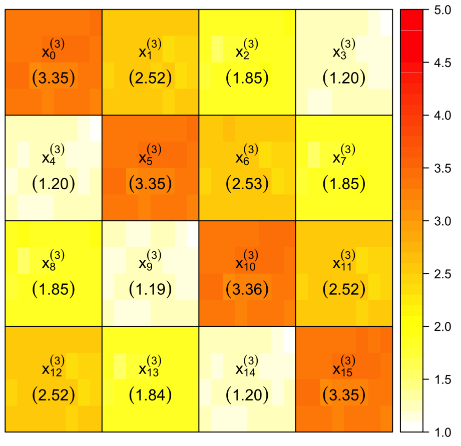


(b) CNM after one round of $ChaCha_{\mathcal{C}}$.

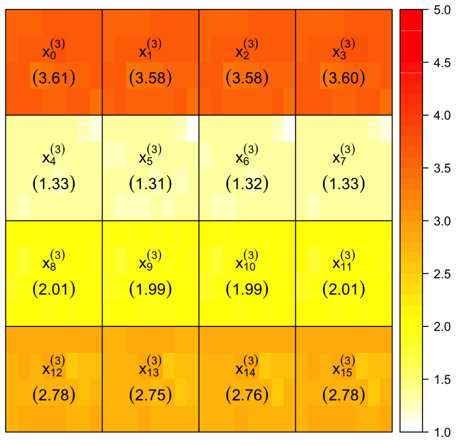


(c) CNM after two rounds of $ChaCha_{\mathcal{C}}$.

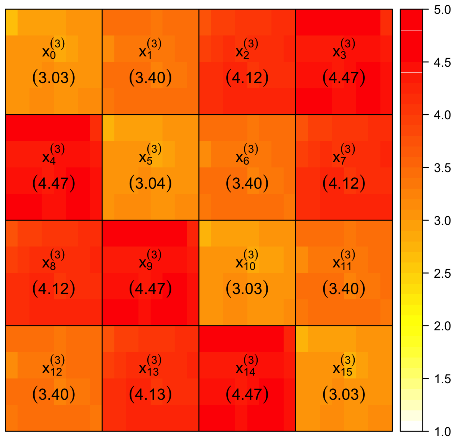
Figure 4.1 – Estimation of the CNM of two rounds of $ChaCha_{\mathcal{C}}$ for input bit $x_{4,0}^{(0)}$. The colors represent the CNM metric of Definition 4.15 for each output bit individually, organized as in Eq. (4.4). Between parenthesis, we have the CNM metric for each integer, i.e., defining $\mathcal{I}$ as a set of 32 indexes.



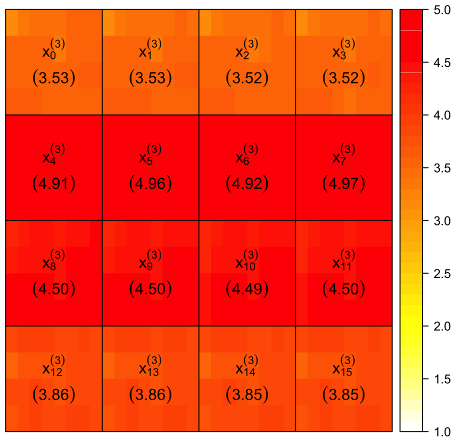
(a) DF for each bit for $Salsa$.



(b) DF for each bit for $Salsa\_pt$.



(c) DF for each bit for $Salsa\_qr$.



(d) DF for each bit for $ChaCha$.

Figure 4.2 – Estimated Diffusion Factor obtained for each bit after 3 rounds of $Salsa$, $ChaCha$, $Salsa\_pt$ and $Salsa\_qr$. The colors represent the DF of each bit, computed from Definition 4.16 and organized as in Eq. (4.4). The average DF of each integer $x_i^{(r)}$ is shown between parentheses.

4.4.2 Continuous Diffusion Analysis of Salsa, ChaCha, AES, and Speck

In this section, we present the CDA of Salsa, ChaCha, AES, and Speck, using our new metrics. To obtain a comparison of the new metrics of Section 4.3.3 with other diffusion metrics, we use the Avalanche Factor proposed by [1], which we describe in Section 4.2. Table 4.1 shows the results of the AF, from Definition 4.2. In addition, Table 4.2 shows the results of the CAF, from Definition 4.14. The new metrics have several advantages. First, notice that the AF provides useful information only for 3 rounds of Salsa or ChaCha, in contrast, the CAF can provide results for several rounds. Moreover, it is possible to balance this behavior by changing the value of the threshold λ. This trend reveals the usefulness of the CAF because decreasing the value of λ, we can go further into the rounds of each algorithm. When comparing these metrics, we also note that the AF cannot distinguish between all algorithms clearly, for instance, it is not clear whether $ChaCha$ is better than $Salsa\_qr$. Contrarily, the CAF results suggests very clearly that when comparing Salsa and ChaCha, we have

$$Salsa_{\mathcal{C}} < Salsa\_pt_{\mathcal{C}} < Salsa\_qr_{\mathcal{C}} < ChaCha_{\mathcal{C}},$$

i.e., ChaCha's diffusion is higher than Salsa's, and the most important change from Salsa to ChaCha is the new QRF, although the pattern of operation also improve diffusion.

The DF further confirms this hypothesis. Table 4.3 shows the DF for each round of both Salsa and ChaCha, again ChaCha has better performance under this metric. It is interesting to note that the DF for 6 rounds of ChaCha is approximately the DF for 7 rounds of Salsa. Also, the DF for 7 rounds of ChaCha is approximately the DF for 8 rounds of Salsa. This suggests that ChaCha20/6 and ChaCha20/7 have approximately the same security of Salsa20/7, and Salsa20/8, respectively. This hypothesis is actually confirmed by the best attacks from the literature that show a security of order $2^{102.2}$, $2^{231.9}$, 2^{137}, and $2^{244.9}$ for ChaCha20/6, ChaCha20/7, Salsa20/7, and Salsa20/8, respectively (see [24] and [25] for these attack). Therefore, we could extrapolate and say that, based on Table 4.3, ChaCha20/17 is as secure as Salsa20.

When comparing the diffusion factor for different algorithms, we should not compare the algorithms directly by rounds, because the number of operations in each round is usually different for each algorithm. Therefore, we should not try to find only the highest diffusion, but also define which algorithm has the *faster diffusion*. To that, we could divide the diffusion factor by the performance in cycles per encrypted byte, obtaining a metric that can be interpreted as how much diffusion we get for each byte per computer cycle. To make this comparison, we used data from the SUPERCOP benchmark [51] (<https://bench.cr.yp.to/supercop.html>). In the SUPERCOP page, we can find the performance of ChaCha, Salsa, Speck, and AES for several different processors. As an example, we used the

Rounds	$Salsa$	$Salsa\_pt$	$Salsa\_qr$	$ChaCha$	Speck	AES
1	0.064	0.063	0.101	0.102	0.066	0.063
2	0.436	0.441	0.725	0.714	0.182	0.250
3	0.919	0.955	0.999	0.998	0.326	≈ 1
4	≈ 1	≈ 1	≈ 1	≈ 1	0.547	≈ 1
5	≈ 1	≈ 1	≈ 1	≈ 1	0.686	≈ 1
6	≈ 1	≈ 1	≈ 1	≈ 1	0.888	≈ 1
7	≈ 1	≈ 1	≈ 1	≈ 1	0.973	≈ 1
8	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1

Table 4.1 – The Avalanche Factor, proposed by [1], for $Salsa$, $Salsa\_pt$, $Salsa\_qr$, $ChaCha$, Speck, and AES. A completely random result would produce an AF of 1.

data available for Intel Core i7-8809G. The results are presented in Table 4.4, where we can see that ChaCha seems to be the best option when balancing for diffusion and performance, for this particular Intel i7 processor.

Rounds	$\mathcal{CAF}$ $(\lambda = 10^{-3})$						$\mathcal{CAF}$ $(\lambda = 10^{-5})$					
	$Sal._{\mathcal{C}}$	$Sal.\_pt_{\mathcal{C}}$	$Sal.\_qr_{\mathcal{C}}$	$ChaCha_{\mathcal{C}}$	Speck	AES	$Sal._{\mathcal{C}}$	$Sal.\_pt_{\mathcal{C}}$	$Sal.\_qr_{\mathcal{C}}$	$ChaCha_{\mathcal{C}}$	Speck	AES
1	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
2	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
3	≈ 0	≈ 0	0.0012	0.0070	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
4	0.132	0.193	0.327	0.531	≈ 0	0.005	≈ 0	≈ 0	≈ 0	0.002	≈ 0	≈ 0
5	0.759	0.825	0.937	0.985	≈ 0	0.734	0.080	0.111	0.190	0.446	≈ 0	≈ 0
6	0.991	0.995	≈ 1	≈ 1	≈ 0	0.981	0.665	0.76	0.895	0.976	≈ 0	0.479
7	≈ 1	≈ 1	≈ 1	≈ 1	≈ 0	≈ 1	0.989	0.991	0.997	≈ 1	≈ 0	0.959
8	≈ 1	≈ 1	≈ 1	≈ 1	0.018	≈ 1	0.998	0.997	≈ 1	≈ 1	≈ 0	≈ 1
9	≈ 1	≈ 1	≈ 1	≈ 1	0.103	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 0	≈ 1
10	≈ 1	≈ 1	≈ 1	≈ 1	0.384	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 0	≈ 1
11	≈ 1	≈ 1	≈ 1	≈ 1	0.722	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	0.006	≈ 1
12	≈ 1	≈ 1	≈ 1	≈ 1	0.923	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	0.085	≈ 1
13	≈ 1	≈ 1	≈ 1	≈ 1	0.979	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	0.348	≈ 1
14	≈ 1	≈ 1	≈ 1	≈ 1	0.995	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	0.697	≈ 1
15	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	0.915	≈ 1
16	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	0.984	≈ 1
17	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1

Table 4.2 – The CAF (with $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$) for $Salsa_{\mathcal{C}}$, $Salsa\_pt_{\mathcal{C}}$, $Salsa\_qr_{\mathcal{C}}$, $ChaCha_{\mathcal{C}}$, Speck, and AES. A completely random result would produce a CAF of 1.

Rounds	$Salsa$	$Salsa\_pt$	$Salsa\_qr$	$ChaCha$	Speck	AES
1	0.02	0.02	0.04	0.04	0.02	0.01
2	0.37	0.40	0.82	0.88	0.06	0.19
3	2.21	2.43	3.75	4.20	0.20	1.46
4	5.54	5.86	7.39	8.08	0.54	3.76
5	9.00	9.43	11.31	12.24	1.19	6.44
6	12.56	12.97	14.94	15.98	2.23	9.33
7	15.77	16.26	18.45	19.72	3.60	12.57
8	19.00	19.50	21.94	23.42	5.15	15.39
9	22.17	22.75	25.51	27.17	6.73	18.19
10	25.37	25.98	28.99	30.89	8.26	20.93
11	28.54	29.23	32.53	34.60	9.88	23.74
12	31.74	32.46	36.07	38.31	11.50	26.63
13	34.90	35.72	39.56	42.02	12.96	29.45
14	38.10	38.99	43.09	45.75	14.37	32.22
15	41.27	42.24	46.59	49.46	15.77	-
16	44.46	45.45	50.11	53.19	17.17	-
17	47.65	48.68	53.63	56.89	18.58	-
18	50.84	51.92	57.14	60.62	19.99	-
19	54.05	55.20	60.66	64.34	21.37	-
20	57.20	58.45	64.20	68.09	22.74	-
21	-	-	-	-	24.15	-
22	-	-	-	-	25.54	-
23	-	-	-	-	26.93	-
24	-	-	-	-	28.30	-
25	-	-	-	-	29.70	-
26	-	-	-	-	31.09	-
27	-	-	-	-	32.48	-
28	-	-	-	-	33.86	-
29	-	-	-	-	35.26	-
30	-	-	-	-	36.65	-
31	-	-	-	-	38.04	-
32	-	-	-	-	39.43	-
33	-	-	-	-	40.81	-
34	-	-	-	-	42.21	-

Table 4.3 – Diffusion Factor for each round of Salsa, ChaCha, Speck, and AES.

	Salsa	ChaCha	Speck	AES
Performance	1.33	1.16	1.30	0.94
DF	57.20	68.09	42.21	32.22
DF/Performance	43.01	58.70	32.47	34.27

Table 4.4 – Performance in cycles/byte for ChaCha, Salsa, AES, and Speck with 256-bit keys for Intel Core i7-8809G processor, for long messages and the Diffusion Factor for all algorithms.

4.5 LIBRARY

We developed an open source library , called `libfpco` that implements continuous generalizations (available at <https://github.com/MurCoutinho/pda>). See Appendix A.1 for more details.

5 *COLORED*: A NEW FRAMEWORK TO EVALUATE SECURITY AGAINST DIFFERENTIAL CRYPTANALYSIS

5.1 INTRODUCTION

Differential cryptanalysis (DC) is one of the most powerful attacks available in the realm of cryptanalysis. It was Invented by Biham and Shamir [32] as a method to attack the block cipher Data Encryption Standard (DES). Later, in 1994, Don Coppersmith, a member of the team that designed DES, published a paper stating that differential cryptanalysis was known to IBM as early as 1974, and that defending against differential cryptanalysis had been a design goal of DES [52]. As the time passed, it became clear that differential cryptanalysis was extremely important, being applied to attack and design many different cryptographic primitives such as block ciphers [30], stream ciphers [24], and hash functions [53].

Since its invention of differential cryptanalysis, tweaks and generalizations have been proposed. In 1994, Lai introduced the concept of high order differential [54] and Knudsen proposed new applications to this technique [55]. Still in [55], Knudsen proposed so-called truncated differentials attacks, which can be used to analyze and make predictions of a subset of bits instead of the full block of a cipher. Another variant of differential cryptanalysis is called impossible differential cryptanalysis, which was proposed independently by Knudsen [56] and Biham [57] and uses differentials with probability zero. In 1999, Wagner introduced the boomerang attack [58], which allows to concatenate any two, not necessarily coinciding, differentials over parts of a cipher. Wagner used the boomerang attack to break the cipher COCONUT98. Later, the boomerang attack itself has been generalized to amplified boomerang attack [59], the rectangle attack [60], and the retracing boomerang attack [61]. Additionally, differential cryptanalysis was combined with linear cryptanalysis to produce so-called differential-linear attacks [39, 40, 27].

Great progress has been made in designing and analyzing block ciphers, especially with the introduction of the AES, but also more recently with many block ciphers appearing in the area of lightweight cryptography. However, there is still research on fundamental aspects of these ciphers going on and important questions are still not understood. For example, we are not able to prove the security of a block cipher in a general way, but only test its security against known attacks.

To evaluate the security of a block cipher against differential cryptanalysis, there are usually two approaches. One is to calculate the minimum number of differentially active S-boxes to obtain an upper bound of the maximum probability of possible differential trails. The other approach is to search for the best differential characteristics to calculate the maximum probability. For some block ciphers, e.g., for block ciphers with large block sizes, this needs a huge workload and is likely to be impossible to be accomplished in a reasonable time.

In particular, it is difficult to compare different algorithms to decide which one is safer because the attacks are effective only when considering simplifications of these algorithms, such as reducing the number of rounds. We also encounter such difficulties when trying to solve other problems, such as measuring the diffusion of cryptographic algorithms. Nevertheless, the previous chapter, we propose a new technique called Continuous Diffusion Analysis (CDA) that can be used to study, design, and compare cryptographic algorithms without the need of reducing the number of rounds. CDA allows us to generalize cryptographic algorithms by transforming the discrete bits into probabilities such that the algorithm is generalized into a continuous mathematical function. One interesting feature of CDA is that it allows to make tiny changes to the input of an algorithm, in the form of continuous perturbations. Thus, it is possible to change "less" than one bit and to consider bits that are in some kind of superposition of the states 0 and 1.

Considering these concepts, in this chapter, we propose *ColoreD*, a new framework to study the security against differential cryptanalysis that allows us to consider Continuous Differences (*ColoreD*), instead of just binary (black and white) differences. More precisely, we propose a theoretical attack model in which is possible to compare pairs of ciphertexts generated by a cipher and its continuous generalization in such a way that we can mount a key recovery attack by exploring statistical properties of the differences of resulting ciphertexts. In this chapter, using the concept of continuous generalizations from [62], we propose a technique called Continuous Differential Cryptanalysis (CDC), which allows us to mount attacks using continuous differences, instead of discrete differences. This technique enables us to observe new statistical properties when comparing to standard differential cryptanalysis. Moreover, we propose a theoretical attack applying CDC that can recover the key of full AES and PRESENT.

The rest of this chapter is organized as follows: in Section 5.2, we present a simple motivational example to introduce the reader to the topic of continuous generalizations and continuous differences. In Section 5.3, we present the framework *ColoreD*, including the idea of CDC. In Section 5.4, we study AES and PRESENT using the framework *ColoreD*.

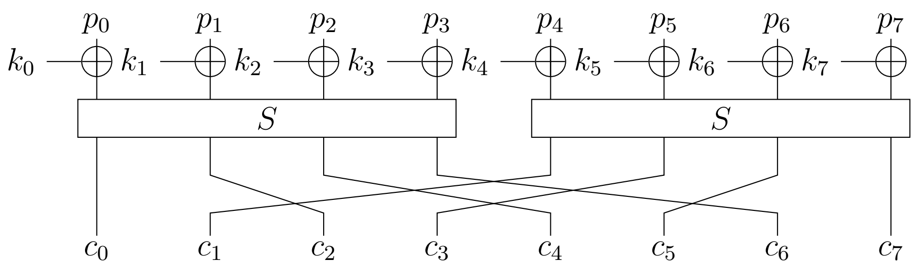


Figure 5.1 – One round of SP, denoted as $C = R(P, K)$. It is a simple SPN, where S is defined in Table 2.2.

5.2 MOTIVATION

In this section, we present a simple example to introduce the notion of studying resistance against differential cryptanalysis through the use of continuous generalizations. To do that, we divide this section in two parts: in Subsection 5.2.1, we propose a toy cipher called Simplified PRESENT, and in Subsection 5.2.2, we present the referred example.

5.2.1 Simplified PRESENT

In this section, we propose a toy cipher, which we call *Simplified PRESENT* (SP). In each round, SP updates an 8-bit block by applying a simple SPN that uses PRESENT's S-box and the following permutation:

i	0	1	2	3	4	5	6	7
$P(i)$	0	2	4	6	1	3	5	7

SP also uses PRESENT's key schedule; however, it only uses the first 8 bits of each round key. Figure 5.1 summarizes this description and presents the permutation layer. To simplify the notation, we define one round of SP as the function $R : \mathbb{F}^8 \times \mathbb{F}^8 \to \mathbb{F}^8$.

Next, using the theory presented in Section 4.3.2, we define a continuous generalization for SP. Notice that SP only uses 3 operations: XOR, bitwise permutations, and substitutions via a single S-Box. Then, we can create a continuous generalization simply by applying Definitions 4.4, 4.11 and 4.12. For the S-Box, we need to write $S(x) = (f_0(x), f_1(x), f_2(x), f_3(x))$, where $x = (x_0, x_1, x_2, x_3) \in \mathcal{B}^4$. To compute $f_i : \mathbb{F}_2^4 \to \mathbb{F}_2$, we wrote the truth table for each bit from the S-Box and then compute the algebraic normal form using SAGE. For completeness, we list the functions explicitly:

$$\begin{aligned}
f_0(x) &= x_0 + x_1x_2 + x_2 + x_3,\\
f_1(x) &= x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_3 + x_1 + x_2x_3 + x_3,\\
f_2(x) &= x_0x_1x_3 + x_0x_1 + x_0x_2x_3 + x_0x_3 + x_1x_3 + x_2 + x_3 + 1,\\
f_3(x) &= x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_0 + x_1x_2 + x_1 + x_3 + 1.
\end{aligned}$$

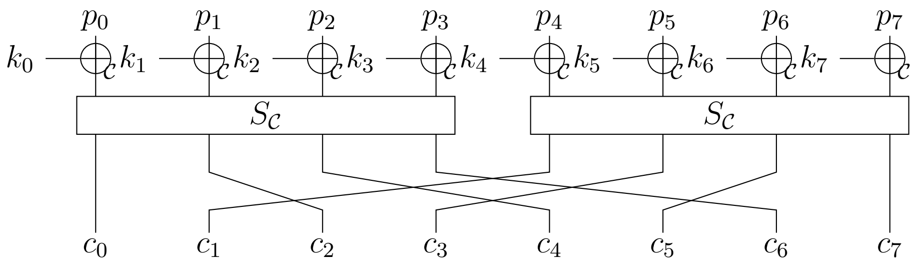


Figure 5.2 – Continuous generalization of the round function R of the cipher SP.

Thus, we can generate a continuous generalization for the round function R, i.e., we define $R_{\mathcal{C}} : \mathcal{B}^8 \times \mathcal{B}^8 \to \mathcal{B}^8$ such that $R_{\mathcal{C}} \overset{\sim}{\leftarrow} R$. We represent $R_{\mathcal{C}}$ in Figure 5.2.

5.2.2 Analyzing differences using a continuous generalization

In differential cryptanalysis, we exploit the high probability of certain occurrences of plaintext differences and differences into the last round of the cipher. For example, consider two inputs to our toy cipher SP be P and P'. Then, after each round r, we get corresponding subciphers C_r and C'_r, respectively. The input difference is given by $\Delta P = P \oplus P'$, and the difference for each round is defined as $\Delta C_r = C_r \oplus C'_r$. In an ideally randomizing cipher, the probability that a particular output difference ΔC occurs given a particular input difference ΔP is 2^{-n} where n is the number of bits of P.

In the majority of cases, differential cryptanalysis is most effective when the input difference affects few bits. In other words, we try to understand and find statistical behaviors of the cipher when changing the input just a little bit. However, modern ciphers are designed so that small changes leads to big changes after a few rounds, behaving as an ideal cipher. For instance, consider the example of Figure 5.3. In this case, we change only one bit of the input of the proposed toy cipher SP and after 2 rounds many bits are already changed.

Would not it be nice if it was possible to apply a change smaller than one bit? This is exactly the idea behind continuous generalizations proposed in [62]. Therefore, now we consider the example of Figure 5.4. Here, we compare differences between SP and its continuous generalization. Notice that unlike the example of 5.3, the subcipher C'_2 is very similar to the control case.

The advantage of this approach is that we can go further into the cipher, allowing comparisons of the full cipher, instead of reducing the number of rounds. This is possible because we are working with real numbers, and we can use an arbitrarily small difference to the input. For example, Figure 5.5 shows that making the input difference closer to 1 leads to an output difference also closer to 1 to every bit. The next question is if we can study these differences to propose some kind of differential analysis.

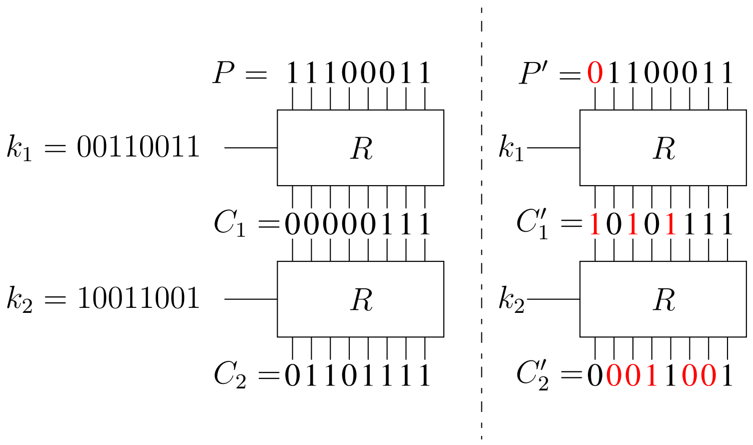


Figure 5.3 – Analyzing differences on SP. Here, two rounds of SP are executed. We consider two different inputs for SP. On the left hand, we have a control case. On the right hand we change the first bit of the input using the same round keys k_1 and k_2. The different bits are represented using the color red.

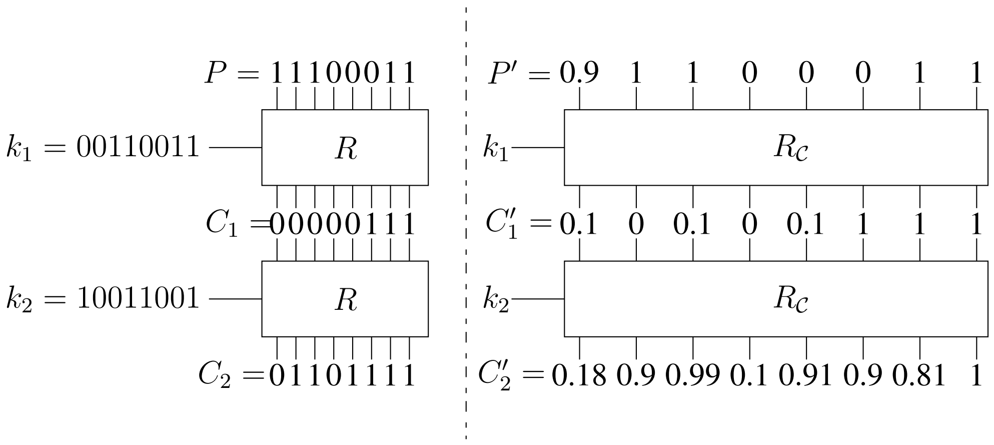


Figure 5.4 – Analyzing differences on SP against its continuous generalization. On the left hand, we have the same control case of two rounds of SP, as in Figure 5.3. However, on the right hand we change the first bit of the input of the continuous generalization from 1 to 0.9 (here we use p instead of ε) and using the same round keys k_1 and k_2. After two rounds, it is possible to note that the subciphers are very close to each other.

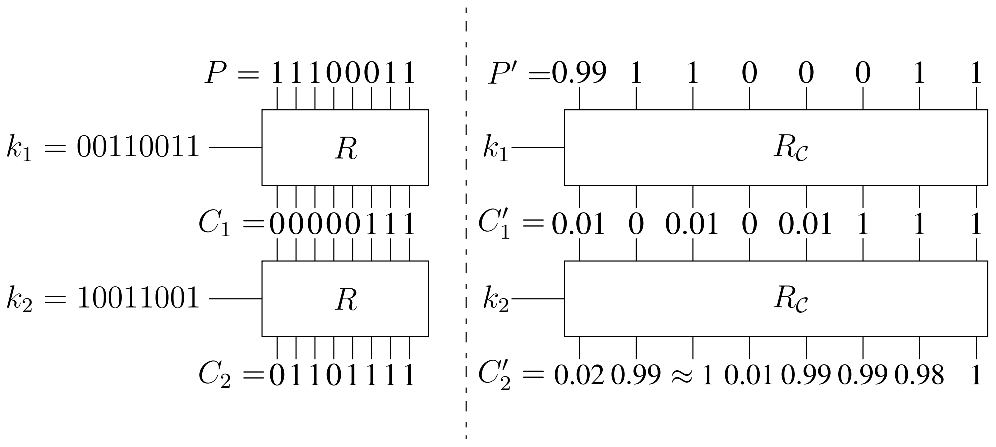

$P = 11100011$
$k_1 = 00110011$
R
$C_1 = 00000111$
$k_2 = 10011001$
R
$C_2 = 01101111$
$P' = 0.99\ 1\ 1\ 0\ 0\ 0\ 1\ 1$
k_1
$R_{\mathcal{C}}$
$C'_1 = 0.01\ 0\ 0.01\ 0\ 0.01\ 1\ 1\ 1$
k_2
$R_{\mathcal{C}}$
$C'_2 = 0.02\ 0.99 \approx 1\ 0.01\ 0.99\ 0.99\ 0.98\ 1$

Figure 5.5 – Analyzing differences on SP against its continuous generalization. Very similar with the example of Figure 5.4; however, we change the first bit of the input of the continuous generalization from 1 to 0.99. After two rounds, it is possible to note that the subciphers are even closer than in Figure 5.4.

5.3 CONTINUOUS DIFFERENCES (*COLORED*) FRAMEWORK

In this section, we propose the framework *ColoreD*. The main idea in *ColoreD* is, as in Section 5.2, to use a continuous generalization to analyze small differences in cryptographic algorithms. More precisely, we propose a theoretical attack model in which is possible to compare pairs of ciphertexts generated by a cipher and its continuous generalization in such a way that we can mount a key recovery attack by exploring statistical properties of the differences of resulting ciphertexts. Thus, we divided this section in two parts: in subsection 5.3.1, we propose the technique of Continuous Differential Cryptanalysis, and in subsection 5.3.2, we propose a new mode of attack called Continuous Chosen-Plaintext Attack.

5.3.1 Continuous Differential Cryptanalysis (CDC)

In this section, we combine the theories of Differential Cryptanalysis (Section 3.1) and Continuous Generalizations of Cryptographic Algorithms (Section 4.3.2). From now on, we denote the differentials of standard differential cryptanalysis as classic differential. Let $x, y \in \mathcal{B}$, then we define the difference between x and y as $\Delta_{\mathcal{C}}(x, y) = x \oplus_{\mathcal{C}} y$, where $\oplus_{\mathcal{C}}$ is given by Definition 4.4. Now let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function, and let $F_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ such that $F_{\mathcal{C}} \overset{\sim}{\leftarrow} F$. Then, define:

DEFINITION 5.1 (Continuous differential) A continuous differential is the pair (β, γ) of an input difference $\beta \in \mathcal{B}^n$ and an output difference $\gamma \in \mathcal{B}^m$, i.e.,

$$F_{\mathcal{C}}(\phi(x)) \oplus_{\mathcal{C}} F_{\mathcal{C}}(\phi(x) \oplus_{\mathcal{C}} \beta) = \gamma,$$

where $x \in \mathbb{F}_2^n$.

DEFINITION 5.2 (Discretized output difference) Let (β, γ) be a continuous differential, then we call $\Delta = \sigma(\gamma)$ the *discretized output difference* of (β, γ).

Note that continuous differentials are a generalization of classic differentials because if β is discrete, then the resulting differential is equal to a classic differential. In other words, classic differentials can be viewed as a subset of the set of continuous differentials. To prove this statement, we propose the following couple of lemmas and a theorem.

LEMMA 5.1 Let $x, y \in \mathcal{B}^n$, such that $x, y \neq 0$. Then, it holds that $\sigma(x \oplus_{\mathcal{C}} y) = \sigma(x) \oplus \sigma(y)$.

proof.

From the definitions, both $\sigma(.)$ and $\oplus_{\mathcal{C}}$, operate on each entry of the vectors x and y independently. Thus, we just need to show that $\sigma(x_i \oplus_{\mathcal{C}} y_i) = \sigma(x_i) \oplus_{\mathcal{C}} \sigma(y_i)$. From the definitions of $\sigma(.)$ and $\oplus_{\mathcal{C}}$, we get

$$\sigma(x_i \oplus_{\mathcal{C}} y_i) = \sigma(-x_i y_i) = \frac{1}{2}(\operatorname{sgn}(-x_i y_i) + 1) = \frac{1}{2}(-\operatorname{sgn}(x_i)\operatorname{sgn}(y_i) + 1)$$

and

$$\sigma(x_i) \oplus \sigma(y_i) = \left[\frac{1}{2}(\operatorname{sgn}(x_i) + 1)\right] \oplus \left[\frac{1}{2}(\operatorname{sgn}(y_i) + 1)\right].$$

Since $x_i, y_i \neq 0$, we only have 4 possibilities $(\operatorname{sgn}(x_i), \operatorname{sgn}(y_i)) = \{(1,1), (1,-1), (-1,1), (-1,-1)\}$, for each case is straightforward to see that we achieve the same result using both equations. ■

LEMMA 5.2 Let $x \in \mathbb{F}_2^n$ and $y \in \mathcal{B}^n$, such that y is discrete, i.e., $y_i = \pm 1, i = (0, 1, ..., n-1)$. Then it holds that $\phi(x) \oplus_{\mathcal{C}} y = \phi(x \oplus \sigma(y))$.

proof.

From the definitions, $\sigma(.)$, $\phi(.)$ and $\oplus_{\mathcal{C}}$ operate on each entry of the vectors x and y independently. Thus, we just need to show that $\phi(x_i) \oplus_{\mathcal{C}} y_i = \phi(x_i \oplus \sigma(y_i))$. From the definitions, we must have

$$\phi(x_i) \oplus_{\mathcal{C}} y_i = -(2x_i - 1)y_i$$

and

$$\phi(x_i \oplus \sigma(y_i)) = 2(x_i \oplus ((\mathrm{sgn}(y_i) + 1)/2)) - 1.$$

Now, we only have 4 possibilities $(x_i, y_i) = \{(1,1), (1,-1), (0,1), (0,-1)\}$, for each case is easy to see that we get the same result using both equations. ■

THEOREM 5.1 Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function, and let $F_{\mathcal{C}} : \mathcal{B}^n \to \mathcal{B}^m$ such that $F_{\mathcal{C}} \overset{\sim}{\leftarrow} F$. Also, let (β, γ) be a continuous differential for $F_{\mathcal{C}}$ and (δ, Δ) be a classic differential for F such that $\delta = \sigma(\beta)$. If β is discrete, then the discrete output difference of (β, γ) is Δ, i.e., the continuous differential is identical to the classic differential.

proof.

The discrete output difference of (β, γ) is given by $\sigma(\gamma)$. Using Definition 5.1, we find that

$$\sigma(\gamma) = \sigma(F_{\mathcal{C}}(\phi(x)) \oplus_{\mathcal{C}} F_{\mathcal{C}}(\phi(x) \oplus_{\mathcal{C}} \beta)).$$

Note that by Definition 4.3, we must have $F(x) = \sigma(F_{\mathcal{C}}(\phi(x)))$, therefore, from the definition of $\sigma(.)$, we must have that $F_{\mathcal{C}}(\phi(x)) \neq 0$, for all $x \in \mathbb{F}_2^n$. Additionally, it is easy to see that if we have all possible $x \in \mathbb{F}_2^n$, then by the definition of ϕ, it follows that $\phi(x)$ generates all possible arrays of size n composed of ± 1 elements, in other words $\phi : \mathbb{F}_2^n \to \{-1, 1\}^n$ can be viewed as a bijection. Thus, since β is discrete, i.e., $\beta_i = \pm 1$ or $\beta \in \{-1, 1\}^n$, then by the definition of $\oplus_{\mathcal{C}}$ is easy to see that $\phi(x) \oplus_{\mathcal{C}} \beta$ defines a permutation on the set $\{-1, 1\}^n$. Hence, as before, from the definition of $\sigma(.)$, we must have that $F_{\mathcal{C}}(\phi(x) \oplus_{\mathcal{C}} \beta) \neq 0$. Now we can use Lemma 5.1 to get

$$\sigma(\gamma) = \sigma(F_{\mathcal{C}}(\phi(x))) \oplus \sigma(F_{\mathcal{C}}(\phi(x) \oplus_{\mathcal{C}} \beta)),$$

and using Lemma 5.2 on the second term of the equation, we get

$$\sigma(\gamma) = \sigma(F_{\mathcal{C}}(\phi(x))) \oplus \sigma(F_{\mathcal{C}}(\phi(x \oplus \sigma(\beta))).$$

Finally, replacing $\sigma(\beta)$ by δ and using Definition 4.3, we get $\sigma(\gamma) = F(x) \oplus F(x \oplus \delta) = \Delta$. ■

One very common property explored in differential cryptanalysis is that when using it to attack ciphers in which the key is applied only via a XOR operation, the key bits can generally be ignored when finding a differential because if we have $w = x \oplus k$ and $w' = x' \oplus k$, then we get $\Delta w = x \oplus k \oplus x' \oplus k = \Delta x$. In the same way, when applying CDC to these ciphers, the key can be ignored, we formalize this concept in the following lemma:

LEMMA 5.3 Let $w, w', x, x' \in \mathcal{B}$ and $k \in \mathbb{F}_2$ such that $w = x \oplus_{\mathcal{C}} \phi(k)$ and $w' = x' \oplus_{\mathcal{C}} \phi(k)$, then we can write $\Delta_{\mathcal{C}} w = \Delta_{\mathcal{C}} x$.

proof.

This result is straightforward since using Definition 4.4, we have

$$\begin{aligned}\Delta_{\mathcal{C}} w &= w \oplus_{\mathcal{C}} w' = -ww' = -(x \oplus_{\mathcal{C}} \phi(k))(x' \oplus_{\mathcal{C}} \phi(k)) \\ &= -(-x\phi(k))(-x'\phi(k)) = -xx'\phi^2(k) = -xx' \\ &= \Delta_{\mathcal{C}} x.\end{aligned}$$

■

In Section 3.1, we explained the concept of difference distribution tables. Unfortunately, since the input difference β is continuous, when working with continuous differentials we do not have such a table because there are an infinite number of differentials. However, for a fixed β, we can compute an array of differentials.

DEFINITION 5.3 (β-difference distribution array) We define the β-difference distribution array as $A^{(\beta)} = \{A_i^{(\beta)}\}$, where

$$A_i^{(\beta)} = |\{x \in \mathbb{F}_2^n : \sigma(F_{\mathcal{C}}(\phi(x)) \oplus_{\mathcal{C}} F_{\mathcal{C}}(\phi(x) \oplus_{\mathcal{C}} \beta)) = i\}|.$$

In standard differential cryptanalysis, if we divide the cipher in multiple components $(c_0, c_1, ..., c_s)$, we can define the *continuous differential characteristic*, a series of differences, notated as an $(s+1)-$tuple $(\beta_0, ..., \beta_s)$, where $\beta_i \in \mathcal{B}^n$. In each step we have $\Delta_{\mathcal{C}} c_i = \phi(c_i) \oplus_{\mathcal{C}} \tilde{c}_i$, where $c_i \in \mathbb{F}_2^n = \mathcal{M}$ and $\tilde{c}_i = \phi(c_i) \oplus_{\mathcal{C}} \beta_i$. We can also compute the probability of occurrence of the characteristic

$$\Pr_{\mathcal{M},\mathcal{K}}\left(\Delta_{\mathcal{C}} c_s = \beta_s, \Delta_{\mathcal{C}} c_{s-1} = \beta_{s-1}, \ldots, \Delta_{\mathcal{C}} c_1 = \beta_1 \mid \Delta_{\mathcal{C}} m = \beta_0\right),$$

where the probability is taken over all choices of the plaintext and the key.

Next, we show that we can derive a generalization of the concept of Markov ciphers.

DEFINITION 5.4 (Continuous Markov Cipher) A continuous generalization of an iterated cipher is called a *Continuous Markov Cipher* (CMC) if $\Pr\left(\Delta_{\mathcal{C}} c_1 = \gamma \mid \Delta_{\mathcal{C}} c_0 = \beta, c_0 = m\right)$ is independent of $m \in \mathbb{F}_2^n = \mathcal{M}$ for all $\gamma, \beta \in \mathcal{B}^n$ when the round key k is chosen uniformly at random from the set $\mathcal{K}$.

THEOREM 5.2 If an r-round continuous generalization of an iterated cipher is a CMC and the r round keys are independent and uniformly random, then the sequence of continuous differences $\Delta_{\mathcal{C}} c_0, \Delta_{\mathcal{C}} c_1, ..., \Delta_{\mathcal{C}} c_r$ is a homogeneous Markov chain.

proof.

First, we show that the sequence $\Delta_{\mathcal{C}} c_0, \Delta_{\mathcal{C}} c_1, ..., \Delta_{\mathcal{C}} c_r$ is a Markov chain. To do that, it is sufficient to show for the second round that

$$\Pr(\Delta_{\mathcal{C}} c_2 = \beta_2 | \Delta_{\mathcal{C}} c_1 = \beta_1, \Delta_{\mathcal{C}} c_0 = \beta_0) = \Pr(\Delta_{\mathcal{C}} c_2 = \beta_2 | \Delta_{\mathcal{C}} c_1 = \beta_1).$$

To show this, we use the law of total probability and the Bayes theorem to write

$$\begin{aligned}
&\Pr(\Delta_{\mathcal{C}} c_2 = \beta_2 | \Delta_{\mathcal{C}} c_1 = \beta_1, \Delta_{\mathcal{C}} c_0 = \beta_0) = \\
&\textstyle\sum_x \Pr(c_1 = x, \Delta_{\mathcal{C}} c_2 = \beta_2 | \Delta_{\mathcal{C}} c_1 = \beta_1, \Delta_{\mathcal{C}} c_0 = \beta_0) = \\
&\textstyle\sum_x \Pr(c_1 = x | \Delta_{\mathcal{C}} c_1 = \beta_1, \Delta_{\mathcal{C}} c_0 = \beta_0) \Pr(\Delta_{\mathcal{C}} c_2 = \beta_2 | c_1 = x, \Delta_{\mathcal{C}} c_1 = \beta_1, \Delta_{\mathcal{C}} c_0 = \beta_0) = \\
&\textstyle\sum_x \Pr(c_1 = x | \Delta_{\mathcal{C}} c_1 = \beta_1, \Delta_{\mathcal{C}} c_0 = \beta_0) \Pr(\Delta_{\mathcal{C}} c_2 = \beta_2 | c_1 = x, \Delta_{\mathcal{C}} c_1 = \beta_1) = \\
&\Pr(\Delta_{\mathcal{C}} c_2 = \beta_2 | \Delta_{\mathcal{C}} c_1 = \beta_1),
\end{aligned}$$

where the third equality comes from the fact that given c_1 and $\Delta_{\mathcal{C}} c_1$, we can compute $\tilde{c}_1$, c_2 and $\tilde{c}_2$, therefore, $\Delta_{\mathcal{C}} c_2$ has no further dependence on $\Delta_{\mathcal{C}} c_0$. Also, the fourth equality follows from Definition 5.4. Finally, since the same round function is used in each round, this Markov chain is homogeneous. ■

Thus, for a CMC with independent round keys, the probability of a s-round continuous characteristic is the product of the probabilities of the s one-round continuous characteristics or, more formally,

$$\begin{aligned}
\Pr\left(\Delta_{\mathcal{C}} c_s = \beta_s, \Delta_{\mathcal{C}} c_{s-1} = \beta_{s-1}, \ldots, \Delta_{\mathcal{C}} c_1 = \beta_1 \mid \Delta_{\mathcal{C}} m = \beta_0\right) = \\
\textstyle\prod_{i=1}^{s} \Pr\left(\Delta_{\mathcal{C}} c_i = \beta_i \mid \Delta_{\mathcal{C}} c_{i-1} = \beta_{i-1}\right).
\end{aligned}$$

The following theorem creates a relationship between Markov Ciphers and CMCs.

THEOREM 5.3 Let E be a r-round iterated cipher such that each round is computed via a round function $R : \mathbb{F}_2^n \times \mathcal{K} \rightarrow \mathbb{F}_2^n$, which receives as input the state at a particular round and an independent and uniformly random round key. Also, let $E_\mathcal{C} \overset{\sim}{\leftarrow} E$ such that each round of the continuous cipher $E_\mathcal{C}$ is computed via a round function $R_\mathcal{C} : \mathbb{B}^n \times \mathcal{K} \rightarrow \mathbb{B}^n$ and $R_\mathcal{C} \overset{\sim}{\leftarrow} R$. If $E_\mathcal{C}$ is a CMC, then E is a Markov cipher.

proof.

Since $E_\mathcal{C}$ is a CMC, it follows that

$$\Pr(\Delta_\mathcal{C} c_1 = \gamma | \Delta_\mathcal{C} c_0 = \beta, c_0) = \Pr(\Delta_\mathcal{C} c_1 = \gamma | \Delta_\mathcal{C} c_0 = \beta).$$

We have that $\tilde{c}_0 = \phi(c_0) \oplus_\mathcal{C} \beta$, and we can write c_1 and $\tilde{c}_1$ in terms of the round function $R_\mathcal{C}$, given the round key k:

$$\Pr(R_\mathcal{C}(\phi(c_0), \phi(k)) \oplus R_\mathcal{C}(\phi(c_0) \oplus_\mathcal{C} \beta, \phi(k)) = \gamma | \Delta_\mathcal{C} c_0 = \beta, c_0) =$$

$$\Pr(R_\mathcal{C}(\phi(c_0), \phi(k)) \oplus_\mathcal{C} R_\mathcal{C}(\phi(c_0) \oplus_\mathcal{C} \beta, \phi(k)) = \gamma | \Delta_\mathcal{C} c_0 = \beta).$$

Clearly, from the previous equation, we conclude that expression $R_\mathcal{C}(\phi(c_0), \phi(k)) \oplus_\mathcal{C} R_\mathcal{C}(\phi(c_0) \oplus_\mathcal{C} \beta, \phi(k))$ does not actually depend on c_0 when k is uniformly random, thus, we can write $F_{\mathcal{K}_\mathcal{C}}(\beta) = R_\mathcal{C}(\phi(c_0), \phi(k)) \oplus_\mathcal{C} R_\mathcal{C}(\phi(c_0) \oplus_\mathcal{C} \beta, \phi(k))$. Now, considering E and R, we can write

$$\Pr(\Delta c_1 = \alpha | \Delta c_0 = \delta, c_0) = \Pr(R(c_0, k) \oplus R(c_0 \oplus \delta, k) = \alpha | \Delta c_0 = \delta, c_0).$$

However, due to Lemma 4.2, we know it must be the case that $F_{\mathcal{K}_\mathcal{C}} \overset{\sim}{\leftarrow} F_\mathcal{K}$. Since $F_{\mathcal{K}_\mathcal{C}}$ does not depend on c_0 and it is a generalization of $F_\mathcal{K}$, then $F_\mathcal{K}$ does not depend on c_0, and we can write

$$\begin{array}{rcl}\Pr(\Delta c_1 = \alpha | \Delta c_0 = \delta, c_0) & = & \Pr(F_\mathcal{K}(\delta) = \alpha | \Delta c_0 = \delta, c_0) = \\ \Pr(F_\mathcal{K}(\delta) = \alpha | \Delta c_0 = \delta) & = & \Pr(\Delta c_1 = \alpha | \Delta c_0 = \delta).\end{array}$$

Thus, E is a Markov cipher. ■

Since we are working with real numbers, we can evaluate the average behavior of the continuous differential given a fixed input difference β. As we will see later in Section 5.4.3, this concept is very useful when we need to define an initial value for β such that we are able to apply CDC.

Definition 5.5 (β-average differential) Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function, and let $F_\mathcal{C} : \mathcal{B}^n \to \mathcal{B}^m$ such that $F_\mathcal{C} \overset{\sim}{\leftarrow} F$. Given a fixed input difference $\beta \in \mathcal{B}^n$, then we define the β-average differential (β-AD) $\theta \in \mathcal{B}$ as the average of the output difference computed from all possible values of $x \in \mathbb{F}_2^n$, i.e.,

$$\theta = \frac{1}{m2^n} \sum_{i=0}^{m-1} \sum_{x \in \mathbb{F}_2^n} F_{i_\mathcal{C}}(\phi(x)) \oplus_\mathcal{C} F_{i_\mathcal{C}}(\phi(x) \oplus_\mathcal{C} \beta). \tag{5.1}$$

5.3.2 Continuous Chosen-Plaintext Attack (CCPA)

In this section, we propose a key-recovery attack using CDC. We consider an attack model similar to a chosen-plaintext attack (CPA) with the difference that the attacker may query a continuous generalization of the algorithm. We formalize this notion in the following definition.

Definition 5.6 (CCPA). Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$, be an encryption algorithm and $E_\mathcal{C} : \mathcal{B}^n \times \mathcal{B}^m \to \mathcal{B}^n$ such that $E_\mathcal{C} \overset{\sim}{\leftarrow} E$. Let $C = E(P, K)$, where $C, P \in \mathbb{F}_2^n$ and $K \in \mathbb{F}_2^m$ define the ciphertext, the plaintext and the key, respectively. In a Continuous Chosen-Plaintext Attack (CCPA) against algorithm E, an attacker is able to make q queries to E and $E_\mathcal{C}$, giving as input plaintexts $P_0, ..., P_{q-1} \in \mathbb{F}_2^n$ and arbitrary values $\beta_0, \beta_1, ..., \beta_{q-1} \in \mathcal{B}^n$, obtaining q pairs of ciphertexts $(C_0, \tilde{C}_0), ..., (C_{q-1}, \tilde{C}_{q-1}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, where $C_i = E(P_i, K)$ and $\tilde{C}_i = \sigma(E_\mathcal{C}(\phi(P_i) \oplus_\mathcal{C} \beta_i, \phi(K)))$. From $P_0, ..., P_{q-1}$ and $(C_0, \tilde{C}_0), ..., (C_{q-1}, \tilde{C}_{q-1})$ the attacker objective is to devise an algorithm $\mathcal{A}$ to recover the key K.

Note from Definition 5.6 that although the continuous generalization is being used to compute the ciphertexts, the attacker does not observe any continuous bits under a CCPA, i.e., like in a CPA the attacker only observes discrete 0 or 1 bits. Additionally, note that CPA is a sub-case of CCPA, we formalize this fact in the following proposition.

Proposition 5.1 If $\beta_i = (-1, -1, ..., -1)$ for all $i = (0, 1, ..., q-1)$, then the CCPA simplifies to a standard CPA.

proof.

When $\beta_i = (-1, -1, ..., -1)$, we have $\phi(P_i) \oplus_{\mathcal{C}} \beta_i = \phi(P_i)$. Therefore, we get

$$\tilde{C}_i = \sigma(E_{\mathcal{C}}(\phi(P_i), \phi(K))),$$

and by Definition 4.3, we have $\tilde{C}_i = E(P_i, K) = C_i$. ■

Thus, our aim is to choose β_i such that we can observe statistical patterns and mount a key recovery algorithm. In the following sections, we show how to choose a useful β experimentally, such that $\beta_i = \beta$ for $i \in [0, 1, ..., q-1]$. Right now, we focus only on the key recovery algorithm presented in Algorithm 3, where $\mathrm{w_h} : \mathbb{F}_2^n \to \mathbb{Z}$ is the hamming weight function. The idea behind this algorithm is that if we choose a small enough β, than the probability that the differences observed in the last round is zero is higher. Therefore, we hope that, when decrypting with an incorrect last round subkey, we will not observe the property mentioned above, and when decrypting with the correct last round subkey k, we will have a low hamming weight for the difference $\delta = \Delta c_{r-1}$. To conclude our discussion, note that in the case of SPNs like AES and PRESENT the decryption of the last round can be divided in independent parts for each S-Box. Therefore, we can use Algorithm 3 considering independent blocks of bits.

Algorithm 3 Key recovery

```
 1: INPUT: ciphertexts pairs $(C_0, \tilde{C}_0), ..., (C_{q-1}, \tilde{C}_{q-1})$, a round function $R : \mathbb{F}_2^n \times \mathcal{K} \to \mathbb{F}_2^n$.
 2: OUTPUT: the subkey for the last round
 3: $k^* = 0$
 4: $M = nq$
 5: for $k \in \mathcal{K}$ do
 6:     $w = 0$
 7:     for $i \in \{0, 1, ..., q-1\}$ do
 8:         $\delta = R^{-1}(C_i, k) \oplus R^{-1}(\tilde{C}_i, k)$
 9:         $w = w + \mathrm{w_h}(\delta)$
10:     if $w < M$ then
11:         $M = w$
12:         $k^* = k$
13: return $k^*$
```

5.4 CASE STUDY: USING *COLORED* TO EVALUATE AES AND PRESENT

In this section, we use the framework *ColoreD*, presented in Section 5.3, to evaluate and to compare AES and PRESENT. To do so, in Section 5.4.1 we apply *ColoreD* to our toy cipher SP to demonstrate the use of the framework. Then, in Section 5.4.2 show how *ColoreD* can be useful when designing new cryptographic primitives. To do that, we propose

3 weaker variations of AES, and we apply the framework to each one of them. Finally, in Section 5.4.3, we compare AES and PRESENT with *ColoreD*, including a key recovery attack using CDC.

5.4.1 Applying *ColoreD* to Simplified PRESENT

Our goal is to apply *ColoreD* to evaluated AES and PRESENT. To make our explanation clearer, we first consider our toy cipher SP, defined in Section 5.2.1. Thus, let $SP : \mathbb{F}_2^8 \times \mathbb{F}_2^8 \to \mathbb{F}_2^8$ be the encryption function for SP, such that $C = SP(P, K)$ where $P, C, K \in \mathbb{F}_2^8$ represent the plaintext, ciphertext, and key, respectively. Also, let $SP_{\mathcal{C}} : \mathcal{B}^8 \times \mathcal{B}^8 \to \mathcal{B}^8$ such that $SP_{\mathcal{C}} \overset{\sim}{\leftarrow} SP$. Then, to apply Algorithm 3 under a CCPA, by Definition 5.6 we need to define plaintexts P_i and parameters β_i for $i \in \{0, 1, ..., q-1\}$ to receive back, ciphertext pairs $(C_0, \tilde{C}_0), ..., (C_{q-1}, \tilde{C}_{q-1})$. For SP, we will consider all possible values for P_i and we define β such that $\beta_i = \beta$ for all i.

We started by applying differential cryptanalysis to SP using Algorithm 3. In this case, given an unknown key $K \in \mathbb{F}_2^8$, we have that $C_i = SP(P_i, K)$ and $\tilde{C}_i = SP(P_i \oplus \Delta, K)$ where $\Delta = (\delta_0, \delta_1, ...\delta_7)$ is a single bit difference, i.e., there is $\mu \in \{0, 1, ..., 7\}$ such that $\delta_j = 1$ when $j = \mu$ and $\delta_j = 0$ otherwise. Equivalently, we can define that $C_i = SP(P_i, K)$ and $\tilde{C}_i = \sigma(SP_{\mathcal{C}}(P_i \oplus_{\mathcal{C}} \beta, \phi(K)))$, where $\beta = (b_0, b_1, ..., b_7)$ such that $b_j = 1$ when $j = \mu$ and $b_j = -1$ otherwise, because, in this case, $\sigma(SP_{\mathcal{C}}(P_i \oplus_{\mathcal{C}} \beta, \phi(K))) = SP(P_i \oplus \Delta, K)$. Experimentally, we verified that this attack was successful against 3 rounds of SP, recovering all bits of 100 randomly generated keys when $\mu \in \{0, 1, 2, 5, 6\}$.

Next, we applied *ColoreD* to extend the numbers of rounds in SP. To do that, we defined a single bit continuous difference $\beta = (b_0, b_1, ..., b_7)$ such that $b_j > -1$ when $j = \mu$ and $b_j = -1$ otherwise. We tested for all possible values of $\mu \in \{0, 1, ..., 7\}$ and for a predefined set

$$\begin{aligned} b_\mu \in \; & \{0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5, 0.45, 0.4, 0.35, 0.3, 0.25, \\ & 0.2, 0.15, 0.1, 0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, \\ & 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.97, 0.99, 0.995, 0.997, 0.999\}. \end{aligned}$$

For a total of 100 randomly generated keys (enough to get an average behavior), we were able to recover the entire key in almost all cases for several combinations of parameters. Table 5.1 summarizes the results.

The most important takeaway of Table 5.1 is that when the number of rounds grows (in other words when the cipher gets stronger and more secure), we need b_μ to be closer to -1 to get a successful key recovery. For these cases, we also computed the β-AD (see Definition 5.5) using 2^{10} keys and all possible plaintexts.

μ	b_μ	rounds	Number of fully recovered keys	Avg error (in bits)	β-AD
0	1	3	100	0.00	-0.015
1	1	3	100	0.00	-0.001
2	1	3	100	0.00	0.086
5	1	3	100	0.00	0.047
6	1	3	100	0.00	0.149
0	0.100	4	100	0.00	-0.091
3	0.100	4	100	0.00	-0.062
1	-0.250	5	100	0.00	-0.057
3	-0.550	6	99	0.04	-0.023
3	-0.750	7	97	0.06	-0.014
1	-0.850	8	99	0.02	-0.011
6	-0.950	9	98	0.08	-0.016
2	-0.970	10	92	0.20	-0.008
0	-0.990	11	96	0.10	-0.011
5	-0.997	12	96	0.12	-0.021
4	-0.999	13	97	0.12	-0.026

Table 5.1 – Best results found using Algorithm 3 against SP. For each set of parameters, we used Algorithm 3 for 100 randomly selected keys. When $b_\mu = 1$, we have classic differences, i.e., CDC reduces to differential cryptanalysis. Conversely, when $b_\mu < 1$, we have continuous differences (*ColoreD*). Note that when the number of rounds grows, we need that b_μ be closer and closer to -1 to get a successful key recovery. Additionally, the β-AD for each case were computed for 2^{10} keys.

Furthermore, note from Table 5.1 that in general the β-AD is between -0.1 and -0.01. In practice, we noticed that this is a very useful result, because it is very hard to test many input differences since generating the inputs and executing Algorithm 3 is computationally intensive. Conversely, estimating the β-AD is cheaper, thus, we can test many input differences until we find one that generates a β-AD between the desired intervals. We call this a *candidate difference*, and to find it, we propose Algorithm 4.

5.4.2 Using *ColoreD* to Design Cryptographic Primitives

In this section, we show how our proposed framework can be useful when designing cryptographic primitives. To do that, we will compare 4 different versions of AES using *ColoreD* to understand which is stronger. First, we define these versions:

- V_a - The S-box of AES is replaced by S-box (a) of Figure 5.6.
- V_b - The S-box of AES is replaced by S-box (b) of Figure 5.6.
- V_c - The S-box of AES is replaced by S-box (c) of Figure 5.6.
- V_d - The original AES.

Algorithm 4 Finding a candidate difference to apply CDC

```
1: INPUT: a position $i$, a function $\beta(\delta, i)$ that computes the $\beta$-AD given $i$ and a difference $\delta$.
2: $maxD = 1, minD = -1, \delta = -0.1$
3: $\theta = \beta(\delta, i)$
4: if $\beta > -0.01$ then
5:     $maxD = \delta$
6:     $\delta = (\delta + minD)/2$
7:     Go to line 3.
8: else if $\beta < -0.1$ then
9:     $minD = \delta$
10:    $\delta = (\delta + maxD)/2$
11:    Go to line 3.
12: else
13:    return $\delta$
```

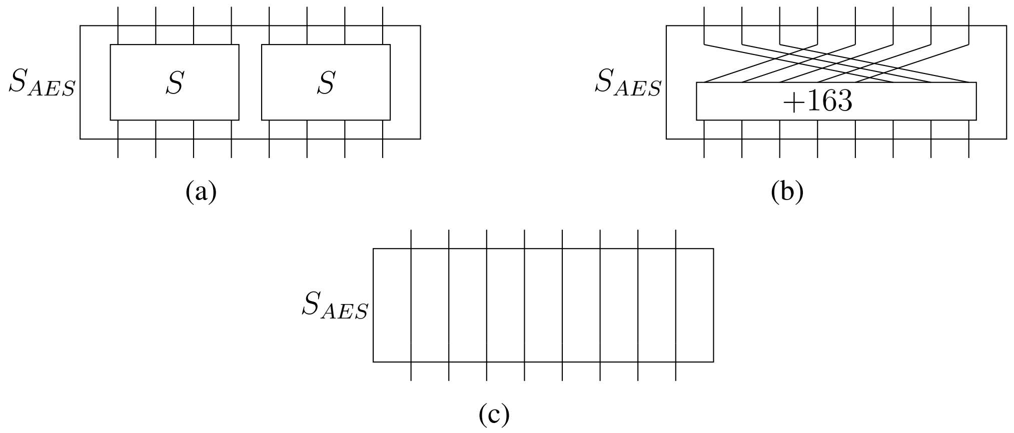


Figure 5.6 – Three alternative (weaker) S-boxes for AES. In the case 5.6a, the 4-bit S-box of PRESENT is used two times in parallel. In the case 5.6b, we have a bit rotation and a constant addition. Case 5.6c is just the identity function.

Clearly, from these four versions, V_c is weaker because it is just the identity function, not providing any diffusion or confusion. Also, V_d is stronger because the S-box of AES has very good properties. In between, we have V_a and V_b

We will compare these ciphers by computing the β-AD and applying Algorithm 4. We considered all possible bit positions $i = (0, 1, 2, ..., 127)$ and different number of rounds. In these cases, the resulting difference δ of Algorithm 4 is very close to -1, therefore, to make the comparison easier to visualize, we use the formula $\log_{10}(\delta + 1)$. The results are presented in Figure 5.7. As expected, Figure 5.7 shows that V_c is the weakest version and V_d is the strongest. Additionally, we can see that V_a is stronger than V_b, thus, using PRESENT's S-box is better than combining a rotation and a constant addition.

Also, in Figure 5.7 is possible to note that removing the S-box of AES (V_c) is worse than reducing the number of rounds to 5 in the original AES (V_d). Similarly, according to this metric, changing the S-box of AES to use the S-box of PRESENT is equivalent to reducing the number of rounds to 8, in terms of security. Clearly, these kinds of comparisons provided by the framework *ColoreD* can be really useful when designing new algorithms, helping the designer to choose new layers or increase the number of rounds of the algorithm.

5.4.3 Evaluating AES and PRESENT

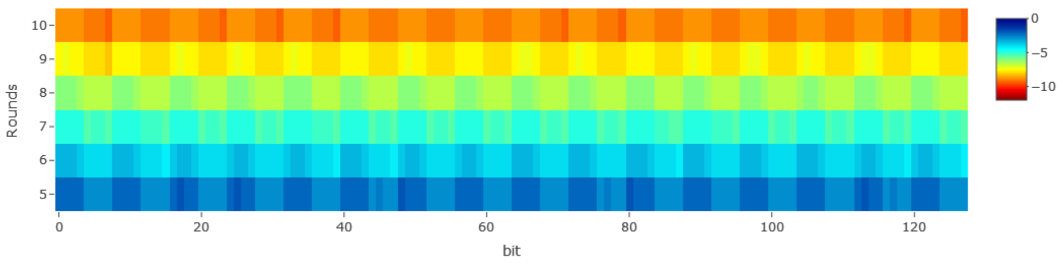
We start by using Algorithm 4 to compute a candidate difference to PRESENT. We considered all possible bit positions $i = (0, 1, 2, ..., 63)$. Additionally, we executed Algorithm 4 when considering PRESENT with reduced and extended number of rounds. More precisely, we considered 10 up to 37 rounds of PRESENT. The results are presented in Figure 5.8. Comparing to Figure 5.7d, these results suggests that PRESENT needs 37 rounds to achieve the same security as AES.

Next, we applied CDC to AES and PRESENT. Notice from Figures 5.7d and 5.8 that the candidate differences do not change significantly when varying the bit position. Therefore, we defined a continuous difference only at the first bit, i.e., we defined $\beta = (b_0, -1, -1, ..., -1)$ such that $b_0 \neq -1$. For PRESENT, using Algorithm 4, we obtained a value of $b_0 \approx -0.9999999997$ that lead to a β-average differential of -0.015. From that, we executed the attack of Algorithm 3 considering $q = 2^5$ as an initial value, and then increasing q until we were able to recover the entire subkey of the last round. We executed this procedure for different values of b_0 and the results are listed in Table 5.2. We should note that a continuous generalization is way slower than the original algorithm. In the case of PRESENT, the continuous generalization is 27 times slower. Therefore, we need to add 4.8 to the time complexity when compared to data complexity.

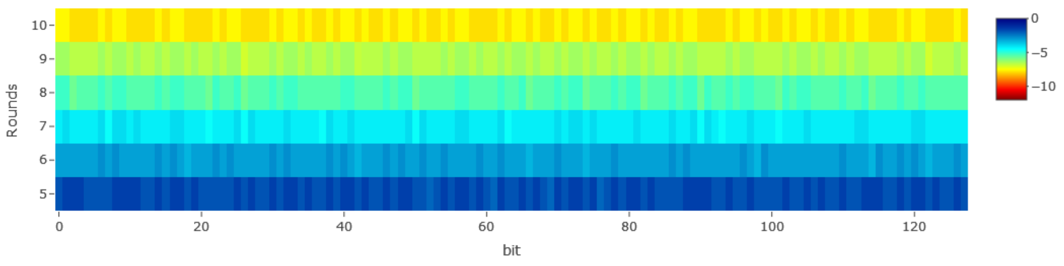
In the case of AES-128, we also defined a continuous difference at the first bit, i.e., we defined $\beta = (b_0, -1, -1, ..., -1)$ such that $b_0 \neq -1$. Using Algorithm 4, we get the value of $b_0 \approx -0.99999999999$, which have β-average differential of -0.0138, then we executed the attack. We present the results for AES-128 in Table 5.3. In the case of AES, the continuous generalization is 5743 times slower. Therefore, we need to add 12.5 to the time complexity when compared to data complexity.

As stated, the proposed attack recovers the last subkey from both AES and PRESENT. Is straightforward to recover the original key in the case of AES because, as presented in Section 2.3.4, the key schedule is invertible. For PRESENT, we are missing 15 bits from the key schedule internal state. Therefore, to find the original key, we just need to perform an exhaustive search, inverting the key schedule to a testable initial key. Using plaintext-ciphertext pairs is easy to find the correct key. Thus, the time complexity of the attack increases by 15 bits in Table 5.2.

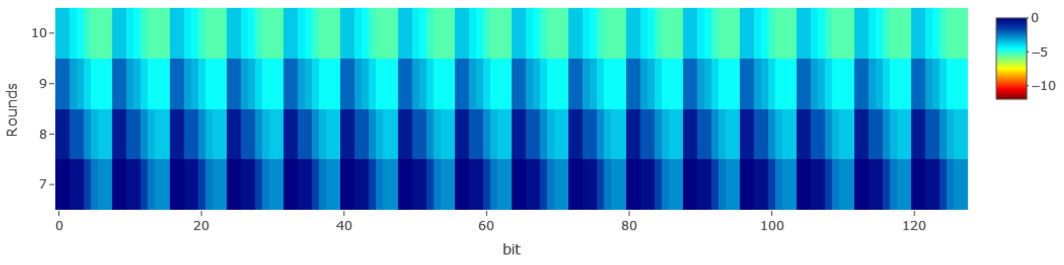
Figure 5.9 compares the attacks against AES and PRESENT. We note that the values of



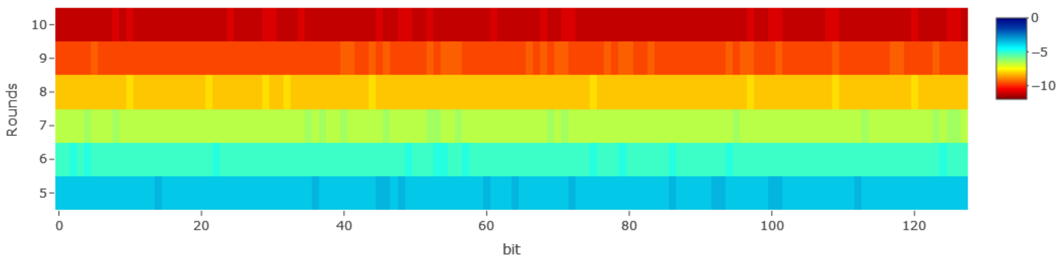
(a) V_a: Using PRESENT S-box.



(b) V_b: Using rotation S-box.



(c) V_c: Using identity S-box.



(d) V_d: AES.

Figure 5.7 – Candidate differences computed using Algorithm 4 with AES and the proposed variations for different bits and number of rounds. To make a good visualization the colors represent the result of the formula $\log_{10}(1 + \delta)$, where δ is the output of Algorithm 4. Therefore, warmer colors indicate a more secure algorithm.

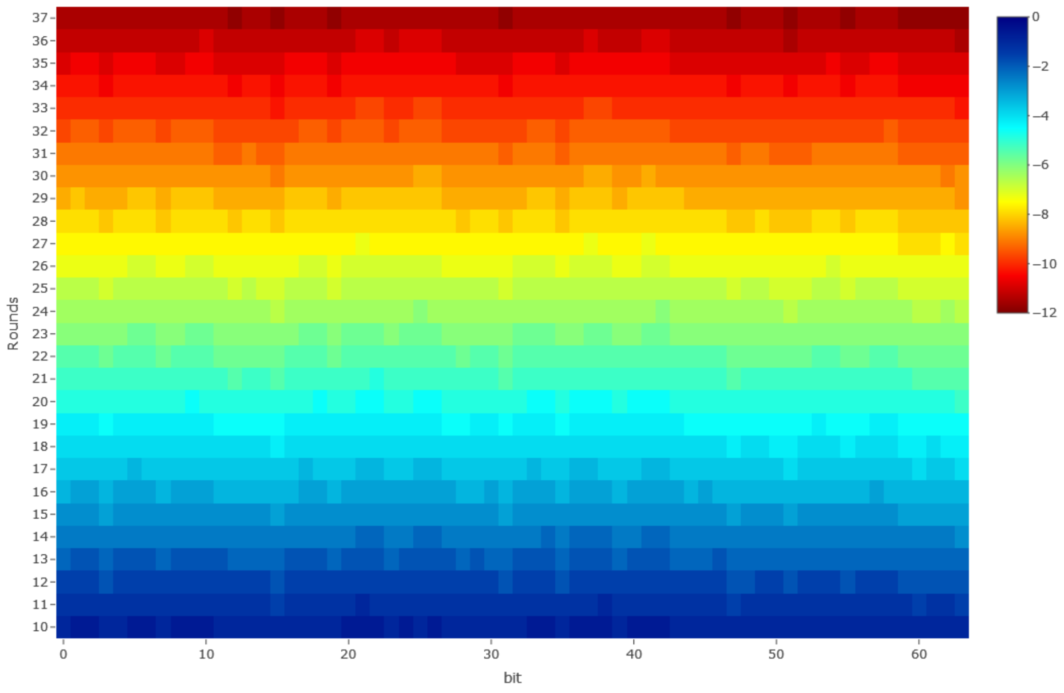


Figure 5.8 – Candidate differences computed using Algorithm 4 with PRESENT. To make a good visualization the colors represent the result of the formula $\log_{10}(1 + \delta)$, where δ is the output of Algorithm 4. Therefore, warmer colors indicate a more secure algorithm.

Time complexity	Data complexity	b_0	β-AD
-	-	-0.9999999999	-0.112000
14.8	10	-0.9999999997	-0.016809
15.8	11	-0.9999999995	-0.005666
17.8	13	-0.9999999993	-0.002699
17.8	13	-0.9999999991	-0.001870
18.8	14	-0.9999999989	-0.000901
18.8	14	-0.9999999987	-0.000892
20.8	16	-0.9999999985	-0.000652
20.8	16	-0.9999999983	-0.000470
21.8	17	-0.9999999981	-0.000306
22.8	18	-0.9999999979	-0.000298
22.8	18	-0.9999999977	-0.000267
-	-	-0.9999999975	-0.000181

Table 5.2 – Data and time complexity (in bits) for a successful key recovery attack against PRESENT for different values of β. With these complexities, the attack correctly finds all the key bits of the last round subkey of PRESENT. The entries with empty complexity mean that the attack did not work even when considering $q = 2^{18}$ ciphertexts.

Time complexity	Data complexity	b_0	β-AD
24.5	12	-0.999999999997	-0.146465
21.5	9	-0.999999999990	-0.015860
21.5	9	-0.999999999983	-0.005361
21.5	9	-0.999999999976	-0.002031
22.5	10	-0.999999999969	-0.000854
22.5	10	-0.999999999962	-0.000532
22.5	10	-0.999999999955	-0.000266
22.5	10	-0.999999999948	-0.000136
22.5	10	-0.999999999941	-0.000096
24.5	12	-0.999999999934	-0.000063
24.5	12	-0.999999999927	-0.000057
23.5	11	-0.999999999920	-0.000029
24.5	12	-0.999999999913	-0.000022
24.5	12	-0.999999999906	-0.000025
25.5	13	-0.999999999899	-0.000012
25.5	13	-0.999999999892	-0.000006
-	-	-0.999999999885	-0.000002

Table 5.3 – Data and time complexity (in bits) for a successful key recovery attack against AES-128 for different values of β. With these complexities, the attack correctly finds all the key bits of the last round subkey of AES-128. The entries with empty complexity mean that the attack did not work even when considering $q = 2^{13}$ ciphertexts.

b_0 are closer to -1 for AES for attacks with the same data complexity. We could interpret this fact by saying that to attack AES with CDC, we need smaller differences to make the attack work; therefore, we can say that AES is stronger than PRESENT.

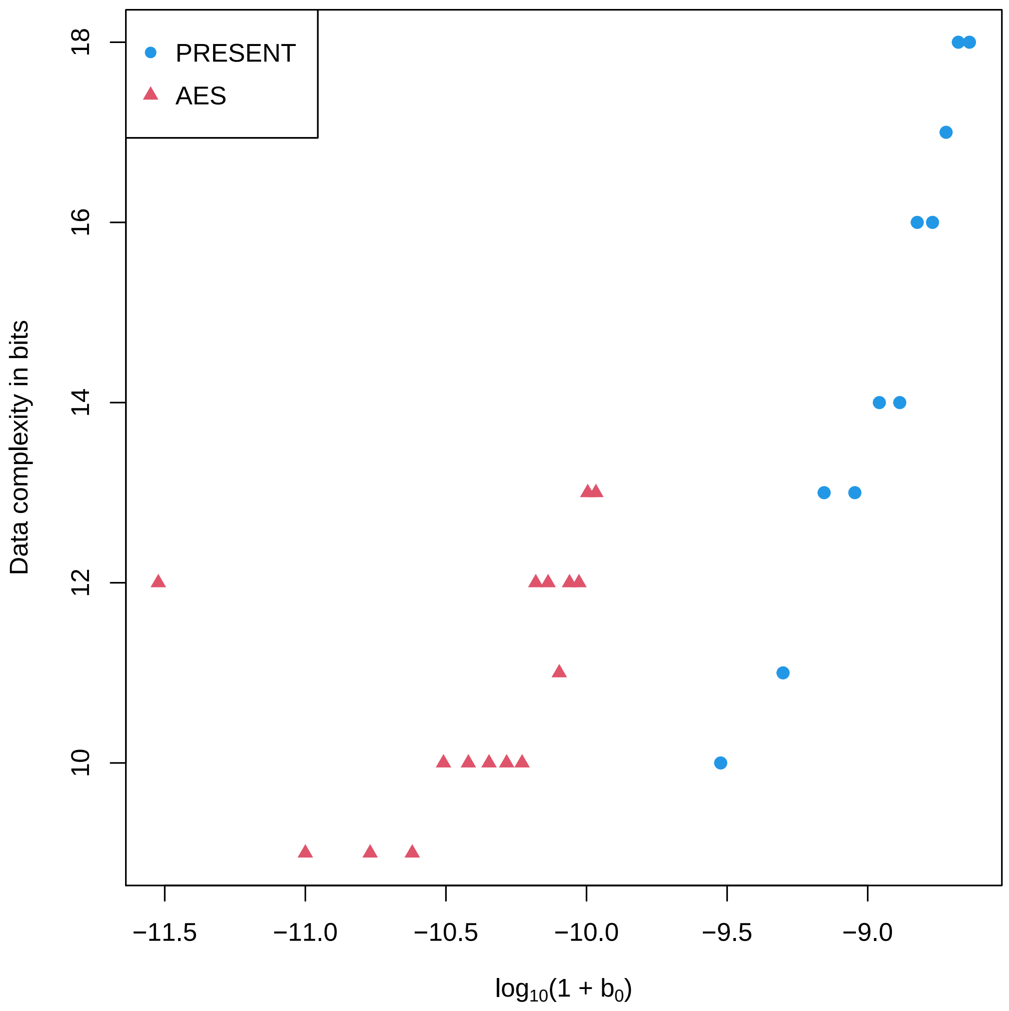


Figure 5.9 – Data complexity for a successful key recovery attack against AES and PRESENT b_0.

Part III

Cryptanalysis of ARX Algorithms

6 IMPROVED DIFFERENTIAL-LINEAR CRYPTANALYSIS OF CHACHA

6.1 INTRODUCTION

Symmetric cryptographic primitives are heavily used in a variety of contexts. In particular, ARX-based design is a major building block of modern ciphers due to its efficiency in software. ARX stands for addition, word-wise rotation, and XOR. Indeed, ciphers following this framework are composed of those operations and avoid the computation of smaller S-boxes through look-up tables. The ARX-based design approach is used to design stream ciphers (e.g., Salsa20 [8] and ChaCha [10]), efficient block ciphers (e.g., Sparx [46]), cryptographic permutations (e.g., Sparkle [63]) and hash functions (e.g., Blake [47]).

ARX-based designs are not only efficient but provide good security properties. The algebraic degree of ARX ciphers is usually high after only a very few rounds as the carry bit within one modular addition already reaches almost maximal degree. For differential and linear attacks, ARX-based designs show weaknesses for a small number of rounds. However, after some rounds the differential and linear probabilities decrease rapidly. Thus, the probabilities of differentials and the absolute correlations of linear approximations decrease very quickly as we increase the number of rounds. In fact, this property led to the long-trail strategy for designing ARX-based ciphers [46].

Ciphers and primitives based on Salsa20 and ChaCha families are heavily used in practice. In 2005, Bernstein proposed the stream cipher Salsa20 [8] as a contender to the eSTREAM [9], the ECRYPT Stream Cipher Project. As outlined by the author, Salsa20 is an ARX type family of algorithms which can be ran with several number of rounds, including the well-known Salsa20/12 and Salsa20/8 versions. Later, in 2008, Bernstein proposed some modifications to Salsa20 to provide better diffusion per round and higher resistance to cryptanalysis. These changes originated a new stream cipher, a variant which he called ChaCha [10]. Although Salsa20 was one of the winners of the eSTREAM competition, ChaCha has received much more attention through the years. Nowadays, we see the usage of this cipher in several projects and applications.

ChaCha, along with Poly1305 [11], is in one of the cipher suits of the new TLS 1.3 [12], which has been used by Google on both Chrome and Android. Not only has ChaCha been used in TLS but also in many other protocols such as SSH, Noise, and S/MIME 4.0. In addition, the RFC 7634 proposes the use of ChaCha in IKE and IPsec. ChaCha has been used not only for encryption, but also as a pseudo-random number generator in any operating system running Linux kernel 4.8 or newer [13, 14]. Additionally, ChaCha has been used in several applications such as WireGuard (VPN) (see [15] for a huge list of applications, protocols and libraries using ChaCha).

Related Work. Since ChaCha is so heavily used, it is very important to understand its security. Indeed, the cryptanalysis of ChaCha is well understood and several authors studied its security [19, 24, 17, 64, 23, 65, 18, 66, 67, 21, 22, 68, 20, 69], which show weaknesses in the reduced round versions of the cipher.

The cryptanalysis of Salsa20 was introduced by Crowley [17] in 2005. Crowley developed a differential attack against Salsa20/5, namely the 5-round version of Salsa20, and received the $1000 prize offered by Bernstein for the most interesting Salsa20 cryptanalysis in that year. In 2006, Fischer et al. [18] improved the attack against Salsa20/5 and presented their attack against Salsa20/6.

One of the most important cryptanalyses in this regard was proposed by Aumasson et al. at FSE 2008 [19] with the introduction of Probabilistic Neutral Bits (PNBs), showing attacks against Salsa20/7, Salsa20/8, ChaCha20/6 and ChaCha20/7. After that, several authors proposed small enhancements on the attack of Aumasson et al. The work by Shi et al. [20] introduced the concept of Column Chaining Distinguisher (CCD) to achieve some incremental advancements over [19] for both Salsa and ChaCha.

Maitra, Paul, and Meier [21] studied an interesting observation regarding reversal round of Salsa, but no significant cryptanalytic improvement could be obtained using this method. Maitra [22] used a technique of Chosen IVs to obtain certain improvements over existing results. Dey and Sarkar [23] showed how to choose values for the PNB to further improve the attack.

In a paper presented in FSE 2017, Choudhuri and Maitra [24] significantly improved the attacks by considering the mathematical structure of both Salsa and ChaCha to find differential characteristics with much higher correlations. Recently, Coutinho and Souza [70] proposed new multi-bit differentials using the mathematical framework of Choudhuri and Maitra. In Crypto 2020, Beierle et al. [26] proposed improvements to the framework of differential-linear cryptanalysis against ARX-based designs and further improved the attacks against ChaCha.

Our Contribution. In this chapter, we provide a new framework to find linear approximations for ARX ciphers. Using this framework, we provide the first explicitly derived linear approximations for 3 and 4 rounds of ChaCha. Exploring these linear approximations, we can improve the attacks for 6 and 7 rounds of ChaCha. Additionally, we present new differentials for 3 and 3.5 rounds of ChaCha. We summarize our findings along with other significant attacks for comparison in Table 6.1. Also, we verified all theoretical results with random experiments. We provide the source code to reproduce this work in Github

Rounds	Time Complexity	Data Complexity	Reference
4	2^{6}	2^{6}	[24]
4.5	2^{12}	2^{12}	[24]
5	2^{16}	2^{16}	[24]
6	2^{139}	2^{30}	[19]
	2^{136}	2^{28}	[20]
	2^{130}	2^{35}	[24]
	$2^{127.5}$	$2^{37.5}$	[24]
	2^{116}	2^{116}	[24]
	$2^{102.2}$	2^{56}	[70]
	$2^{77.4}$	2^{58}	[26]
	2^{75}	2^{75}	[70]
	2^{51}	2^{51}	This work
7	2^{248}	2^{27}	[19]
	$2^{246.5}$	2^{27}	[20]
	$2^{238.9}$	2^{96}	[22]
	$2^{237.7}$	2^{96}	[24]
	$2^{231.9}$	2^{50}	[70]
	$2^{230.86}$	$2^{48.8}$	[26]
	2^{224}	2^{224}	This work

Table 6.1 – The best attacks against ChaCha with 256-bit key.

<https://github.com/MurCoutinho/cryptanalysisChaCha.git>, which is, for the best of our knowledge, the first implementation of cryptanalysis against ChaCha available to the public. We should note that it is possible to find attacks with less complexity for related key attacks or related cipher attacks, but we do not consider them in this work.

6.2 REVIEW OF CRYPTANALYSIS OF CHACHA

In this section, we review the work presented in [24] and in [70]. In these works, the authors developed the theory for selecting specific combination of bits to give high correlations for ChaCha. To do that, in both papers the authors analyzed the QRF directly, representing each equation in its bit level. In the following, we change the original notation of the referred papers to create a notation that will be better for the purposes of this work.

Thus, let $\Theta(x, y) = x \oplus y \oplus (x + y)$ be the carry function of the sum $x + y$. Define $\Theta_i(x, y)$ as the i-th bit of $\Theta(x, y)$. By definition, we have $\Theta_0(x, y) = 0$. We can write the

QRF equations of ChaCha (Eq. 2.6) as

$$\begin{aligned}
x'^{(m-1)}_{a,i} &= x^{(m-1)}_{a,i} \oplus x^{(m-1)}_{b,i} \oplus \Theta_i(x^{(m-1)}_a, x^{(m-1)}_b) \\
x'^{(m-1)}_{d,i+16} &= x^{(m-1)}_{d,i} \oplus x'^{(m-1)}_{a,i} \\
x'^{(m-1)}_{c,i} &= x^{(m-1)}_{c,i} \oplus x'^{(m-1)}_{d,i} \oplus \Theta_i(x^{(m-1)}_c, x'^{(m-1)}_d) \\
x'^{(m-1)}_{b,i+12} &= x^{(m-1)}_{b,i} \oplus x'^{(m-1)}_{c,i} \\
x^{(m)}_{a,i} &= x'^{(m-1)}_{a,i} \oplus x'^{(m-1)}_{b,i} \oplus \Theta_i(x'^{(m-1)}_a, x'^{(m-1)}_b) \\
x^{(m)}_{d,i+8} &= x'^{(m-1)}_{d,i} \oplus x^{(m)}_{a,i} \\
x^{(m)}_{c,i} &= x'^{(m-1)}_{c,i} \oplus x^{(m)}_{d,i} \oplus \Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \\
x^{(m)}_{b,i+7} &= x'^{(m-1)}_{b,i} \oplus x^{(m)}_{c,i}
\end{aligned} \tag{6.1}$$

Inverting these equations, we get

$$x'^{(m-1)}_{b,i} = x^{(m)}_{b,i+7} \oplus x^{(m)}_{c,i} \tag{6.2}$$
$$x'^{(m-1)}_{c,i} = x^{(m)}_{c,i} \oplus x^{(m)}_{d,i} \oplus \Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \tag{6.3}$$
$$x'^{(m-1)}_{d,i} = x^{(m)}_{a,i} \oplus x^{(m)}_{d,i+8} \tag{6.4}$$
$$x'^{(m-1)}_{a,i} = x^{(m)}_{a,i} \oplus x^{(m)}_{b,i+7} \oplus x^{(m)}_{c,i} \oplus \Theta_i(x'^{(m-1)}_a, x'^{(m-1)}_b) \tag{6.5}$$
$$x^{(m-1)}_{b,i} = \mathcal{L}^{(m)}_{b,i} \oplus \Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \tag{6.6}$$
$$x^{(m-1)}_{c,i} = \mathcal{L}^{(m)}_{c,i} \oplus \Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \oplus \Theta_i(x^{(m-1)}_c, x'^{(m-1)}_d) \tag{6.7}$$
$$x^{(m-1)}_{d,i} = \mathcal{L}^{(m)}_{d,i} \oplus \Theta_i(x'^{(m-1)}_a, x'^{(m-1)}_b) \tag{6.8}$$
$$\begin{aligned} x^{(m-1)}_{a,i} = \mathcal{L}^{(m)}_{a,i} &\oplus \Theta_i(x'^{(m-1)}_a, x'^{(m-1)}_b) \oplus \\ &\Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \oplus \Theta_i(x^{(m-1)}_a, x^{(m-1)}_b), \end{aligned} \tag{6.9}$$

where

$$\mathcal{L}^{(m)}_{a,i} = x^{(m)}_{a,i} \oplus x^{(m)}_{b,i+7} \oplus x^{(m)}_{b,i+19} \oplus x^{(m)}_{c,i+12} \oplus x^{(m)}_{d,i} \tag{6.10}$$
$$\mathcal{L}^{(m)}_{b,i} = x^{(m)}_{b,i+19} \oplus x^{(m)}_{c,i} \oplus x^{(m)}_{c,i+12} \oplus x^{(m)}_{d,i} \tag{6.11}$$
$$\mathcal{L}^{(m)}_{c,i} = x^{(m)}_{a,i} \oplus x^{(m)}_{c,i} \oplus x^{(m)}_{d,i} \oplus x^{(m)}_{d,i+8} \tag{6.12}$$
$$\mathcal{L}^{(m)}_{d,i} = x^{(m)}_{a,i} \oplus x^{(m)}_{a,i+16} \oplus x^{(m)}_{b,i+7} \oplus x^{(m)}_{c,i} \oplus x^{(m)}_{d,i+24} \tag{6.13}$$

LEMMA 6.1 It holds that $x^{(m-1)}_{l,0} = \mathcal{L}^{(m)}_{l,0}$, for $l \in \{a, b, c, d\}$.

proof.

This result follows directly from Eqs. (6.6)-(6.9) by using the fact that $\Theta_0(x, y) = 0$. ■

From these equations, we can derive the following lemma:

Lemma 6.2 (Lemma 3 of [24]) Let

$$\begin{aligned}\Delta A^{(m)} &= \Delta x^{(m)}_{\alpha,0} \oplus \Delta x^{(m)}_{\beta,7} \oplus \Delta x^{(m)}_{\beta,19} \oplus \Delta x^{(m)}_{\gamma,12} \oplus \Delta x^{(m)}_{\delta,0}\\ \Delta B^{(m)} &= \Delta x^{(m)}_{\beta,19} \oplus \Delta x^{(m)}_{\gamma,0} \oplus \Delta x^{(m)}_{\gamma,12} \oplus \Delta x^{(m)}_{\delta,0}\\ \Delta C^{(m)} &= \Delta x^{(m)}_{\delta,0} \oplus \Delta x^{(m)}_{\gamma,0} \oplus \Delta x^{(m)}_{\delta,8} \oplus \Delta x^{(m)}_{\alpha,0}\\ \Delta D^{(m)} &= \Delta x^{(m)}_{\delta,24} \oplus \Delta x^{(m)}_{\alpha,16} \oplus \Delta x^{(m)}_{\alpha,0} \oplus \Delta x^{(m)}_{\gamma,0} \oplus \Delta x^{(m)}_{\beta,7}\end{aligned}$$

After m rounds of ChaCha, the following holds:

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

and

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

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

- Case I. m is odd:

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

- Case II. m is even:

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

proof.

See [24]. ■

Lemma 6.3 (Lemma 9 of [24]) For one active input bit in round $m-1$ and multiple active output bits in round m, the following holds for $i > 0$.

$$\begin{aligned}x^{(m-1)}_{b,i} &= \mathcal{L}^{(m)}_{b,i} \oplus x^{(m)}_{d,i-1}, && \text{w.p. } \tfrac{1}{2}\left(1+\tfrac{1}{2}\right)\\ x^{(m-1)}_{a,i} &= \mathcal{L}^{(m)}_{a,i} \oplus x^{(m)}_{b,i+18} \oplus x^{(m)}_{c,i+11} \oplus x^{(m)}_{d,i-2} \oplus x^{(m)}_{d,i+6}, && \text{w.p. } \tfrac{1}{2}\left(1+\tfrac{1}{2^4}\right)\\ x^{(m-1)}_{c,i} &= \mathcal{L}^{(m)}_{c,i} \oplus x^{(m)}_{a,i-1} \oplus x^{(m)}_{d,i+7} \oplus x^{(m)}_{d,i-1}, && \text{w.p. } \tfrac{1}{2}\left(1+\tfrac{1}{2^2}\right)\\ x^{(m-1)}_{d,i} &= \mathcal{L}^{(m)}_{d,i} \oplus x^{(m)}_{c,i-1} \oplus x^{(m)}_{b,i+6}, && \text{w.p. } \tfrac{1}{2}\left(1+\tfrac{1}{2}\right)\end{aligned}$$

proof.

See [24]. ■

Finally, using Lemma 6.2 and Lemma 6.3, it is possible to find linear approximations for two rounds of ChaCha.

LEMMA 6.4 (Lemma 10 of [24]) The following holds with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$

$$\begin{aligned} x_{11,0}^{(3)} &= x_0^{(5)}[0,8,16,24] \oplus x_{1,0}^{(5)} \oplus x_{3,0}^{(5)} \oplus x_{4,7}^{(5)} \oplus x_4^{(5)}[14,15] \oplus x_5^{(5)}[7,19] \oplus \\ & x_8^{(5)}[0,7,8] \oplus x_{9,12}^{(5)} \oplus x_{11,0}^{(5)} \oplus x_{12}^{(5)}[0,24] \oplus x_{13,0}^{(5)} \oplus x_{15}^{(5)}[0,8]. \end{aligned}$$

proof.

See [24]. ■

Recently, Coutinho and Souza [70] found linear approximations with fewer terms using the same techniques.

LEMMA 6.5 (Lemma 5 of [70]) When m is odd, each of the following also holds with probability $\frac{1}{2}(1+\frac{1}{2})$

$$\begin{aligned} x_{0,0}^{(m-2)} \oplus x_{5,0}^{(m-2)} &= x_{0,0}^{(m)} \oplus x_{2,0}^{(m)} \oplus x_{4,7}^{(m)} \oplus x_{4,19}^{(m)} \oplus x_{5,26}^{(m)} \oplus x_{8,12}^{(m)} \oplus x_{9,7}^{(m)} \oplus \\ & x_{9,19}^{(m)} \oplus x_{10,0}^{(m)} \oplus x_{12,0}^{(m)} \oplus x_{13,6}^{(m)} \oplus x_{13,7}^{(m)} \oplus x_{14,0}^{(m)} \oplus x_{14,8}^{(m)} \\ x_{1,0}^{(m-2)} \oplus x_{6,0}^{(m-2)} &= x_{1,0}^{(m)} \oplus x_{3,0}^{(m)} \oplus x_{5,7}^{(m)} \oplus x_{5,19}^{(m)} \oplus x_{6,26}^{(m)} \oplus x_{9,12}^{(m)} \oplus x_{10,7}^{(m)} \oplus \\ & x_{10,19}^{(m)} \oplus x_{11,0}^{(m)} \oplus x_{13,0}^{(m)} \oplus x_{14,6}^{(m)} \oplus x_{14,7}^{(m)} \oplus x_{15,0}^{(m)} \oplus x_{15,8}^{(m)} \\ x_{2,0}^{(m-2)} \oplus x_{7,0}^{(m-2)} &= x_{0,0}^{(m)} \oplus x_{2,0}^{(m)} \oplus x_{6,7}^{(m)} \oplus x_{6,19}^{(m)} \oplus x_{7,26}^{(m)} \oplus x_{8,0}^{(m)} \oplus x_{10,12}^{(m)} \oplus \\ & x_{11,7}^{(m)} \oplus x_{11,19}^{(m)} \oplus x_{12,0}^{(m)} \oplus x_{12,8}^{(m)} \oplus x_{14,0}^{(m)} \oplus x_{15,6}^{(m)} \oplus x_{15,7}^{(m)} \\ x_{3,0}^{(m-2)} \oplus x_{4,0}^{(m-2)} &= x_{1,0}^{(m)} \oplus x_{3,0}^{(m)} \oplus x_{4,26}^{(m)} \oplus x_{7,7}^{(m)} \oplus x_{7,19}^{(m)} \oplus x_{8,7}^{(m)} \oplus x_{8,19}^{(m)} \oplus \\ & x_{9,0}^{(m)} \oplus x_{11,12}^{(m)} \oplus x_{12,6}^{(m)} \oplus x_{12,7}^{(m)} \oplus x_{13,0}^{(m)} \oplus x_{13,8}^{(m)} \oplus x_{15,0}^{(m)} \end{aligned}$$

proof.

See [70]. ■

In [24], the authors showed that using as $\mathcal{ID}$ a single bit at $x_{13,13}^{(0)}$ and $\mathcal{OD}$ at $x_{11,0}^{(3)}$, it is possible to obtain $\varepsilon_d = -0.0272 \approx -\frac{1}{2^{5.2}}$, experimentally. And from Lemma 6.2 it is possible to extend to a 4-round differential-linear correlation with $\varepsilon_L = 1$ when the $\mathcal{OD}$ is $x_{1,0}^{(4)} \oplus x_{11,0}^{(4)} \oplus x_{12,8}^{(4)} \oplus x_{12,0}^{(4)}$. Further, it is possible to extend to a 5-round differential-linear correlation using the last equation from Lemma 6.4 with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. This gives a total differential-linear 5th round correlation of $\varepsilon_d \cdot \varepsilon_L^2 \approx -0.0068 = -\frac{1}{2^{7.2}}$. This leads to a 5 round distinguisher with complexity approximately 2^{16}.

Extending the linear approximation for 3 rounds comes at a cost. As discussed prior to the above lemma, for ChaCha, setting $i = 0$ in Lemma 6.2 allows linear approximation of probability 1 for LSB variables. The cost is thus determined by the non-LSB variables. A simple count of the non-LSB variables in the form (Variable Type, # non-LSB occurrence) gives $(x_a, 3), (x_b, 5), (x_c, 3)$, and $(x_d, 2)$. Now, using the probabilities of Lemma 6.3 and Lemma 6.4, the linear correlation is $\varepsilon_L = 1/2^{1+3\cdot4+5\cdot1+3\cdot2+2\cdot1} = 2^{-26}$. This leads to a 6-round correlation of $\varepsilon_L^2\varepsilon_d \approx \frac{1}{2^{57.2}}$. The distinguisher for this correlation has a complexity of 2^{116}.

In [70], the authors used Lemma 6.5 to derive a distinguisher for 6 rounds. To do that, they found a differential with correlation $\varepsilon_d = 0.00048$ for $(a, b) = (3, 4)$ when the input difference is given by $\Delta x_{14,6}^{(0)} = 1$, and 0 for all remaining bits. Therefore, expanding for 6 rounds from Lemma 6.5 with weights $4, 1, 2, 1$ for x_a, x_b, x_c and x_d, respectively, they got $\varepsilon_L = 1/2^{1+0\cdot4+3\cdot1+3\cdot2+3\cdot1} = 2^{-13}$. Then we have $\varepsilon_d\varepsilon_L^2 \approx 2^{-37.02}$, which leads to an attack against 6 rounds of ChaCha with complexity 2^{75}. This is the currently best known 6-round attack on ChaCha.

6.3 IMPROVED LINEAR APPROXIMATIONS FOR ARX PRIMITIVES

The challenge of finding good linear approximations in ARX-based designs comes from the addition operation, which is responsible for the non-linearity of the design. In 2003, Wallén [42] published a very important paper where a mathematical framework for finding linear approximations of addition modulo 2^n was developed. Since then, several authors used this technique to find linear approximations in ARX-based designs [24].

Therefore, as before, let $\Theta(x, y) = x \oplus y \oplus (x + y)$ be the carry function of the sum $x + y$. Define $\Theta_i(x, y)$ as the i-th bit of $\Theta(x, y)$. By definition, we have $\Theta_0(x, y) = 0$. Using Theorem 3 of [42], we can generate all possible linear approximations with a given correlation. In particular, we will use the following linear approximations:

$$\Pr(\Theta_i(x, y) = y_{i-1}) = \frac{1}{2}\left(1+\frac{1}{2}\right), i > 0 \tag{6.14}$$

and

$$\Pr(\Theta_i(x, y) \oplus \Theta_{i-1}(x, y) = 0) = \frac{1}{2}\left(1+\frac{1}{2}\right), i > 0. \tag{6.15}$$

In previous works of cryptanalysis of ARX ciphers, authors concentrated in finding approximations for particular bits in one round and then repeating the same equations to expand the linear approximation to further rounds (see [24] and [70] for some examples). However, by combining Eqs. 6.14 and 6.15 when attacking ARX ciphers, we can create a strategy to improve linear approximations when considering more rounds. The main idea is that when using Eq. 6.14 in one round, we will create consecutive terms that can be expanded together using Eq. 6.15.

For example, consider the sum $z = x + y$. If we want a linear approximation for the bit z_7, we can use Eq. 6.14 to obtain $z_7 = x_7 \oplus y_7 \oplus \Theta_7(x, y) = x_7 \oplus y_7 \oplus y_6$ with probability 0.75. Since the XOR operation will not change the indexes and the rotation will probably keep y_6 and y_7 adjacent, we can use Eq. 6.15 in the subsequent round to cancel out the non-linear terms rather than expanding them, leading to a linear equation with higher correlation and fewer terms to be expanded further. Next, we will use this technique to find new linear approximations for ChaCha.

6.3.1 Linear Approximations for QR_{chacha}

The first step is to find linear approximations for the QRF of ChaCha. Of course, we already know some of them from previous work (Section 6.2). However, here, we consider adjacent bits and several other combinations that cancel out non-linear terms or use Eq. (6.15). At first glance, these results may seem innocuous, but they prove themselves useful when deriving linear approximations for multiple rounds of ChaCha.

We start with a better linear approximation for $x^{(m-1)}_{a,i}$.

Lemma 6.6 The following holds for $i > 0$

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

proof.

From Eq. (6.9) we have

$$x^{(m-1)}_{a,i} = \mathcal{L}^{(m)}_{a,i} \oplus \Theta_i(x'^{(m-1)}_a, x'^{(m-1)}_b) \oplus \Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \oplus \Theta_i(x^{(m-1)}_a, x^{(m-1)}_b).$$

Using Eq. (6.14) and the Piling-up Lemma, we can write

$$x^{(m-1)}_{a,i} = \mathcal{L}^{(m)}_{a,i} \oplus x'^{(m-1)}_{b,i-1} \oplus \Theta_i(x'^{(m-1)}_c, x^{(m)}_d) \oplus x^{(m-1)}_{b,i-1},$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Using Eq. (6.6), we get

$$x_{a,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus x_{b,i-1}^{\prime(m-1)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \mathcal{L}_{b,i-1}^{(m)} \oplus \Theta_{i-1}(x_c^{\prime(m-1)}, x_d^{(m)}).$$

Using the approximation of Eq. (6.15) and the Piling-up Lemma, we can write

$$x_{a,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus x_{b,i-1}^{\prime(m-1)} \oplus \mathcal{L}_{b,i-1}^{(m)},$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$. Finally, using Eqs. (6.2) and (6.11), we get

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

which completes the proof. ■

Lemma 6.7 For two active input bits in round $m-1$ and multiple active output bits in round m, the following holds for $i > 0$

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

where $(\lambda, \sigma) \in \{(a,3),(b,1),(c,2),(d,1)\}$.

proof.

This proof follows directly from Eqs. (6.6)-(6.9) using the approximation of Eq. (6.15) and the Piling-up Lemma. ■

Lemma 6.8 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

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

proof.

Using Eq. (6.6) and Eq. (6.7), we get

$$\begin{aligned} x_{b,\lambda}^{(m-1)} \oplus x_{c,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = \; & \mathcal{L}_{b,\lambda}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \Theta_\lambda(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \\ & \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_c^{(m-1)}, x_d^{\prime(m-1)}) \oplus \\ & \Theta_{i-1}(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_{i-1}(x_c^{(m-1)}, x_d^{\prime(m-1)}). \end{aligned}$$

Canceling out common factors and using the approximation of Eq. (6.15), we can write

$$x_{b,\lambda}^{(m-1)} \oplus x_{c,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \Theta_{\sigma+1}(x_c'^{(m-1)}, x_d^{(m)}).$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Using Eq. (6.14), we get

$$x_{b,\lambda}^{(m-1)} \oplus x_{c,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus x_{d,\sigma}^{(m)},$$

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

LEMMA 6.9 For multiple active input bits in round $m-1$ and multiple active output bits in round m, the following linear approximations hold for ChaCha with probability

$\frac{1}{2}\left(1+\frac{1}{2^k}\right)$:

$$x_{b,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)} \qquad k=1, i>0 \tag{6.16}$$

$$x_{a,i}^{(m-1)} \oplus x_{b,i}^{(m-1)} = \begin{array}{l}\mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{b,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \\ x_{b,i+6}^{(m)} \oplus x_{c,i-1}^{(m)} \oplus x_{d,i-2}^{(m)}\end{array} \qquad k=3, i>1 \tag{6.17}$$

$$x_{a,1}^{(m-1)} \oplus x_{b,1}^{(m-1)} = \mathcal{L}_{a,1}^{(m)} \oplus \mathcal{L}_{b,0}^{(m)} \oplus \mathcal{L}_{b,1}^{(m)} \oplus x_{b,7}^{(m)} \oplus x_{c,0}^{(m)} \qquad k=2 \tag{6.18}$$

$$x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \begin{array}{l}\mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{b,i-1}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus \\ x_{b,i+6}^{(m)} \oplus x_{c,i-1}^{(m)} \oplus x_{d,i-2}^{(m)} \oplus x_{d,i+7}^{(m)}\end{array} \qquad k=4, i>1 \tag{6.19}$$

$$x_{a,1}^{(m-1)} \oplus x_{c,1}^{(m-1)} = \begin{array}{l}\mathcal{L}_{a,1}^{(m)} \oplus \mathcal{L}_{b,0}^{(m)} \oplus \mathcal{L}_{c,1}^{(m)} \oplus x_{a,0}^{(m)} \oplus \\ x_{b,7}^{(m)} \oplus x_{c,0}^{(m)} \oplus x_{d,8}^{(m)}\end{array} \qquad k=3 \tag{6.20}$$

$$x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \mathcal{L}_{b,i-1}^{(m)} \qquad k=2, i>1 \tag{6.21}$$

$$\begin{array}{c}x_{a,i-1}^{(m-1)} \oplus \\ x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)}\end{array} = \begin{array}{l}\mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \\ x_{d,i-2}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)}\end{array} \qquad k=4, i>1 \tag{6.22}$$

$$\begin{array}{c}x_{a,i}^{(m-1)} \oplus \\ x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)}\end{array} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus x_{d,i-2}^{(m)}, \qquad k=3, i>1 \tag{6.23}$$

$$\begin{array}{c}x_{b,i-1}^{(m-1)} \oplus \\ x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)}\end{array} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus x_{d,i-1}^{(m)}, \qquad k=2, i>1 \tag{6.24}$$

$$\begin{array}{c}x_{b,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus \\ x_{c,i-1}^{(m-1)} \oplus x_{c,i}^{(m-1)}\end{array} = \begin{array}{l}\mathcal{L}_{b,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \\ \mathcal{L}_{c,i-1}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)},\end{array} \qquad k=1, i>1 \tag{6.25}$$

$$\begin{array}{c}x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus \\ x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)}\end{array} = \begin{array}{l}\mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \\ \mathcal{L}_{c,i-1}^{(m)} \oplus x_{a,i-2}^{(m)} \oplus x_{d,i+6}^{(m)},\end{array} \qquad k=3, i>1 \tag{6.26}$$

$$\begin{array}{c}x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus \\ x_{c,i-1}^{(m-1)} \oplus x_{d,i}^{(m-1)} \oplus \\ x_{d,i-1}^{(m-1)}\end{array} = \begin{array}{l}\mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \\ \mathcal{L}_{d,i-1}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus x_{d,i-1}^{(m)} \oplus \\ x_{a,i-2}^{(m)} \oplus x_{d,i+6}^{(m)},\end{array} \qquad k=3, i>2 \tag{6.27}$$

proof.

The proof for each equation follows the same basic steps: (1) cancel common factors; (2) cancel adjacent non-linear terms using Eq. (6.15), updating the probability using the Piling-Up Lemma; (3) substitute the remaining non-linear terms using Eq. (6.14), updating the probability using the Piling-Up Lemma. For completeness, we list all proofs in Appendix A.2. ■

6.3.2 Linear Approximations for Multiple Rounds of ChaCha

In this section, we use the proposed technique to construct several new linear approximations for the stream cipher ChaCha, which are useful to construct better distinguishers. We developed a program (available in <https://github.com/MurCoutinho/cryptanalysisChaCha.git>) that makes the process of finding linear approximations partly automatic. Our program is capable of expanding the equations, and after statistically verifying the correlation, it outputs the resulting linear approximation in LaTeXcode.

We start using the result of Coutinho and Souza [70]. We only consider the equation for $x^{(3)}_{3,0} \oplus x^{(3)}_{4,0}$ of Lemma 6.5, but the same reasoning could be applied to any other equation in that lemma. Then, we have

$$\begin{aligned} x^{(3)}_{3,0} \oplus x^{(3)}_{4,0} = \;& x^{(5)}_{1,0} \oplus x^{(5)}_{3,0} \oplus x^{(5)}_{4,26} \oplus x^{(5)}_{7,7} \oplus x^{(5)}_{7,19} \oplus x^{(5)}_{8,7} \oplus x^{(5)}_{8,19} \oplus \\ & x^{(5)}_{9,0} \oplus x^{(5)}_{11,12} \oplus x^{(5)}_{12,6} \oplus x^{(5)}_{12,7} \oplus x^{(5)}_{13,0} \oplus x^{(5)}_{13,8} \oplus x^{(5)}_{15,0} \end{aligned} \tag{6.28}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$.

As presented in Section 6.2, to expand the equation to the 6th round, we could use only Lemma 6.3 as proposed in [24]. In this case, we have weights $4, 1, 2, 1$ for x_a, x_b, x_c, and x_d, respectively, and a count of $(x_a, 0)$, $(x_b, 3)$, $(x_c, 3)$, and $(x_d, 3)$. Thus, the linear correlation is $\varepsilon_L = 1/2^{1+0\cdot 4+3\cdot 1+3\cdot 2+3\cdot 1} = 2^{-13}$. However, we can do better with the new technique proposed in Section 6.3. This leads us to the following lemma

Lemma 6.10 The following linear approximation holds with probability $\frac{1}{2}\left(1+\frac{1}{2^8}\right)$

$$\begin{aligned} x^{(3)}_{3,0} \oplus x^{(3)}_{4,0} = \;& x^{(6)}_{0}[0,16] \oplus x^{(6)}_{1}[0,6,7,11,12,22,23] \oplus x^{(6)}_{2}[0,6,7,8,16,18,\\ & 19,24] \oplus x^{(6)}_{4}[7,13,19] \oplus x^{(6)}_{5}[7] \oplus x^{(6)}_{6}[7,13,14,19] \oplus \\ & x^{(6)}_{7}[6,7,14,15,26] \oplus x^{(6)}_{8}[0,7,8,19,31] \oplus x^{(6)}_{9}[0,6,12,26] \oplus \\ & x^{(6)}_{10}[0] \oplus x^{(6)}_{11}[6,7] \oplus x^{(6)}_{12}[0,11,12,19,20,30,31] \oplus \\ & x^{(6)}_{13}[0,14,15,24,26,27] \oplus x^{(6)}_{14}[8,25,26] \oplus x^{(6)}_{15}[24]. \end{aligned}$$

proof.

See Appendix A.2. ■

Computational Result 6.1 The linear approximation of Lemma 6.10 holds computationally with $\varepsilon_{L_0} = 0.006942 \approx 2^{-7.17}$. This correlation was verified using 2^{38} random samples.

In [24], the authors remarked that an expansion of this method to 7 rounds would be unlikely to be useful. Indeed, if we only apply Lemma 6.3 (which are the linear approximations proposed by Choudhuri and Maitra), we would have $(x_a, 14)$, $(x_b, 13)$, $(x_c, 9)$, $(x_d, 15)$. Therefore, the aggregated correlation would be $\varepsilon_L = 1/2^{7+14\cdot4+13\cdot1+9\cdot2+15\cdot1} = 2^{-109}$. Thus, using this linear expansion in a differential-linear attack would lead to a distinguisher with complexity no less than 2^{436}. However, using our new linear approximations, we can get a much better result.

LEMMA 6.11 The following linear approximation holds with probability $\frac{1}{2}\left(1+\frac{1}{2^{55}}\right)$

$$x_{3,0}^{(3)} \oplus x_{4,0}^{(3)} = x_0^{(7)}[0,3,4,7,8,11,12,14,15,18,20,27,28] \oplus x_1^{(7)}[0,5,7,8,10, 11,14,15,16,22,23,24,25,27,30,31] \oplus x_2^{(7)}[6,7,9,10,16,18,19, 25,26] \oplus x_3^{(7)}[6,7,8,24] \oplus x_4^{(7)}[0,2,3,5,18,22,23,27] \oplus x_5^{(7)}[1,2, 9,10,13,14,18,21,22,25,29,30] \oplus x_6^{(7)}[2,3,5,7,10,11,13,14,19, 22,23,27,30,31] \oplus x_7^{(7)}[1,2,13,25,26,30,31] \oplus x_8^{(7)}[8,11,13,20, 25,27,28,30,31] \oplus x_9^{(7)}[2,3,6,7,14,15,18,27] \oplus x_{10}^{(7)}[0,3,4,8,12, 13,14,18,20,27,28,30] \oplus x_{11}^{(7)}[6,14,15,18,19,23,24,27] \oplus x_{12}^{(7)}[3,4,6,11,13,22,23,24,26,27,30,31] \oplus x_{13}^{(7)}[1,2,6,7,8,10, 11,13,14,16,18,20,22,23,24,25,26] \oplus x_{14}^{(7)}[0,6,13,14,15,16, 23,24] \oplus x_{15}^{(7)}[16,25,26].$$

proof.

See Appendix A.2. ■

COMPUTATIONAL RESULT 6.2 The linear approximation of Eq. (A.1) holds computationally with $\varepsilon_{L_1} = 0.000301 \approx 2^{-11.70}$. This correlation was verified using 2^{42} random samples.

COMPUTATIONAL RESULT 6.3 The linear approximation of Eq. (A.2) holds computationally with $\varepsilon_{L_2} = 0.000100 \approx 2^{-13.29}$. This correlation was verified using 2^{42} random samples.

COMPUTATIONAL RESULT 6.4 The linear approximation of Eq. (A.3) holds computationally with $\varepsilon_{L_3} = 0.000051 \approx 2^{-14.26}$. This correlation was verified using 2^{42} random samples.

COMPUTATIONAL RESULT 6.5 The linear approximation of Eq. (A.4) holds computationally with $\varepsilon_{L_4} = 0.0625 \approx 2^{-4}$. This correlation was verified using 2^{38} random samples.

6.4 IMPROVED DIFFERENTIAL-LINEAR ATTACKS AGAINST CHACHA

6.4.1 New Differentials

In this section, we present new differentials for 3.5 rounds of ChaCha. As in previous works, these differential correlations were found experimentally. To find these correlations we used the technique proposed by Beierle et al. at Crypto 2020 [26], and described in Section 3.3. Here, the cipher is divided into the sub ciphers E_1 covering 1 round and E_2 covering 2.5 rounds to find a differential path for 3.5 rounds. Thus we want a particular differential characteristic of the form

$$\Delta X^{(0)} \xrightarrow{\text{1 round}} \Delta X^{(1)} \xrightarrow{\text{2.5 rounds}} \Delta X^{(3.5)}.$$

The idea is to generate consistent $\Delta X^{(1)}$ whose Hamming weight is minimized. In [26], the authors showed that the following differential characteristic occurs with probability 2^{-5} on average for the QRF of ChaCha

$$\begin{aligned}\Delta X^{(0)} = (([]),([]),([]),([i])) \to \Delta X^{(1)} = \ & (([i+28]),([i+31,i+23,i+11,\\ & i+3]),([i+24,i+16,i+4]),\\ & ([i+24,i+4])).\end{aligned} \tag{6.29}$$

From there we computed $\Delta X^{(3.5)}$ by generating random states $X^{(1)}$ and $X'^{(1)}$ and statistically testing for correlations in particular bits of $\Delta X^{(3.5)}$. We note that this procedure is computationally intensive as some of the correlations are very small. For some bits, we executed this procedure up to 2^{50} pairs of random states in the first round. To achieve this amount of computation we used 8 NVIDIA GPUs (RTX 2080ti). As in the referred paper, we used $i = 6$. Also, we fixed the differential of Eq. (6.29) in the third column of the state matrix. Table 6.2 shows the results[1].

[1]Since the first version of this paper was published, several independent researches reviewed our results and code. We would like to thank Juan C. G. Vásquez (juan.grados@tii.ae) for identifying an error in the code we made publicly available. That error affected the results of this table in the first version of the paper. Dey et al. [**??**] independently noticed that the results reported were not accurate and computed an alternative version of this table. However, we were only able to reproduce the results reported for $\Delta x_{0,0}^{(3.5)}$ and $\Delta x_{13,0}^{(3.5)}$. More precisely, it seems that Dey et al. had inaccuracies of their own, caused by a small number of samples (2^{37}) which is not enough to compute the true correlation for these bits. After correcting the code, we could not find

$\mathcal{OD}$	$\lvert\varepsilon_d\rvert$
$\Delta x_{0,0}^{(3.5)}$	0.00002797
$\Delta x_{13,0}^{(3.5)}$	0.000003032

Table 6.2 – New differentials after 3.5 rounds, starting from $\Delta X^{(1)}$ in the third column of the state matrix with $i = 6$ in Eq. (6.29).

6.4.2 Distinguishers

Using the linear approximations of Lemma 6.10 and Lemma 6.11, the differential correlation $\varepsilon_d = 0.00048$ for $(a, b) = (3, 4)$ described in [70], and the estimated correlations from the Computational Results 6.1-6.5, we get $\varepsilon_d \varepsilon_{L_0}^2 \approx 2^{-25.37}$ which gives us a distinguisher for 6 rounds of ChaCha with complexity less than 2^{51}. Also, we get $\varepsilon_d(\varepsilon_{L_0}\varepsilon_{L_1}\varepsilon_{L_2}\varepsilon_{L_3}\varepsilon_{L_4})^2 \approx 2^{-111.86}$ which gives us a distinguisher for 7 rounds of ChaCha with complexity less than 2^{224}.

6.4.3 New Attack using Probabilistic Neutral Bits (PNBs)

We can implement new attacks for 7 rounds of ChaCha using the PNB technique by considering the new differential correlation for $\Delta_{13,0}^{(3.5)}$ presented in Table 6.2. Using Eq. (6.4) it is easy to see that we have $x_{13,0}^{(3.5)} = x_{2,0}^{(4)} \oplus x_{13,8}^{(4)}$. Therefore, we consider $\mathcal{ID}$ given by Eq. (6.29) with $i = 6$ and $\mathcal{OD}$ $x_{2,0}^{(4)} \oplus x_{13,8}^{(4)}$. Using $\gamma = 0.35$ we found 83 PNBs, and we obtained $\varepsilon_a = 0.000509$. From that, we get an attack with data complexity of $2^{64.59}$ and time complexity $2^{237.59}$. As in [26], we have to repeat this attack 2^5 times on average. Thus, the final attack has data complexity of $2^{69.58}$ and time complexity, which does not improve previous results. Bellow we list all PNBs:

$$\text{PNB} = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 19, 20, 21, 31, 32, 33, 34, 35, 36, 39, 43, 47, 48, \\ 49, 53, 54, 55, 59, 63, 67, 68, 69, 70, 71, 72, 73, 89, 90, 95, 99, 100, 103, \\ 104, 105, 123, 124, 125, 126, 127, 128, 129, 130, 140, 141, 142, 152, 153, 154, \\ 155, 156, 157, 158, 159, 168, 169, 170, 174, 175, 176, 184, 185, 186, 187, 188, \\ 189, 190, 191, 192, 193, 210, 223, 248, 255).$$

significant results for $\Delta x_{1,0}^{(3.5)}$, $\Delta x_{12,0}^{(3.5)}$ and $\Delta x_{5,0}^{(3.5)}$ as previously reported, even considering 2^{52} samples.

7 BIDIRECTIONAL LINEAR EXPANSIONS TO IMPROVE DIFFERENTIAL-LINEAR ATTACKS AGAINST SALSA20

7.1 INTRODUCTION

In this chapter, we propose a technique called Bidirectional Linear Expansions (BLE), which can be used to improve attacks against Salsa20. More specifically, previous work focused in finding significant differential correlations computationally by leveraging single bit differentials and then expanding it forward using linear approximations. Instead, BLE can be used to expand a single bit in both forward and backward directions. Because of that, in the differential part, we need to find a correlation for a combination of bits instead of just one. As we show, by applying the techniques proposed in Chapter 6 (also see [27]), we can make the forward linear approximations very efficient, allowing us to present the first differential-linear distinguisher against Salsa20/7 and Salsa20/8. Additionally, the differentials obtained using BLE can be leveraged to improve key recovery attacks using PNB.

Using these techniques, we propose new key recovery attacks against Salsa20/7 and Salsa20/8 (see Table 7.1) which significantly improve time complexity to $2^{122.63}$ (from 2^{137}) and $2^{219.56}$ (from $2^{243.93}$), respectively. Additionally, we propose the first differential linear distinguisher for Salsa20/7 and Salsa20/8 with time and data complexity of $2^{108.98}$ and $2^{215.62}$, respectively. We summarize our findings along with other significant attacks for comparison in Table 7.1.

Organization of the chapter. This chapter is divided as follows: in Section 7.2, we present a review of previous attacks against Salsa. In Section 7.3, we propose new differentials and linear approximations for Salsa20. In Section 7.4, we present new attacks against Salsa20, using Probabilistic Neutral Bits and differential-linear cryptanalysis.

Type	Rounds	Time Complexity	Data Complexity	Reference
PNB	7	2^{151}	2^{26}	[19]
		2^{148}	2^{24}	[20]
		2^{139}	2^{32}	[24]
		2^{137}	2^{61}	[24]
		$2^{122.63}$	$2^{103.61}$	Proposed
PNB	8	2^{251}	2^{31}	[19]
		2^{250}	2^{27}	[20]
		$2^{244.9}$	2^{96}	[24]
		$2^{243.93}$	$2^{30.86}$	[23]
		$2^{219.56}$	$2^{115.52}$	Proposed
Diff-lin Distinguisher	5	2^{8}	2^{8}	[24]
Diff-lin Distinguisher	6	2^{32}	2^{32}	[24]
Diff-lin Distinguisher	7	$2^{108.98}$	$2^{108.98}$	Proposed
Diff-lin Distinguisher	8	$2^{215.62}$	$2^{215.62}$	Proposed

Table 7.1 – The best attacks against Salsa20 with 256-bit key.

7.2 LINEAR APPROXIMATIONS FOR SALSA

In the following, we review the work of [24] using the notation of Coutinho and Souza [27]. Let $\Theta(x, y) = x \oplus y \oplus (x + y)$ be the carry function of the sum $x + y$. Define $\Theta_i(x, y)$ as the i-th bit of $\Theta(x, y)$. By definition, we have $\Theta_0(x, y) = 0$. We can write the QRF equations of Salsa20 (Eq. 2.4) as

$$\begin{aligned}
x_{b,i}^{(m)} &= x_{b,i}^{(m-1)} \oplus x_{a,i-7}^{(m-1)} \oplus x_{d,i-7}^{(m-1)} \oplus \\
&\quad \Theta_{i-7}(x_d^{(m-1)}, x_a^{(m-1)}) && (7.1)\\
x_{c,i}^{(m)} &= x_{c,i}^{(m-1)} \oplus x_{b,i-9}^{(m)} \oplus x_{a,i-9}^{(m-1)} \oplus \\
&\quad \Theta_{i-9}(x_a^{(m-1)}, x_b^{(m)}) && (7.2)\\
x_{d,i}^{(m)} &= x_{d,i}^{(m-1)} \oplus x_{c,i-13}^{(m)} \oplus x_{b,i-13}^{(m)} \oplus \\
&\quad \Theta_{i-13}(x_c^{(m)}, x_b^{(m)}) && (7.3)\\
x_{a,i}^{(m)} &= x_{a,i}^{(m-1)} \oplus x_{d,i-18}^{(m)} \oplus x_{c,i-18}^{(m)} \oplus \\
&\quad \Theta_{i-18}(x_d^{(m)}, x_c^{(m)}) && (7.4)
\end{aligned}$$

Inverting these equations and changing to positive indexes, we get:

$$x_{a,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \Theta_{i+14}(x_d^{(m)}, x_c^{(m)}) \tag{7.5}$$

$$x_{d,i}^{(m-1)} = \mathcal{L}_{d,i}^{(m)} \oplus \Theta_{i+19}(x_c^{(m)}, x_b^{(m)}) \tag{7.6}$$

$$x_{c,i}^{(m-1)} = \mathcal{L}_{c,i}^{(m)} \oplus \Theta_{i+23}(x_a^{(m-1)}, x_b^{(m)}) \oplus \Theta_{i+5}(x_d^{(m)}, x_c^{(m)}) \tag{7.7}$$

$$x_{b,i}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \Theta_{i+25}(x_d^{(m-1)}, x_a^{(m-1)}) \oplus \Theta_{i+7}(x_d^{(m)}, x_c^{(m)}) \oplus \Theta_{i+12}(x_c^{(m)}, x_b^{(m)}), \tag{7.8}$$

where

$$\mathcal{L}_{a,i}^{(m)} = x_{a,i}^{(m)} \oplus x_{d,i+14}^{(m)} \oplus x_{c,i+14}^{(m)} \tag{7.9}$$

$$\mathcal{L}_{b,i}^{(m)} = x_{b,i}^{(m)} \oplus x_{a,i+25}^{(m)} \oplus x_{d,i+7}^{(m)} \oplus x_{c,i+7}^{(m)} \oplus x_{d,i+25}^{(m)} \oplus x_{c,i+12}^{(m)} \oplus x_{b,i+12}^{(m)} \tag{7.10}$$

$$\mathcal{L}_{c,i}^{(m)} = x_{c,i}^{(m)} \oplus x_{b,i+23}^{(m)} \oplus x_{a,i+23}^{(m)} \oplus x_{d,i+5}^{(m)} \oplus x_{c,i+5}^{(m)} \tag{7.11}$$

$$\mathcal{L}_{d,i}^{(m)} = x_{d,i}^{(m)} \oplus x_{c,i+19}^{(m)} \oplus x_{b,i+19}^{(m)}. \tag{7.12}$$

From these equations, it is possible to derive the following result:

LEMMA 7.1 For Salsa20 QRF, the following linear approximations hold

Equation	Probability	Condition
$x_{a,18}^{(m-1)} = \mathcal{L}_{a,18}^{(m)}$	1	-
$x_{a,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus x_{c,i+13}$	$\frac{1}{2}(1+\frac{1}{2})$	$i \neq 18$
$x_{d,13}^{(m-1)} = \mathcal{L}_{d,13}^{(m)}$	1	-
$x_{d,i}^{(m-1)} = \mathcal{L}_{d,i}^{(m)} \oplus x_{b,i+18}$	$\frac{1}{2}(1+\frac{1}{2})$	$i \neq 13$
$x_{c,9}^{(m-1)} = \mathcal{L}_{c,9}^{(m)} \oplus x_{c,13}^{(m)}$	$\frac{1}{2}(1+\frac{1}{2})$	-
$x_{c,27}^{(m-1)} = \mathcal{L}_{c,27}^{(m)} \oplus x_{b,17}^{(m)}$	$\frac{1}{2}(1+\frac{1}{2})$	-
$x_{c,i}^{(m-1)} = \mathcal{L}_{c,i}^{(m)} \oplus x_{a,i+22}^{(m)}$	$\frac{1}{2}(1-\frac{1}{4})$	$i \neq 9, 27$
$x_{b,7}^{(m-1)} = \mathcal{L}_{b,7}^{(m)} \oplus x_{c,13}^{(m)} \oplus x_{b,18}^{(m)}$	$\frac{1}{2}(1+\frac{1}{4})$	-
$x_{b,20}^{(m-1)} = \mathcal{L}_{b,20}^{(m)} \oplus x_{a,12}^{(m)}$	$\frac{1}{2}(1-\frac{1}{4})$	-
$x_{b,25}^{(m-1)} = \mathcal{L}_{b,25}^{(m)} \oplus x_{d,17}^{(m)}$	$\frac{1}{2}(1-\frac{1}{4})$	-
$x_{b,i}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus x_{a,i+24}^{(m)} \oplus x_{b,i+11}^{(m)}$	$\frac{1}{2}(1-\frac{1}{8})$	$i \neq 7, 20, 25$

proof.

See Lemmas 2 and 7 of [24]. ■

7.3 BIDIRECTIONAL LINEAR EXPANSIONS TO DIFFERENTIAL-LINEAR ATTACKS

In this section, we propose new differentials and linear approximations for Salsa20. As we see in Section 7.4, combining these results, we can improve attacks against Salsa20. This section is divided in three parts: in subsection 7.3.1, we propose a new technique to construct differential-linear distinguishers for ARX ciphers, in subsection 7.3.2, we propose a new differential reaching 5 rounds of Salsa, and in subsection 7.3.3, we propose new linear approximations for Salsa.

7.3.1 Proposed technique

Previous work on the cryptanalysis of Salsa and ChaCha used a computational approach to find significant correlations on the differential part of the attacks. To do so, authors considered an input differential $\Delta X^{(0)}$ and used several random simulations to estimate a correlation for a single bit $\Delta x_{i,j}^{(m)}$. From this point, this single bit was expanded into several bits using linear approximations, like in the following diagram:

$$\begin{tikzcd}
 & & \Delta x_{i_1,j_1}^{(m+1)} \\
 & & \Delta x_{i_2,j_2}^{(m+1)} \\
\Delta X^{(0)} \arrow[r] & \Delta x_{i,j}^{(m)} \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \cdots \\
 & & \Delta x_{i_p,j_p}^{(m+1)}
\end{tikzcd}$$

In this work, we propose a different approach. More precisely, we expand a single bit in both forward and backward directions. Therefore, in the differential part, we need to find a correlation for a combination of bits instead of just one. This approach leads to worst differential correlations; however, it improves the linear correlations. Since the linear part has a higher weight on the complexity of the attack, the proposed technique leads to better results overall. We illustrate the proposed technique in the following diagram:

$$\begin{tikzcd}
 & \Delta x^{(m)}_{r_1,s_1} & & \Delta x^{(m+2)}_{i_1,j_1} \\
 & \Delta x^{(m)}_{r_2,s_2} & & \Delta x^{(m+2)}_{i_2,j_2} \\
\Delta X^{(0)} \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \cdots & \Delta x^{(m+1)}_{i,j} \arrow[uul] \arrow[ul] \arrow[l] \arrow[dl] \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \cdots \\
 & \Delta x^{(m)}_{r_p,s_p} & & \Delta x^{(m+2)}_{i_p,j_p}
\end{tikzcd}$$

7.3.2 Proposed differential for 5 rounds of Salsa

In this section, we present a new single bit differential correlation for 5 rounds of Salsa, constructed by applying the technique proposed in the previous section. To do so, first, notice from Eq. (7.1) that we can write

$$x^{(5)}_{b,7} = x^{(4)}_{b,7} \oplus x^{(4)}_{a,0} \oplus x^{(4)}_{d,0}, \tag{7.13}$$

with probability 1, where $(a, b, d) \in \{(0, 4, 12), (5, 13, 1), (10, 2, 6), (15, 7, 11)\}$. Using this relationship, we find a correlation for a bit in the fifth round $x^{(5)}_{b,7}$ by combining the correlation of three other bits in the fourth round.

To achieve this result, we start from the single bit $\mathbb{ID}$ of $\Delta x^{(0)}_{7,31} = 1$, i.e.,

$$\Delta X^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0x80000000 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

proposed by Aumasson et al. [19], which is the one that provides the highest correlations presented in the literature. However, instead of relying on computational results only, we expanded the first round theoretically and used the techniques proposed by Beierle et al. [26] (see Section 3.3) to find differentials with amplified probabilities. To do so, note that when dealing with differentials in an ARX cipher, we must consider three different operations. For the XOR, the analysis is simple, since if $Z = X \oplus Y$ and $Z' = X' \oplus Y'$ then, $\Delta Z = \Delta X \oplus \Delta Y$. Also, for the Rotation, if $Z = X \ggg R$ and $Z' = X' \ggg R$ then $\Delta Z = \Delta X \ggg R$. For the Addition, however, things are trickier because it is a non-linear operation in the bits of the arguments.

In this work, we solve this problem by applying the techniques proposed by Lipmaa and Moriai on efficient algorithms for computing differential properties of addition [41]. In the referred work, the authors define the Differential Probability of Addition (DPA) modulo 2^n as a triplet of two input and one output differences, denoted as $(\alpha, \beta \to \gamma)$, where $\alpha, \beta, \gamma \in \mathbb{F}_2^n$, and is defined as

$$\begin{aligned}\mathrm{DP}^{+}(\delta) = &\,\mathrm{DP}^{+}(\alpha, \beta \to \gamma) := \\ &\Pr_{x,y}[(x+y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma].\end{aligned} \tag{7.14}$$

One important question is how to find γ such that $\mathrm{DP}^{+}(\delta)$ is maximum given α and β. In other words, we want to find $\mathrm{DP}^{+}_{\max}(\alpha, \beta) := \max_{\gamma} \mathrm{DP}^{+}(\alpha, \beta \to \gamma)$. In [41], the authors provide two important algorithms to compute $\mathrm{DP}^{+}_{\max}(\alpha, \beta)$. Specifically, Algorithm 3 of [41] returns all (α, β)-optimal output differences γ, and Algorithm 4 of [41] finds an (α, β)-optimal γ in log-time.

Thus, starting from the $\mathbb{ID}$ given by $\Delta X^{(0)}$, we propagated the differential using the algorithms from [41] and chose the one that minimized the hamming weight, from this we get:

$$\Psi = \Delta X^{(1)} = \begin{pmatrix} 0 & 0 & 0 & 0x00000000 \\ 0 & 0 & 0 & 0x80000000 \\ 0 & 0 & 0 & 0x00001000 \\ 0 & 0 & 0 & 0x40020000 \end{pmatrix}.$$

The probability that $\Delta X^{(0)}$ leads to $\Delta X^{(1)}$ is 2^{-1}. To compute this probability, we used Algorithm 2 of [41]. At this point, we used the strategy of Beierle et al. [26] (see Section 3.3) to find differentials with amplified probabilities. We may apply this technique because, as with ChaCha, the QRF of Salsa is independently applied to each column in the first round. Therefore, when the output difference of one QRF is restricted, the input of other three QR functions are trivially independent of the output difference. It implies that we have 96 independent bits, and we can easily amplify the probability of the differential-linear distinguisher.

We summarize the differential part of the proposed attacks in the diagram of Figure 7.1. In the next section, we present the linear expansion for the bit $x^{(5)}_{b,7}$ to complete the differential-linear distinguisher.

$$\begin{tikzcd}
 & & \Delta x^{(m)}_{b,7} & \\
\Delta X^{(0)} \arrow[r] & \Psi \arrow[ur] \arrow[r] \arrow[dr] & \Delta x^{(m)}_{a,0} & \Delta x^{(5)}_{b,7} \arrow[ul] \arrow[l] \arrow[dl] \\
 & & \Delta x^{(m)}_{d,0} &
\end{tikzcd}$$

Figure 7.1 – Differential part of the proposed attack.

7.3.3 New linear approximations for Salsa20

In this section, we develop new linear approximations for Salsa. Here we use Eqs. (6.14) and (6.15) defined in the previous chapter.

Then we propose the following Lemma:

Lemma 7.2 For two active input bits in round $m-1$ and multiple active output bits in round m of Salsa20, the following holds for $i \notin \mathcal{I}$

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

where $(\lambda, \sigma, \mathcal{I}) \in \{(a, 1, \{18\}), (b, 3, \{7, 20, 25\}), (c, 2, \{9, 27\}), (d, 1, \{13\})\}$.

proof.

This proof follows from Eqs. (7.5)-(7.8) by noting that always, we have pair with the form $\Theta_i(x) \oplus \Theta_{i-1}(x)$. When $i > 1$, we apply the approximation of Eq. (6.15) to get $\Theta_i(x) \oplus \Theta_{i-1}(x) = 0$ with probability $\frac{1}{2}(1+\frac{1}{2})$. When $i = 1$, we apply Eq. (6.15) to get $\Theta_1(x) \oplus \Theta_0(x) = \Theta_1(x) = 0$ again with probability $\frac{1}{2}(1+\frac{1}{2})$. When $i = 0$, $\Theta_i(x) \oplus \Theta_{i-1}(x) \neq 0$, thus we exclude these indexes. All that is left is to use the Piling-Up Lemma to combine the probabilities. ■

Next, we consider new linear approximations to the bit $x_{4,7}^{(5)}$.

Lemma 7.3 The following linear approximation holds with probability $\frac{1}{2}\left(1-\frac{1}{2^6}\right)$

$$\begin{aligned} x_{4,7}^{(5)} = \; & x_0^{(7)}[0] \oplus x_2^{(7)}[12,13] \oplus x_3^{(7)}[17] \oplus \\ & x_4^{(7)}[7,18,19] \oplus x_6^{(7)}[25,26] \oplus x_7^{(7)}[26,31] \oplus \\ & x_8^{(7)}[13,14,19] \oplus x_{11}^{(7)}[31] \oplus \\ & x_{12}^{(7)}[0,14] \oplus x_{14}^{(7)}[12,13] \oplus x_{15}^{(7)}[16,17]. \end{aligned}$$

proof.

From $x_{4,7}^{(5)}$, we use the expansion for $x_{d,i}$ of Lemma 7.1 to get $x_{4,7}^{(5)} = x_{4,7}^{(6)} \oplus x_{6,25}^{(6)} \oplus x_{6,26}^{(6)} \oplus x_{7,26}^{(6)}$, with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Then, we use the expansion for $x_{b,7}$ and $x_{c,i}$ of Lemma 7.1 to get $x_{4,7}^{(6)} = \mathcal{L}_{4,7}^{(m)} \oplus x_{8,13}^{(m)} \oplus x_{4,18}^{(m)}$ with probability $\frac{1}{2}\left(1+\frac{1}{4}\right)$, and $x_{7,26}^{(6)} = \mathcal{L}_{7,26}^{(m)} \oplus x_{15,16}^{(m)}$ with probability $\frac{1}{2}\left(1-\frac{1}{4}\right)$. Additionally, using Lemma 7.2 we get $x_{6,25}^{(6)} \oplus x_{6,26}^{(6)} = \mathcal{L}_{6,25}^{(7)} \oplus \mathcal{L}_{6,26}^{(7)}$, with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Finally, using the Piling-Up Lemma to combine the probabilities completes the proof. ■

LEMMA 7.4 The following linear approximation holds with probability $\frac{1}{2}\left(1+\frac{1}{2^{34}}\right)$

$$\begin{aligned} x_{4,7}^{(5)} = \; & x_0^{(8)}[0,3,4] \oplus x_2^{(8)}[4,12,14,17,18] \oplus \\ & x_3^{(8)}[14,18] \oplus x_4^{(8)}[0,1,4,7,31] \oplus \\ & x_5^{(8)}[16,17,18,19,21,22] \oplus x_6^{(8)}[17,22] \oplus \\ & x_7^{(8)}[0,1,4] \oplus x_8^{(8)}[6,11,13,14,18,24] \oplus \\ & x_9^{(8)}[6,18,19] \oplus x_{10}^{(8)}[4,5,9,10,23,24] \oplus \\ & x_{11}^{(8)}[4,5,11,31] \oplus \\ & x_{12}^{(8)}[11,12,14,25,26,30,31] \oplus \\ & x_{13}^{(8)}[0,7,12,21,26,30] \oplus \\ & x_{14}^{(8)}[12,13,21,25,30,31] \oplus \\ & x_{15}^{(8)}[6,7,16,17,24,25]. \end{aligned}$$

proof.

See Appendix A.2. ■

7.4 RESULTS

Next, we use the techniques proposed in Section 7.3 to improve the attacks against Salsa20. This section is divided in three parts: in Section 7.4.1, we present some computational results validating the theory developed in Section 7.3. In Section 7.4.2, we present new differential-linear attacks against 7 and 8 rounds of Salsa. Finally, in Section 7.4.3, we use the techniques from Aumasson et al. [19] to derive new key-recovery attacks using PNB.

7.4.1 Computational results

Considering Figure 7.1, we need to estimate the transition probability from Ψ to $\Delta x_{b,7}^{(5)}$. We performed this task computationally, and we achieve the best results when considering $b = 4$. Thus, consider the following computational result:

COMPUTATIONAL RESULT 7.1 The following differentials were found computationally using 2^{45} random samples.

$\mathbb{ID}$	$\mathbb{OD}$	Correlation
$\Delta X^{(1)} = \Psi$	$\Delta x_{0,0}^{(4)}$	-0.000001515
$\Delta X^{(1)} = \Psi$	$\Delta x_{4,7}^{(4)}$	-0.00085
$\Delta X^{(1)} = \Psi$	$\Delta x_{12,0}^{(4)}$	0.000167

From this result, we can use the Piling-Up Lemma to reach a differential correlation from round 1 to round 5 of Salsa. More precisely, we can write

$$\Pr(\Delta x_{4,7}^{(5)} = 0 | \Delta X^{(1)} = \Psi) = \frac{1}{2}(1 + \varepsilon_d), \tag{7.15}$$

where $\varepsilon_d \approx -2^{-42.01}$.

Additionally, we verified the theoretical results of Lemma 7.3, and Equations (A.5)-(A.8) experimentally:

Computational Result 7.2 The linear approximation of Lemma 7.3 holds computationally with $\varepsilon_{L_0} = -0.015627 \approx -2^{-5.999}$. This correlation was verified using 2^{38} random samples.

Computational Result 7.3 The linear approximations of Eqs. (A.5)-(A.8) hold computationally with $\varepsilon_{L_1} = 0.083980 \approx 2^{-3.57}$, $\varepsilon_{L_2} = 0.007814 \approx 2^{-6.99}$, $\varepsilon_{L_3} = 0.006368 \approx 2^{-7.29}$, $\varepsilon_{L_4} = 0.002234 \approx 2^{-8.81}$, respectively. These correlations were verified using 2^{38} random samples.

7.4.2 Differential-Linear Attacks

Using the linear approximations of Lemma 7.3 and Lemma 7.4, the differential correlation $\varepsilon_d \approx -2^{-42.01}$ given in Eq. (7.15), and the estimated correlations from the Computational Results 7.2 and 7.3, we get $\varepsilon_d(\varepsilon_{L_0})^2 \approx 2^{-53.99}$ and $\varepsilon_d(\varepsilon_{L_0}\varepsilon_{L_1}\varepsilon_{L_2}\varepsilon_{L_3}\varepsilon_{L_4})^2 \approx 2^{-107.31}$ which gives us a distinguisher for 7 and 8 rounds of Salsa20 with complexity less than $2^{-107.98}$ and $2^{-214.62}$, respectively. As in [26], we have to repeat this attack 2 times on average because of the transition probability from $\Delta X^{(0)}$ to $\Delta X^{(1)} = \Psi$. Therefore, we have a distinguisher with data and time complexity of $2^{108.98}$ for Salsa20/7 and $2^{215.62}$ for Salsa20/8.

7.4.3 Probabilistic Neutral Bits Attack

It is straightforward to combine the new differential for 5 rounds presented in Eq. 7.15 with the technique of PNB presented in Section 3.4. More precisely, to use the differential correlation for $\Delta x_{4,7}^{(5)}$, we used the variation of PNB attack described by Beierle in [26]. Thus, consider

$$\left(x_{4,7}^{(5)} | \Delta X^{(1)} = \Psi\right).$$

To attack 7 rounds, we need to go back 2 rounds to reach the desired differential. In this case, using $\gamma = 0.3$ we found 237 PNBs, which we list in Eq. (7.16), and we obtained $\varepsilon_a = 0.027053$. From that, we get an attack with data complexity of $2^{102.61}$ and time complexity

$2^{121.63}$. As in [26], we have to repeat this attack 2 times on average because of the transition probability from $\Delta X^{(0)}$ to $\Delta X^{(1)} = \Psi$. Thus, the final attack has data complexity of $2^{103.61}$ and time complexity $2^{122.63}$.

$$\begin{aligned}
\text{PNB}_7 = \{&0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,\\
&17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,\\
&33, 34, 35, 36, 37, 38, 39, 40, 41, 45, 46, 47, 48, 49, 50, 51,\\
&52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,\\
&68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 81, 82, 83, 84, 85, 86,\\
&87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 103,\\
&104, 105, 106, 107, 108, 109, 110, 111, 112, 115, 116, 117,\\
&118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,\\
&130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,\\
&142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153,\\
&154, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,\\
&170, 171, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183,\\
&184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,\\
&196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,\\
&208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219,\\
&220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231,\\
&232, 233, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,\\
&247, 248, 249, 250, 251, 252, 253, 254, 255\}.
\end{aligned} \tag{7.16}$$

To attack 8 rounds, we need to go back 3 rounds to reach the desired differential. In this case, using $\gamma = 0.3$ we found 152 PNBs, which we list in Eq. (7.17), and we obtained $\varepsilon_a = 0.000275$. From that, (already adding 1 on the exponent), we get an attack with data complexity of $2^{115.52}$ and time complexity $2^{219.56}$.

$$\begin{aligned}
&\text{PNB}_8 = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,\\
&17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,\\
&36, 37, 38, 39, 40, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,\\
&60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,\\
&75, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,\\
&100, 103, 104, 105, 106, 107, 108, 109, 110, 115, 116, 117,\\
&118, 119, 120, 121, 122, 128, 129, 139, 140, 141, 142, 143,\\
&144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 159, 160,\\
&161, 162, 163, 164, 165, 166, 167, 168, 174, 175, 176, 177,\\
&178, 179, 180, 181, 186, 187, 192, 193, 194, 195, 199, 200,\\
&204, 205, 206, 207, 208, 213, 218, 224, 225, 231, 232, 237,\\
&238, 239, 240, 245, 249, 250, 255\}.
\end{aligned} \tag{7.17}$$

Part IV

Design of New Stream Ciphers

8 IMPROVING CHACHA AGAINST CRYPTANALYSIS

8.1 INTRODUCTION

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

This chapter is organized as follows: in Section 8.2, we provide an intensive analysis of the security of the algorithm for all combinations of rotation distances showing that it is possible to improve the security of ChaCha. In Section 8.3, we provide a security comparison of the original ChaCha, and its new proposed version.

8.2 IMPROVING CHACHA

In [10], Bernstein justify the choice of the rotation distances 16, 12, 8, 7 with the argument:

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

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

8.2.1 Testing Differential Paths

In [19], the authors presented attacks for 6 and 7 rounds of ChaCha, however, both attacks use differential paths for $r = 3$ rounds. Leveraging this fact, our first test consists in computing the best differential path for 3 rounds of ChaCha considering all single-bit input differentials and all output bits. In other words, we estimated the bias ε_d for all combinations of differentials $(\Delta^r_{p,q}|\Delta^0_{i,j})$ for each combination of rotations distances. Since each rotation have 32 values and since we have 128 input differentials and 512 output bits, we conclude that we computed $128 \times 512 \times 32^4 = 2^{36}$ different biases.

More specifically, we used Algorithm 5 to compute the highest bias for all combinations of rotations distances. Unfortunately, since we are performing an empirical estimation, we need to execute the same procedure several times for each input differential. To reduce the number of necessary computations, we used the same key, nonce, and counter for all output bits simultaneously. To test all combinations of rotation distances, we must execute Algorithm 5 2^{20} times. In addition, we defined the number of keys tested $N_k = 32$ and the number of tests per key $N_t = 1024$. Therefore, we have 2^{43} computations in total for 3 rounds of ChaCha. To achieve this amount of computation, we implemented Algorithm 5 in CUDA and executed it on a NVIDIA Quadro 4000 GPU, which required approximately 6 days of computation.

Algorithm 5 Differential Path Computation

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

The results revealed that the bias of the differential path varies significantly for each combination of rotation distances. For example, if we set $r_1 = 0$ and $r_4 = 0$ (in other words, remove these rotations), we get the biases presented in Figure 8.1, which are all equal, or very close to one. In comparison, if we set $r_1 = 16$ and $r_4 = 7$, we get much better results (see Figure 8.2) although there are still some very high biases for certain values of r_2 and r_3. Notice in Figure 8.2 that the maximum bias found for ChaCha with the original rotation distances $16, 12, 8, 7$ is not the best choice, since there are several combinations with smaller biases.

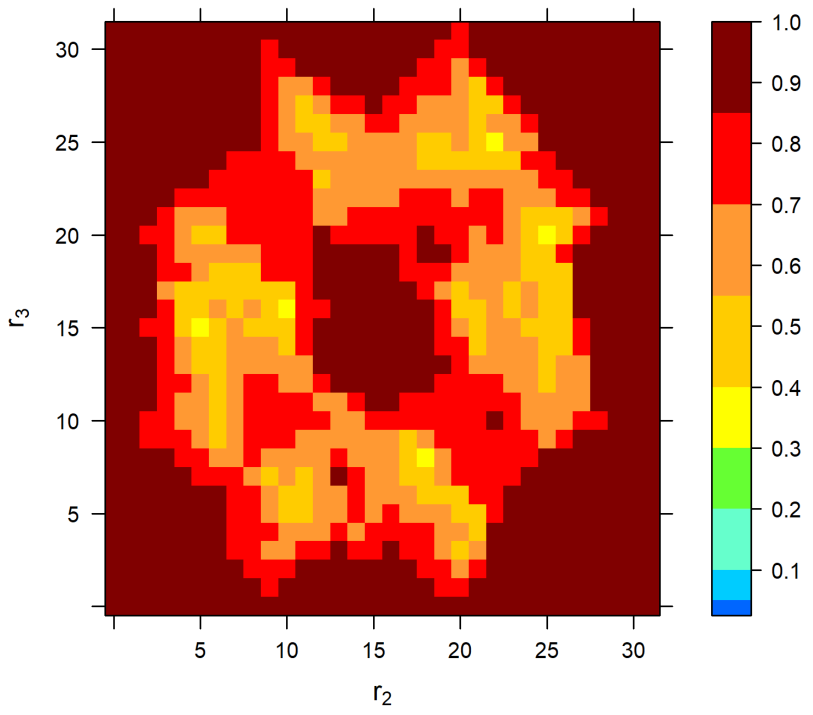


Figure 8.1 – The biases were obtained for 3 rounds of ChaCha using rotations $r_1 = r_4 = 0$ and varying all values for r_2 and r_3. The color of the figure indicates the maximum absolute bias obtained for each combination of rotations. These are very poor results since that all biases are close to 1.

8.2.2 Finding Probabilistic Neutral Bits

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

The complexity of the attack depends not only on the bias of the differential path but also

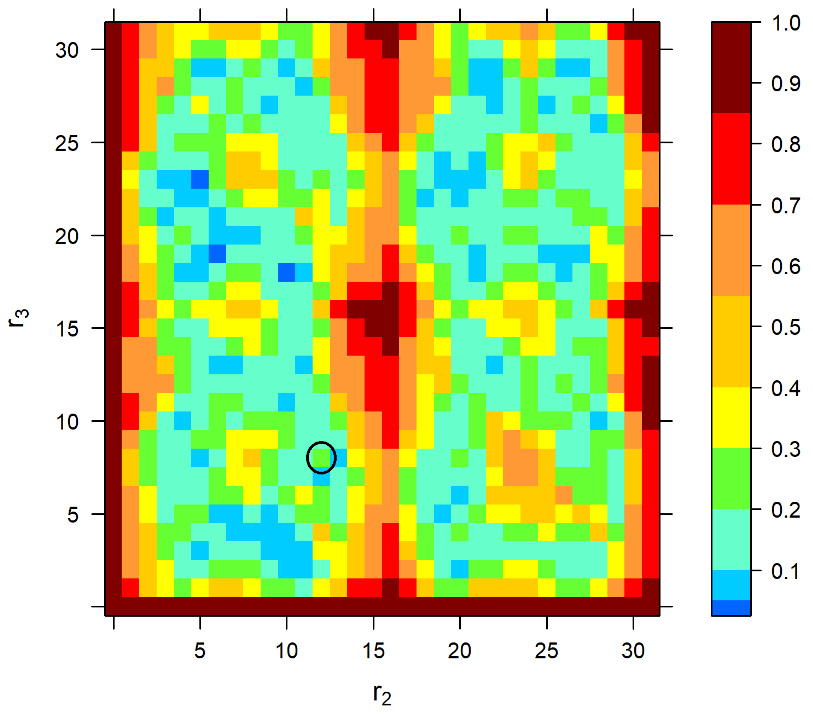


Figure 8.2 – The biases we obtained for 3 rounds of ChaCha using rotations $r_1 = 16$ and $r_4 = 7$ and varying all values for r_2 and r_3. The color of the figure indicates the maximum absolute bias obtained for each combination of rotations. The value obtained for the original ChaCha is depicted inside a black circle.

on the number of PNB. Thus, we performed another test to gather data about the behavior of PNB for each combination of rotation distances. It turns out that the computation necessary for this test increases significantly because we must test not only for each pair of input-output bits but also for each key bit individually. Fortunately, we empirically verified that, for ChaCha, the set of neutral bits are roughly the same for a particular output bit for any input bit. Thus, we can drastically reduce the necessary computation by randomly choosing the single-bit input differential. We computed the average neutrality for each output bit by performing 2^{16} iterations for each key bit, obtaining an array of 512 values. Our final statistic is defined as the maximum value in this array. We performed this test considering 7 rounds of ChaCha.

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

8.3 SECURITY COMPARISON

8.3.1 Estimating the Complexity of the PNB Attack

In [19], the authors reported an attack on 256-bit ChaCha20/6 and ChaCha20/7. For ChaCha20/6, they used the differential $(\Delta^3_{11,0}|\Delta^0_{13,13})$ with $|\varepsilon^\star_d| = 0.026$. The $\mathcal{OD}$ is observed after working 3 rounds backward from a 6-round keystream block. For the threshold $\gamma = 0.6$ they found a set of 147 non-significant key bits, with $|\varepsilon| = 0.00048$. This results in an attack in time 2^{139} and data 2^{30}. For ChaCha20/7, they used the same differential. The $\mathcal{OD}$ is observed after working 4 rounds backward from a 7-round keystream block. For the threshold $\gamma = 0.5$, they found a set of 35 non-significant key bits with $|\varepsilon| = 0.00059$. This results an attack in time 2^{248} and data 2^{27}.

We ran the attacks for ChaCha again, obtaining very similar complexity results. Using the same program, we ran the attack for ChaCha with rotation distances $19, 17, 25, 11$, showing that we get a stronger cipher. In fact, for 7 rounds, we did not find any attack with time $< 2^{256}$, see Table 8.1 for the results.

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

Table 8.1 – Best attacks obtained for ChaCha and for its modified version with rotation distances $19, 17, 25, 11$, denoted here by *ChaCha. We could not find any attacks for the modified version of ChaCha with 7 rounds.

8.3.2 Multi-bit differential

In [24], the authors provide several different attacks for ChaCha20/4, ChaCha20/5, ChaCha20/6, and ChaCha20/7. Considering the first row of Table 8.2, we note a bias $\varepsilon_d = 0.1984$ and thus $\frac{1}{\varepsilon_d^2/2} < 51$. That is, with 2^6 samples it is enough to distinguish 4-round ChaCha from a uniform random source. However, when changing the rotation distances, the best bias we get is $\varepsilon_d = -0.009179$ and thus $\frac{1}{\varepsilon_d^2/2} < 23738$. That is, with 2^{15} samples it is enough to distinguish 4-round ChaCha with rotation distances $19, 17, 25, 11$ from a uniform random source.

For ChaCha20/5, if we define $\mathcal{ID}$ at $\Delta x^0_{13,13}$ and $\mathcal{OD}$ at $\Delta x^3_{11,0}$, we obtain $\varepsilon_d = -0.0272$. By Lemma 6.2, we can extend this bias to 4 rounds, and by Lemma 10 of [24], we can further extend this bias to 5 rounds with probability $3/4$, or $\varepsilon_L = 1/2$. This gives a total differential-linear 5-th round bias of $\varepsilon = \varepsilon_d \varepsilon_L^2 = -0.0068$ thus $\frac{1}{\varepsilon^2/2} < 43253$. That is, with 2^{16} samples it is enough to distinguish 5-round ChaCha from a uniform random source. However, changing the rotation distances and if we define $\mathcal{ID}$ at $\Delta^0_{14,12}$ and $\mathcal{OD}$ at $\Delta^3_{8,0}$, we obtain $\varepsilon_d = -0.000915$, and from Lemmas 6.2 and 10 of [24], we get a total differential-linear 5-th round bias of $\varepsilon = \varepsilon_d \varepsilon_L^2 = -0.00022875$ thus $\frac{1}{\varepsilon^2/2} < 38221506$. That is, with 2^{26} samples it is enough to distinguish 5-round ChaCha with rotation distances $19, 17, 25, 11$ from a uniform random source.

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

Table 8.2 – The best multi-bit differentials for ChaCha and for its modified version with rotation distances $19, 17, 25, 11$, denoted here by *ChaCha. Notice that we can reduce the bias significantly.

Extending the linear approximation for 3 rounds comes at a cost. As discussed in [24], for 6 rounds, the linear bias after expanding any equation from Lemma 10 of [24] is $\varepsilon_L = 1/(2\cdot 1+3\cdot 4+5\cdot 1+3\cdot 2+2\cdot 1) = 1/2^{26}$. To use this extension, we searched for the input differential that maximizes the differential bias for $\Delta x^{(3)}_{8,0}, \Delta x^{(3)}_{9,0}, \Delta x^{(3)}_{10,0}$ or $\Delta x^{(3)}_{11,0}$, which leads to the differential pair $(\Delta x^{(3)}_{9,0} | \Delta x^{(0)}_{15,12})$ with $\varepsilon_d = 0.000792$. This leads to a 6-round bias of $\varepsilon_L^2 \varepsilon_d \approx \frac{1}{2^{62.3}}$ and a distinguisher with complexity of 2^{125}.

For a key recovery attack against 6 rounds of ChaCha, we must use PNB. The best attack we obtained when considering the proposed rotation distances uses 5 rounds forward and then 1 round backward. For this the $\mathcal{ID}$ is $\Delta^{(0)}_{12,12}$ and the $\mathcal{OD}$ in the third round is $\Delta^{(3)}_{10,0}$, thus, using the third equation of Lemma 10 of [24], we can mount an attack. We got 157 PNBs using $\gamma = 0.6$ from which we estimated $\varepsilon = -0.000024$, $\varepsilon^\star = -0.000023$ leading to an attack with data complexity $2^{38.7}$ and time complexity $2^{137.7}$. For 7 rounds of ChaCha with the proposed rotation distances, we did not find any significant attacks. Also, we could not use the equations from Lemma 6.5 since we could not find any significant bias for a double output differential bias. Table 8.3 summarizes our findings[1].

[1]Important: this chapter was compiled from the results of the paper "Improving the Security of ChaCha against Differential-Linear Cryptanalysis" written in the beginning of 2020. For this reason, the results of the cryptanalysis of the proposed version of ChaCha do not considered some results and techniques that appeared later in [26] and [27]

Algorithm	Rounds	Data	Time	Type	Reference
ChaCha	4	2^{6}	2^{6}	Distinguisher	[24]
	5	2^{16}	2^{16}	Distinguisher	[24]
	6	2^{51}	2^{51}	Distinguisher	This work
	6	2^{58}	$2^{77.4}$	Key recovery	[26]
	7	2^{224}	2^{224}	Distinguisher	This work
	7	$2^{48.8}$	$2^{230.86}$	Key recovery	[26]
*ChaCha	4	2^{15}	2^{15}	Distinguisher	This work
	5	2^{26}	2^{26}	Distinguisher	This work
	6	2^{125}	2^{125}	Distinguisher	This work
	6	$2^{38.7}$	$2^{137.7}$	Key recovery	This work
	7	–	–	–	–

Table 8.3 – Attacks obtained considering the techniques presented in Section 6.2. Notice that the complexity of the attacks for ChaCha with rotation distances $19, 17, 25, 11$, denoted here by *ChaCha, are higher, thus, the proposed modification is more secure against these attacks.

9 A NEW ALGORITHM: FORRÓ

In this chapter, we conclude the thesis with the proposition of a new stream cipher named Forró. After improving attacks of ARX algorithms, in particular, of the Salsa and ChaCha family, we decided to research for new ways to improve the security of these algorithms against known attacks. This chapter is divided into the following way: in Section 9.1, we present the algorithm Forró. In Section 9.2, we apply all know techniques to evaluate security of Forró. Finally, in Section 9.3, we compare the performance of Forró with ChaCha and Salsa.

9.1 FORRÓ

Although they have very similar structure, the literature suggests that ChaCha is safer than Salsa. Therefore, a natural question that arises is if we can do better with fewer operations, it turns out the answers is yes. To do that, first, we will introduce a new concept that we call *Pollination*. Then, we discuss how to change ChaCha to achieve it, leading to a new algorithm.

9.1.1 Pollination

In this section, we propose a new technique that we call *Pollination*. We chose this name as an analogy to the real Pollination in nature: when a bee collects nectar and pollen from the flower of a plant sticks to the hairs of her body. When she visits the next flower, some of this pollen is rubbed off onto the stigma, making fertilization possible. Here, our idea is to use the element that is likely to maximize confusion and diffusion (we call this best element *pollen*) to bring non-linearity and confusion to other elements in the state matrix.

Actually, one of the reasons behind the improved diffusion of ChaCha when compared to Salsa is, in fact, pollination. Since the QRF function updates one element after the other, using the previously updated element as input, then it is a natural consequence that the element updated last ($x_b^{(r)}$) has higher diffusion. In ChaCha, the pattern of application of the QRF actually makes that the elements in the second row (which are the parameter $x_b^{(r)}$ for each QRF application), is used to update the first element in the next round. Salsa does not have such a property, hence the improved diffusion of ChaCha.

ChaCha achieves pollination from one round to another, however, it fails to do so within each round because the QRF is applied independently in each column or diagonal. Thus, it is possible to have more diffusion in fewer operations if we create a chain of pollination from one application of the QRF to the other. It can be argued that we lose parallelism in each round, however, as we show later, the improved diffusion allow, the same security in fewer rounds. Also, in Section 9.3, we show that the parallelism is not actually lost, since we can apply it in another way.

9.1.2 Forró's Round Function

To create pollination from one round to the others, we propose to include an extra parameter into the QRF. Nevertheless, we want to maintain (or to decrease) the number of arithmetic operations to achieve competitive performance. Notice that each rotation in Eq. (2.6) actually make the same element be updated twice in a row thus, we could update more elements if we had fewer rotations.

Actually, in [8], Bernstein asked the question of whether there should be fewer rotations in the QRF, because rotations account for about 1/3 of the integer operations in Salsa (and also in ChaCha), he wrote:

> "If rotations are simulated by shift-shift-xor (as they are on the UltraSPARC and with XMM instructions), then they account for about 1/2 of the integer operations in Salsa20. Replacing some of the rotations with a comparable number of additions might achieve comparable diffusion in less time."

With those ideas in mind, we define $QR_{forro}\left(x_a^{(r-1)}, x_b^{(r-1)}, x_c^{(r-1)}, x_d^{(r-1)}, x_e^{(r-1)}\right)$ as the following set of operations

$$\begin{array}{lcl}
x_d'^{(m-1)} & = & x_d^{(m-1)} + x_e^{(m-1)} \\
x_c'^{(m-1)} & = & x_c^{(m-1)} \oplus x_d'^{(m-1)} \\
x_b'^{(m-1)} & = & \left(x_b^{(m-1)} + x_c'^{(m-1)}\right) \lll r_1 \\
x_a'^{(m-1)} & = & x_a^{(m-1)} + x_b'^{(m-1)} \\
x_e^{(m)} & = & x_e^{(m-1)} \oplus x_a'^{(m-1)} \\
x_d^{(m)} & = & \left(x_d'^{(m-1)} + x_e^{(m)}\right) \lll r_2 \\
x_c^{(m)} & = & x_c'^{(m-1)} + x_d^{(m)} \\
x_b^{(m)} & = & x_b'^{(m-1)} \oplus x_c^{(m)} \\
x_a^{(m)} & = & \left(x_a'^{(m-1)} + x_b^{(m)}\right) \lll r_3
\end{array} \tag{9.1}$$

where $r_1 = 10, r_2 = 27$ and $r_3 = 8$.

Notice that QR_{forro} has a total of 12 operations, just like QR_{ChaCha}, but fewer rotations. Also, notice that QR_{forro} is asymmetric in the sense that of all elements there is one, namely $x_e^{(r)}$ that is updated less frequently than the others. However, this behavior is actually acceptable since $x_e^{(r)}$ is the element used for pollination. Thus, its job is to provide non-linearity and confusion and not to gain more necessarily. In addition, if this is not the first use of QR_{forro} in the algorithm, this element was updated recently in the last iteration of the function QR_{forro}. Finally, notice that as the element $x_a^{(r)}$ is the last to be updated, then it likely has the more complex Boolean functions in comparison to $x_b^{(r)}, x_c^{(r)}, x_d^{(r)}$ and $x_e^{(r)}$, therefore $x_a^{(r)}$ will become the pollen for the next application of QR_{forro}. Then, in an odd round, when $r \in \{1, 3, 5, 7, ...\}$, $X^{(r)}$ is defined from $X^{(r-1)}$ in the following manner

$$\begin{aligned}
\left(x_{0*}^{(r-1)}, x_4^{(r)}, x_8^{(r)}, x_{12}^{(r)}, x_{3*}^{(r-1)}\right) &= QR_{forro}\left(x_0^{(r-1)}, x_4^{(r-1)}, x_8^{(r-1)}, x_{12}^{(r-1)}, x_3^{(r-1)}\right)\\
\left(x_{1*}^{(r-1)}, x_5^{(r)}, x_9^{(r)}, x_{13}^{(r)}, x_0^{(r)}\right) &= QR_{forro}\left(x_1^{(r-1)}, x_5^{(r-1)}, x_9^{(r-1)}, x_{13}^{(r-1)}, x_{0*}^{(r-1)}\right)\\
\left(x_{2*}^{(r-1)}, x_6^{(r)}, x_{10}^{(r)}, x_{14}^{(r)}, x_1^{(r)}\right) &= QR_{forro}\left(x_2^{(r-1)}, x_6^{(r-1)}, x_{10}^{(r-1)}, x_{14}^{(r-1)}, x_{1*}^{(r-1)}\right)\\
\left(x_3^{(r)}, x_7^{(r)}, x_{11}^{(r)}, x_{15}^{(r)}, x_2^{(r)}\right) &= QR_{forro}\left(x_{3*}^{(r-1)}, x_7^{(r-1)}, x_{11}^{(r-1)}, x_{15}^{(r-1)}, x_{2*}^{(r-1)}\right)
\end{aligned} \tag{9.2}$$

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

$$\begin{aligned}
\left(x_{0*}^{(r-1)}, x_5^{(r)}, x_{10}^{(r)}, x_{15}^{(r)}, x_{3*}^{(r-1)}\right) &= QR_{forro}\left(x_0^{(r-1)}, x_5^{(r-1)}, x_{10}^{(r-1)}, x_{15}^{(r-1)}, x_3^{(r-1)}\right)\\
\left(x_{1*}^{(r-1)}, x_6^{(r)}, x_{11}^{(r)}, x_{12}^{(r)}, x_0^{(r)}\right) &= QR_{forro}\left(x_1^{(r-1)}, x_6^{(r-1)}, x_{11}^{(r-1)}, x_{12}^{(r-1)}, x_{0*}^{(r-1)}\right)\\
\left(x_{2*}^{(r-1)}, x_7^{(r)}, x_8^{(r)}, x_{13}^{(r)}, x_1^{(r)}\right) &= QR_{forro}\left(x_2^{(r-1)}, x_7^{(r-1)}, x_8^{(r-1)}, x_{13}^{(r-1)}, x_{1*}^{(r-1)}\right)\\
\left(x_3^{(r)}, x_4^{(r)}, x_9^{(r)}, x_{14}^{(r)}, x_2^{(r)}\right) &= QR_{forro}\left(x_{3*}^{(r-1)}, x_4^{(r-1)}, x_9^{(r-1)}, x_{14}^{(r-1)}, x_{2*}^{(r-1)}\right).
\end{aligned} \tag{9.3}$$

9.1.3 Initialization

To initialize the state matrix we have 16 integers available, being 8 key words, 2 nonce words, 2 counter words and 4 constants. All positions in the state matrix are different in terms of diffusion and whether it is used sooner or later. Forró's initialization matrix is defined by

$$X^{(0)} = \begin{pmatrix} x_0^{(0)} & x_1^{(0)} & x_2^{(0)} & x_3^{(0)}\\ x_4^{(0)} & x_5^{(0)} & x_6^{(0)} & x_7^{(0)}\\ x_8^{(0)} & x_9^{(0)} & x_{10}^{(0)} & x_{11}^{(0)}\\ x_{12}^{(0)} & x_{13}^{(0)} & x_{14}^{(0)} & x_{15}^{(0)} \end{pmatrix} = \begin{pmatrix} k_0 & k_1 & k_2 & k_3\\ t_0 & t_1 & c_0 & c_1\\ k_4 & k_5 & k_6 & k_7\\ v_0 & v_1 & c_2 & c_3 \end{pmatrix}. \tag{9.4}$$

When comparing Eqs. (2.5) and (9.4), one can notice that Forró's initialization is different than ChaCha's. In differential cryptanalysis usually the attacker is allowed to chose arbitrary values to t_0, t_1, v_0 and v_1, thus it is a good idea to update this values as soon as possible allowing the differential to be propagated faster decreasing the probability of a differential characteristic. Thus, we defined the initialization in such a way that t_0, t_1, v_0 and v_1 are used in the first two columns.

9.1.4 Rotations

The rotation distances for Forró are set as $r_1 = 10, r_2 = 27$ and $r_3 = 8$. Most authors of ARX algorithms in the literature do not justify the choice of the rotation distances with a numerical argument. It is generally argued that it is difficult to find bad rotation distances for ARX. Therefore, authors tend to choose aligned rotation distances (multiple of 8) since these are much faster than unaligned rotation counts on many non-64-bit architectures. For example, many 8-bit microcontrollers have only 1-bit shifts of bytes, so rotation by (e.g.) 3 bits is particularly expensive. Even 64-bit systems can benefit from alignment, for example, when a sequence of shift-shift-xor can be replaced by SSSE3's `pshufb` byte-shuffling instruction [71].

On the other hand, it may be possible to improve the security of the algorithm by carefully studying the behavior of the cipher when each combination of rotation distances are evaluated. This approach could allow for a reduced number of rounds to achieve the desired security. Hence, this approach could also improve performance. For example, in Chapter 8 (also see [72]) we showed that changing the rotation distances of ChaCha to $(19, 17, 25, 11)$ improved the resistance of ChaCha against known attacks.

Here, the rotation distances were defined following a similar approach as proposed in Chapter 8, with some adaptations. First, we define $\mathcal{R}$ as the set of all combinations of rotation distances (note that $|\mathcal{R}| = 32^3$). Next, we define Algorithm 6, which returns the maximum observed differential correlation among all single bit differentials $(\mathcal{ID}, \mathcal{OD})$ for a given combination of rotation distances $\mathbf{r} = (r_1, r_2, r_3) \in \mathcal{R}$ when considering N random trials. Then, to define the optimal rotation distances we executed the following steps:

Algorithm 6 Returns the maximum observed differential correlation for all possible single bit differentials.

```
1: INPUT: rotation distances $(r_1, r_2, r_3)$, the number of trials $N$.
2: Setup Forró with rotation distances $(r_1, r_2, r_3)$.
3: for each single bit input difference $\mathcal{ID}$ do
4:     for $i \in \{1, 2, ..., N\}$ do
5:         Generate random key $k$, nonce $v$, and counter $t$.
6:         Initialize Forró's state matrix $X$.
7:         Execute 2 rounds of Forró from $X$, obtaining $Y$.
8:         Compute $X' = \mathcal{ID} \oplus X$.
9:         Execute 2 rounds of Forró from $X'$, obtaining $Y'$.
10:        Compute $\mathcal{OD} = Y \oplus Y'$.
11:        Update the differential correlation $\delta_{\mathcal{ID},j}$ for each bit of $\mathcal{OD}$, where $j \in \{0, 1, ..., 512\}$.
12: return $\max(|\delta_{\mathcal{ID},j}|)$
```

1. Execute Algorithm 6 for all $\mathbf{r}_i \in \mathcal{R}$, obtaining a list $L = \{\delta_{\mathbf{r}_i}\}$.

2. Compute $\delta_{\min} = \min(L)$.

3. For each $\delta_{\mathbf{r}_i} \in L$, test the hypothesis $H_i : \delta_{\mathbf{r}_i} = \delta_{\min}$. More precisely, we used the standard statistical test to compare two proportions by converting the correlation to a probability $p_{\mathbf{r}_i} = (\delta_{\mathbf{r}_i} + 1)/2$. In addition, since we are dealing with multiple hypotheses tests, we used the Family-Wise Error Rate (FWER) technique to guard against type-I errors.

4. Discard all rotations distances $\mathbf{r}_i \in \mathcal{R}$ that lead to the hypothesis H_i being rejected. Thus, we are left with a subset of rotation distances $\mathcal{R}^* \subset \mathcal{R}$.

5. For each $\mathbf{r}_j \in \mathcal{R}^*$, compute the average neutrality measure $\bar{\gamma}_{\mathbf{r}_j}$ using Algorithm 1 of [19]. In this case, we considered a encryption with 5 rounds of Forró and 3 rounds executed backwards.

6. For each $\mathbf{r}_j \in \mathcal{R}^*$, define the metric $\mu_{\mathbf{r}_j} = \delta_{\mathbf{r}_j} \times \bar{\gamma}_{\mathbf{r}_j}$.

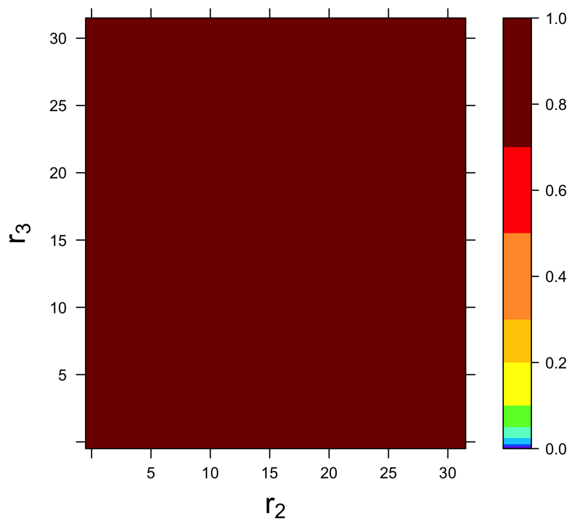
7. Define the rotation distances for Forró as $\arg\min_{\mathbf{r}_j}\{\mu_{\mathbf{r}_j}\}$.

We executed these steps using a cluster of 24 *NVIDIA GPUs RTX 2080ti*. This setup allowed us to run Algorithm 6 with $N = 24 \times 2^{20}$, for all $\mathbf{r} \in \mathcal{R}$, in two days of computation. This led to $\delta_{\min} = 0.003056$ when $\mathbf{r}_i = (24, 28, 13)$. The results are very interesting as some patterns can be observed, as illustrated in Figures 9.1-9.8.

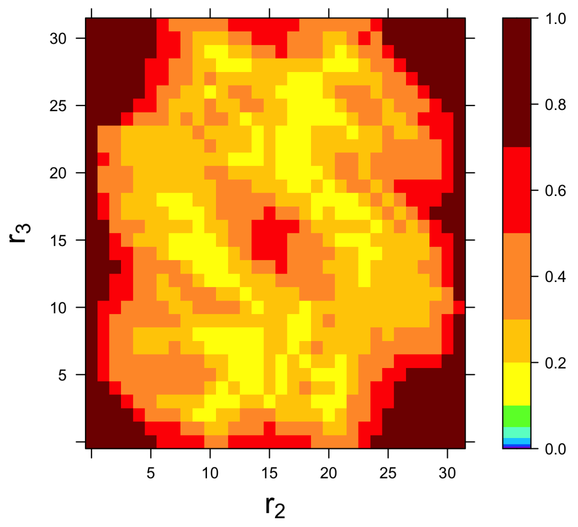
Then, after hypothesis testing, we were left with 51 rotation distances, i.e., $|\mathcal{R}^*| = 51$ (these rotations distances are marked in purple in Figures 9.1-9.8). Note that this step means that from all 32^3 possible rotation distances, this group of 51 were statistically better in terms of security against differential cryptanalysis. From these, we defined Forró's rotation distances as $(10, 27, 8) = \arg\min_{\mathbf{r}_j}\{\mu_{\mathbf{r}_j}\}$, where $\delta_{(10,27,8)} = 0.003773$ and $\bar{\gamma}_{(10,27,8)} = 0.058866$. Luckily enough, we also have one aligned rotation distance ($r_3 = 8$), which will help performance further.

9.1.5 Constants

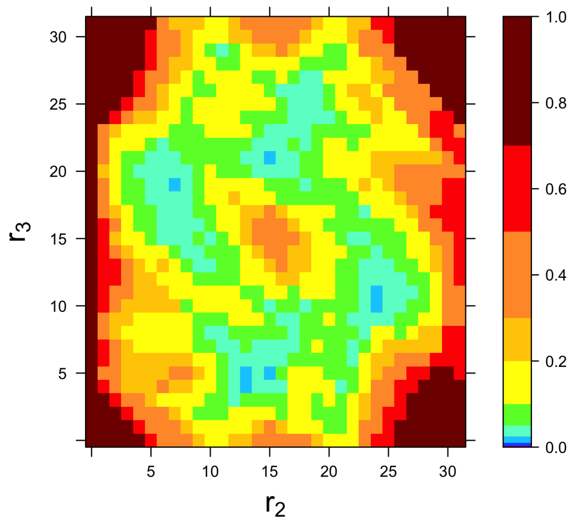
Since the choice of the constants do not impact security or performance, we decided to go through a cultural route: the constants correspond to the ASCII string "*voltadaasabranca*", little-endian encoded. "*A volta da asa branca*" is the name of a song of the brazilian singer Luiz Gonzaga. It is a continuation of the song "*asa branca*", one of the greatest classics of Brazilian music, composed more than 70 years ago. In "*asa branca*", Luiz Gonzaga and Humberto Teixeira tell us the story of a man who lost everything due to the drought in the Brazilian northeast region and had to leave his home in search of better living conditions. In "*a volta da asa branca*", he returns to his home and is reunited with his love with whom he intends to marry.



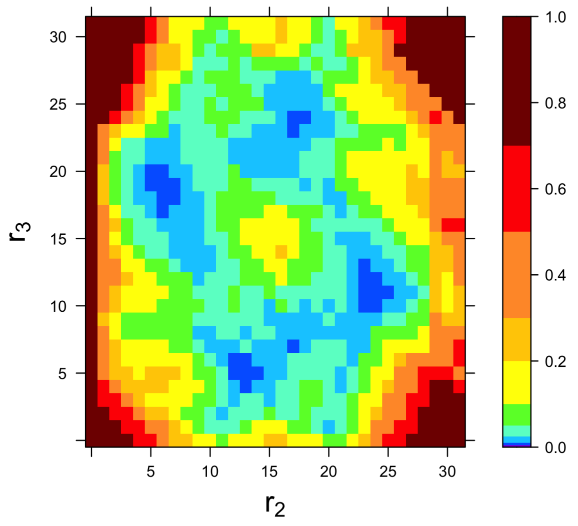
(a) Correlations for $r_1 = 0$.



(b) Correlations for $r_1 = 1$.

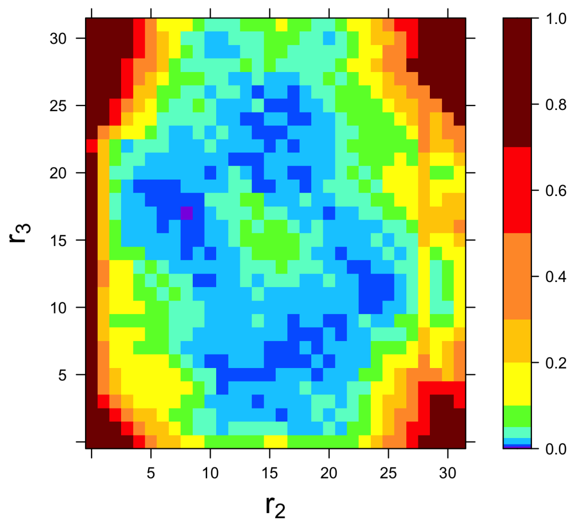


(c) Correlations for $r_1 = 2$.

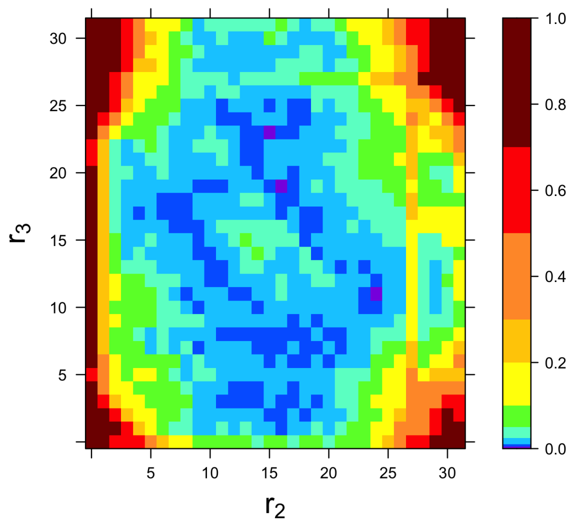


(d) Correlations for $r_1 = 3$.

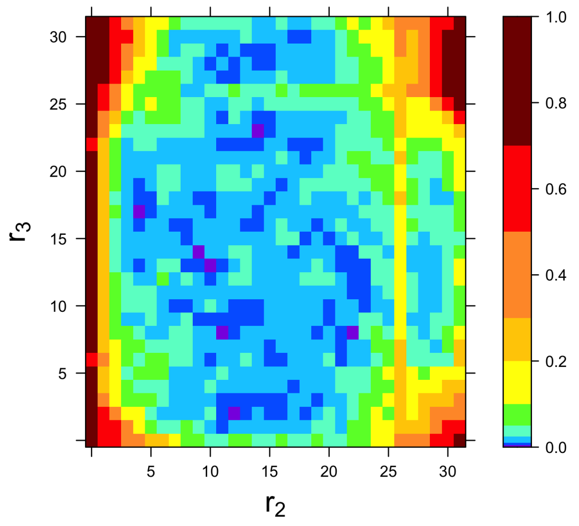
Figure 9.1 – The result of Algorithm 6 for each combination of rotation distances.



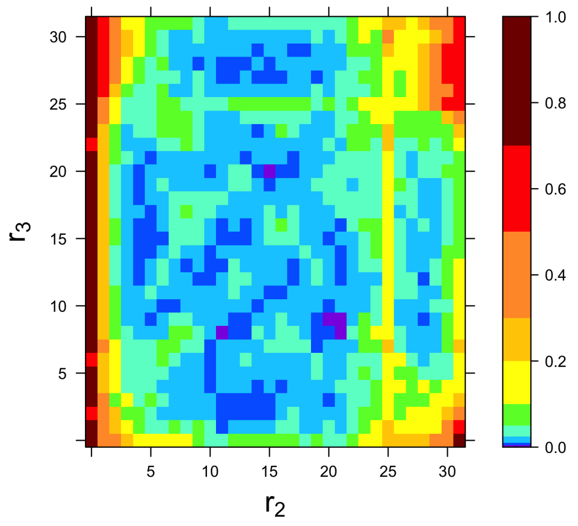
(a) Correlations for $r_1 = 4$.



(b) Correlations for $r_1 = 5$.

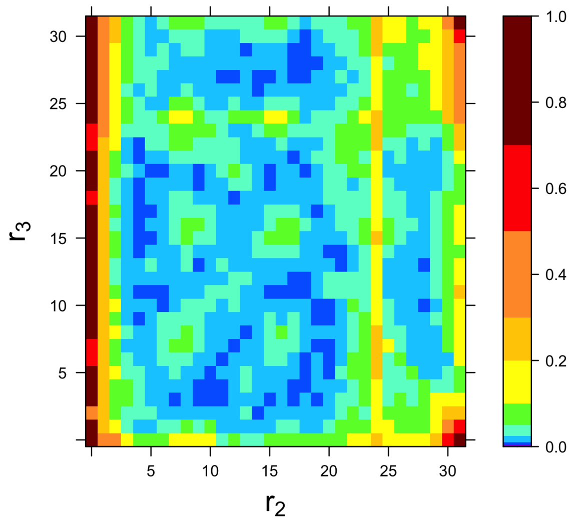


(c) Correlations for $r_1 = 6$.

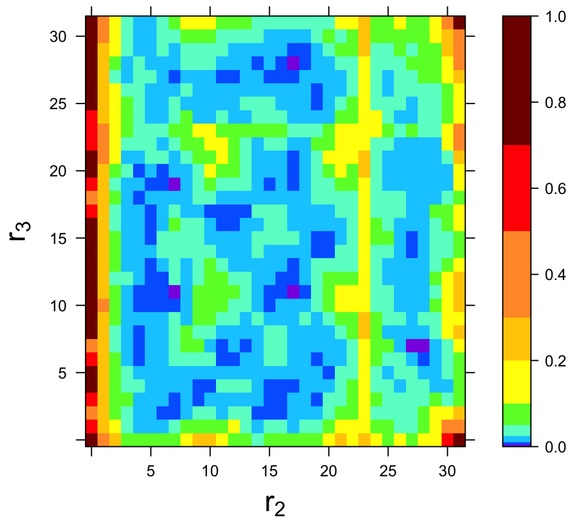


(d) Correlations for $r_1 = 7$.

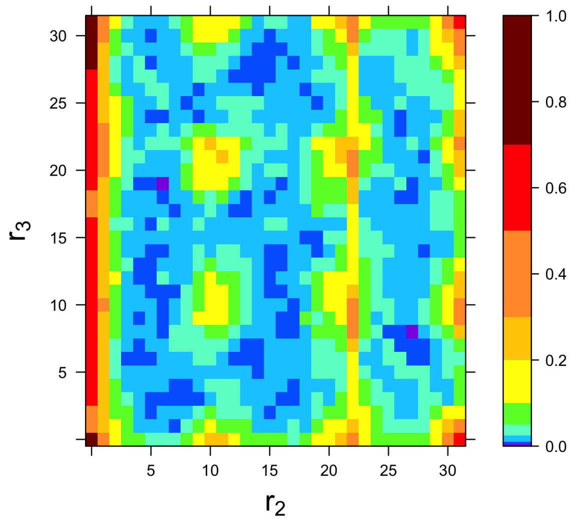
Figure 9.2 – The result of Algorithm 6 for each combination of rotation distances.



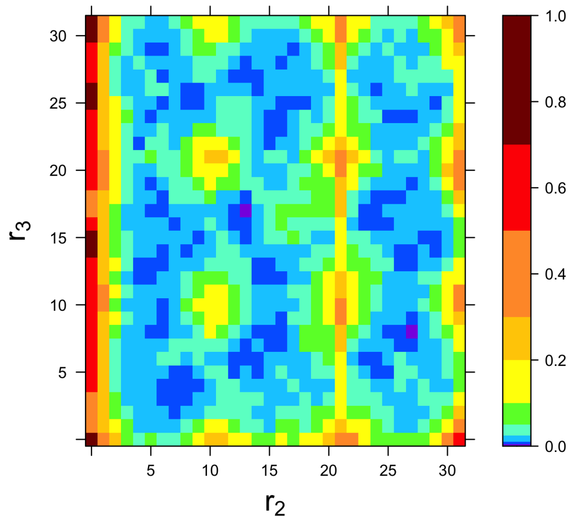
(a) Correlations for $r_1 = 8$.



(b) Correlations for $r_1 = 9$.

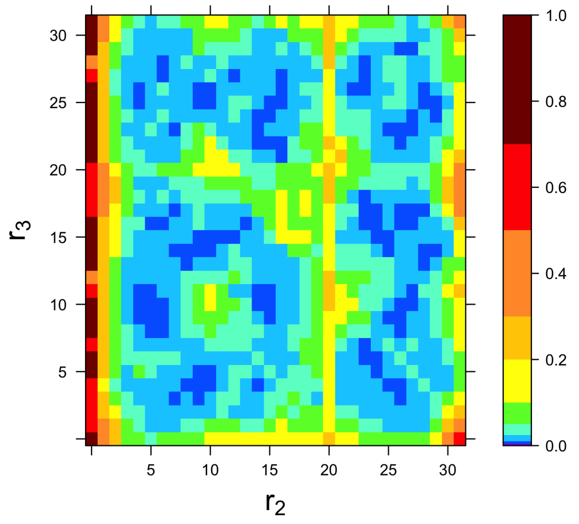


(c) Correlations for $r_1 = 10$.

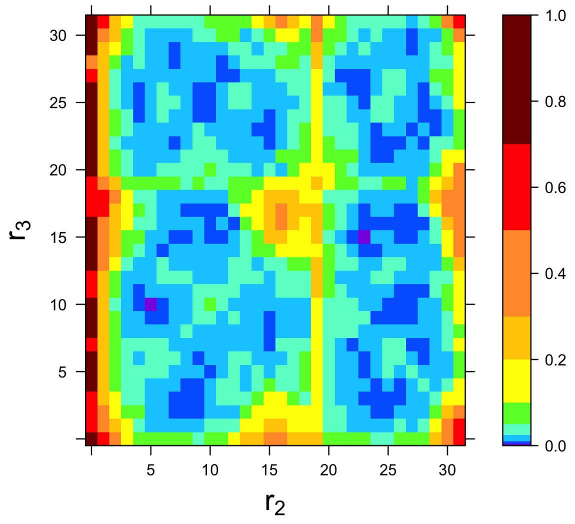


(d) Correlations for $r_1 = 11$.

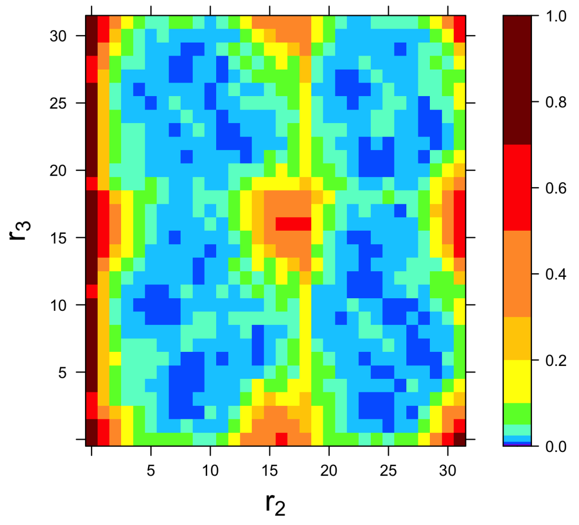
Figure 9.3 – The result of Algorithm 6 for each combination of rotation distances.



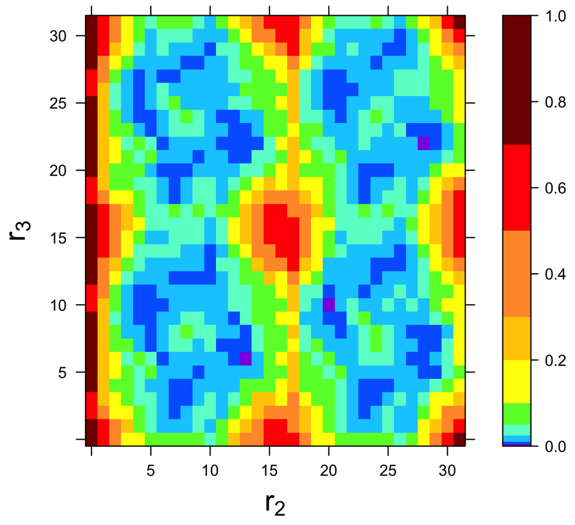
(a) Correlations for $r_1 = 12$.



(b) Correlations for $r_1 = 13$.

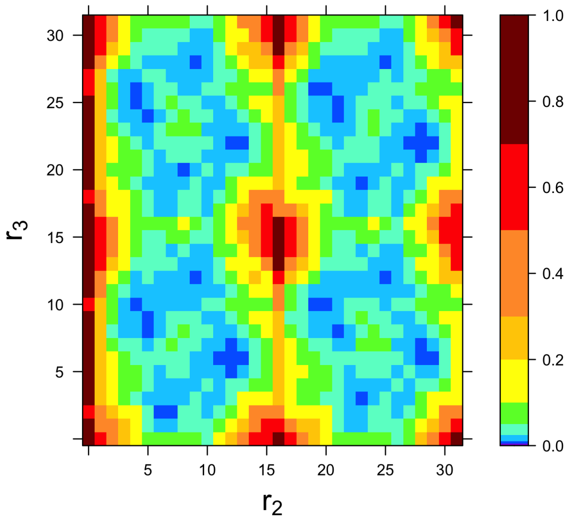


(c) Correlations for $r_1 = 14$.

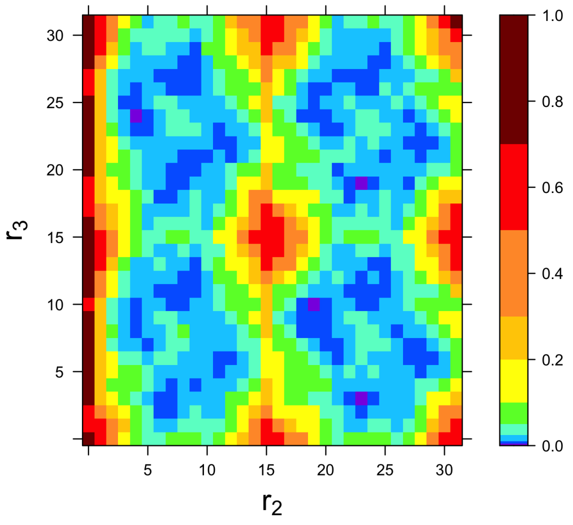


(d) Correlations for $r_1 = 15$.

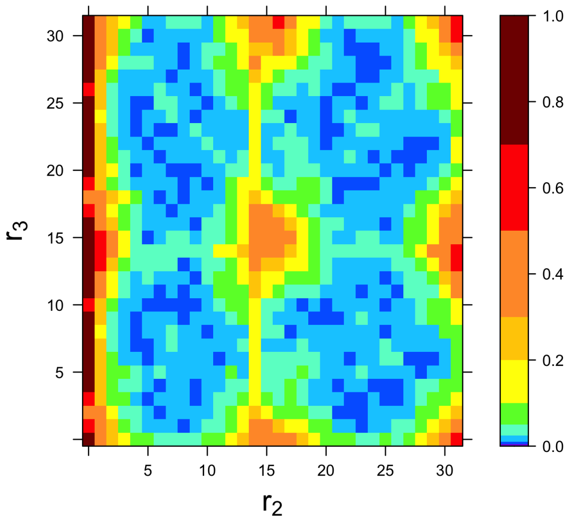
Figure 9.4 – The result of Algorithm 6 for each combination of rotation distances.



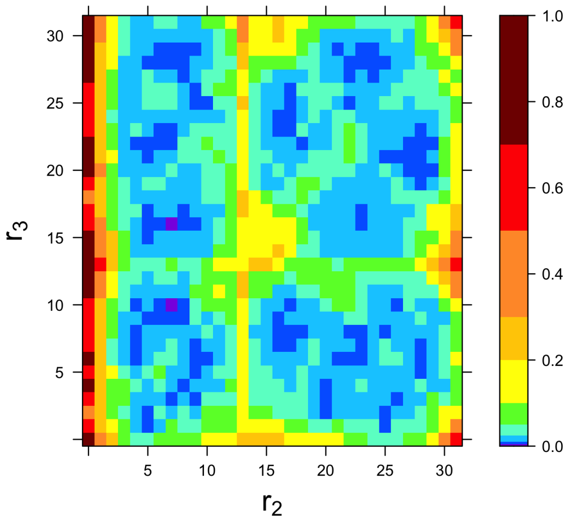
(a) Correlations for $r_1 = 16$.



(b) Correlations for $r_1 = 17$.

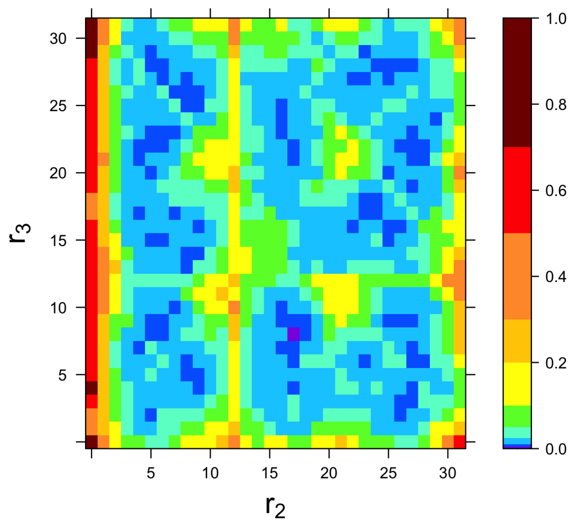


(c) Correlations for $r_1 = 18$.

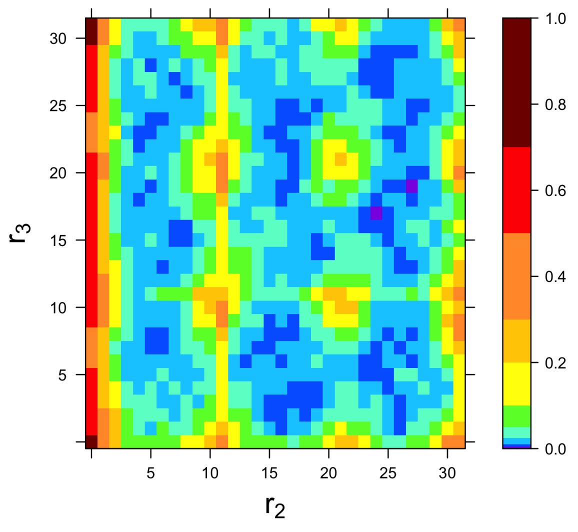


(d) Correlations for $r_1 = 19$.

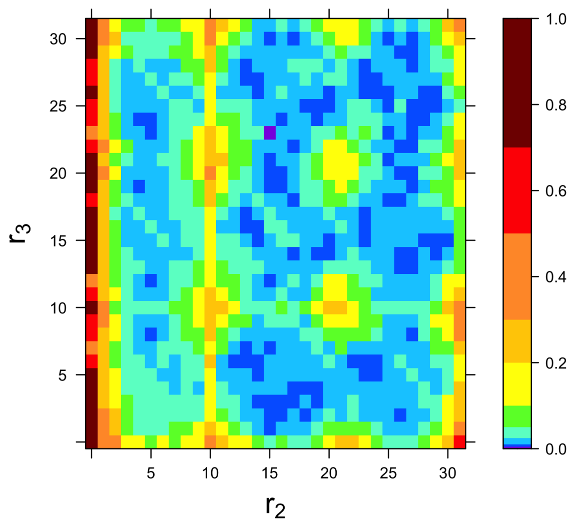
Figure 9.5 – The result of Algorithm 6 for each combination of rotation distances.



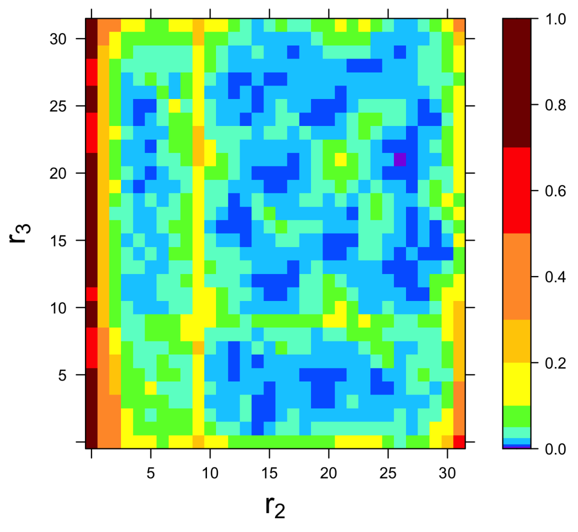
(a) Correlations for $r_1 = 20$.



(b) Correlations for $r_1 = 21$.

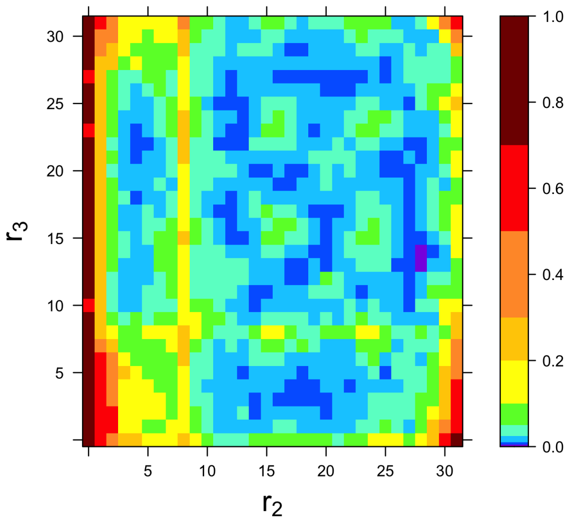


(c) Correlations for $r_1 = 22$.

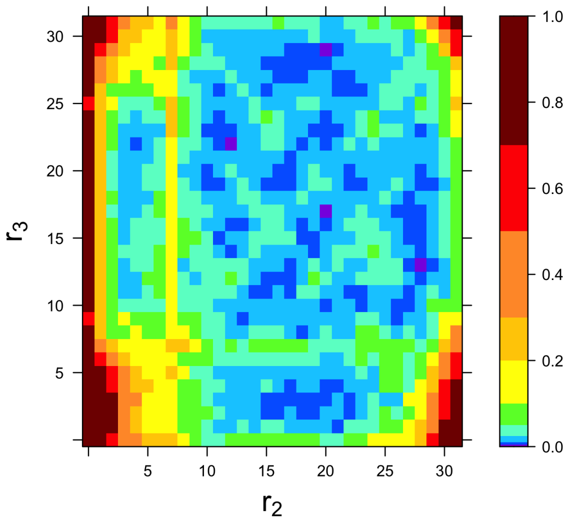


(d) Correlations for $r_1 = 23$.

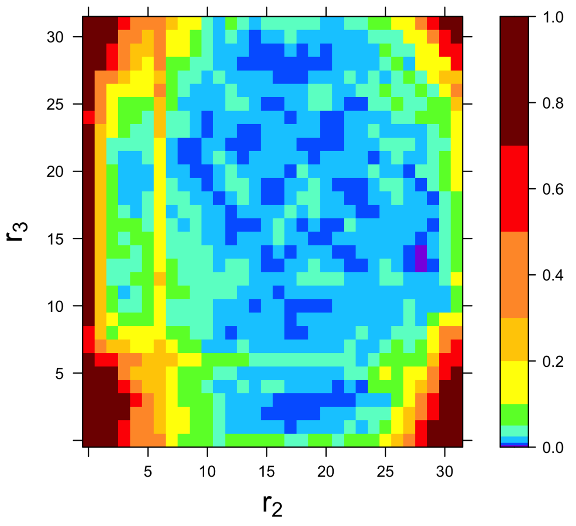
Figure 9.6 – The result of Algorithm 6 for each combination of rotation distances.



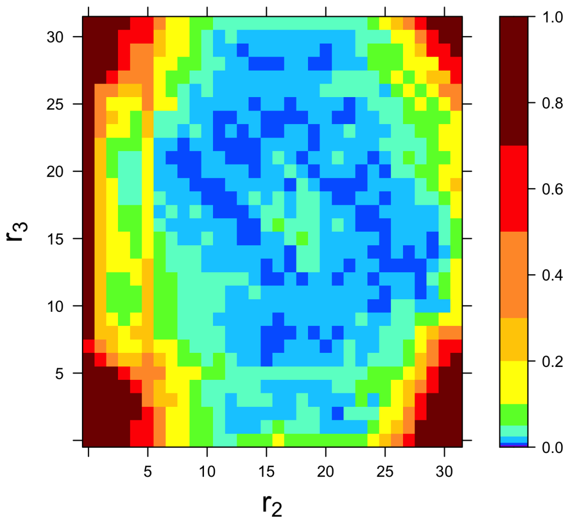
(a) Correlations for $r_1 = 24$.



(b) Correlations for $r_1 = 25$.

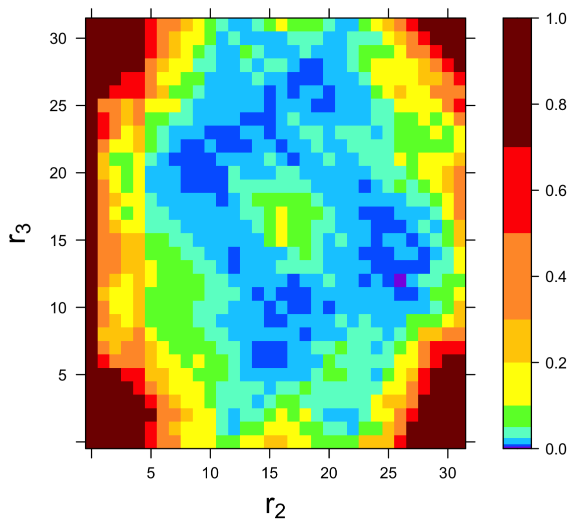


(c) Correlations for $r_1 = 26$.

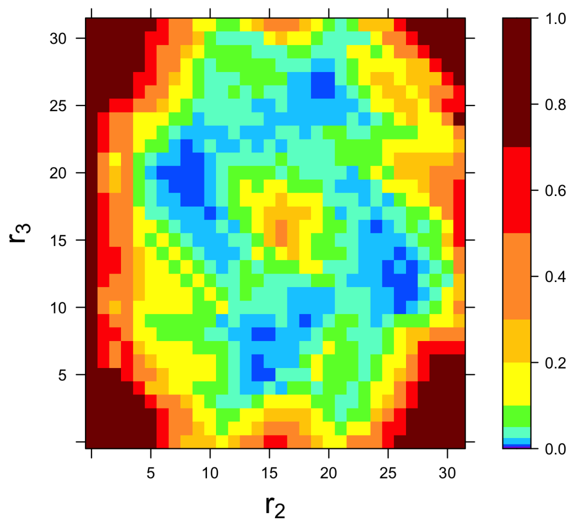


(d) Correlations for $r_1 = 27$.

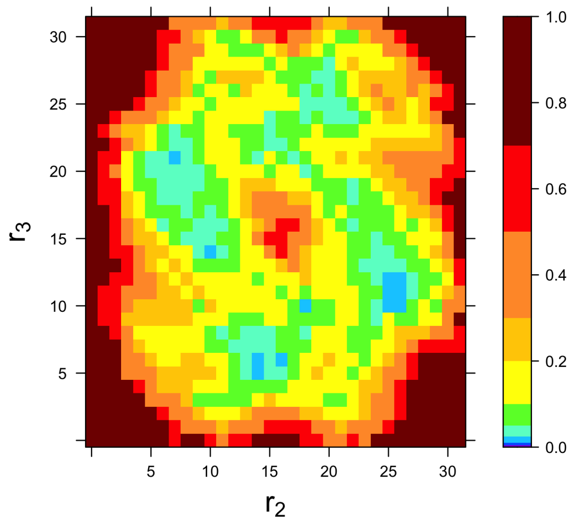
Figure 9.7 – The result of Algorithm 6 for each combination of rotation distances.



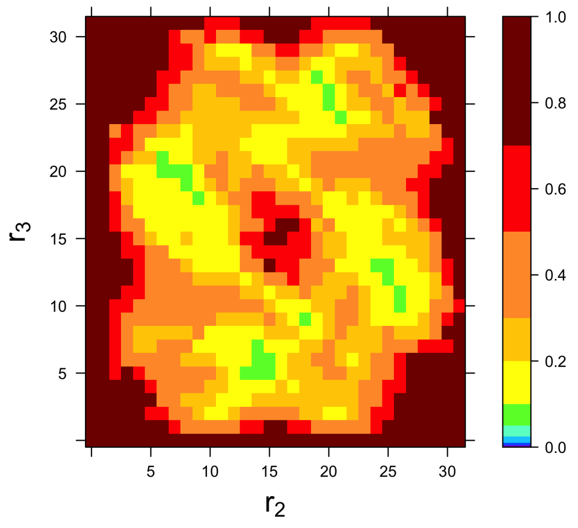
(a) Correlations for $r_1 = 28$.



(b) Correlations for $r_1 = 29$.



(c) Correlations for $r_1 = 30$.



(d) Correlations for $r_1 = 31$.

Figure 9.8 – The result of Algorithm 6 for each combination of rotation distances.

9.2 SECURITY

Here, we analyze the security of Forró by first replicating and checking the results of the attack of Aumasson [19] and of Choudhuri [24] for both and Salsa and ChaCha and then applying the technique against Forró. We chose these attacks since they are the most important works on the cryptanalysis of Salsa and ChaCha to this day.

In this section, we define the subround of Forró as a single application of the QR_{forro}. Therefore, we can say that Forró has 12 rounds or 48 subrounds. As we will see, the subround view of Forró is better to understand its cryptanalysis. Hence, we adapt the notation of the state $X^{(m)}$ as $X^{[s]}$, where m denote the number of complete rounds and s the number of subrounds. For example, we have that $X^{(4)} = X^{[16]}$.

9.2.1 Linear approximations for Forró

In this section, we use the techniques introduced in Chapter 6 to construct linear relations between distinct subrounds of Forró, and use them to construct multi-bit differentials that can be use to construct distinguishers to round reduced Forró. We can write the QRF equations of Forró (Eq. (9.1)) as

$$\begin{aligned}
x'^{[s-1]}_{d,i} &= x^{[s-1]}_{d,i} \oplus x^{[s-1]}_{e,i} \oplus \Theta_i(x^{[s-1]}_{d}, x^{[s-1]}_{e}) \\
x'^{[s-1]}_{c,i} &= x^{[s-1]}_{c,i} \oplus x'^{[s-1]}_{d,i} \\
x'^{[s-1]}_{b,i+10} &= x^{[s-1]}_{b,i} \oplus x'^{[s-1]}_{c,i} \oplus \Theta_i(x^{[s-1]}_{b}, x'^{[s-1]}_{c}) \\
x'^{[s-1]}_{a,i} &= x^{[s-1]}_{a,i} \oplus x'^{[s-1]}_{b,i} \oplus \Theta_i(x^{[s-1]}_{a}, x'^{[s-1]}_{b}) \\
x^{[s]}_{e,i} &= x^{[s-1]}_{e,i} \oplus x'^{[s-1]}_{a,i} \\
x^{[s]}_{d,i+27} &= x'^{[s-1]}_{d,i} \oplus x^{[s]}_{e,i} \oplus \Theta_i(x'^{[s-1]}_{d}, x^{[s]}_{e}) \\
x^{[s]}_{c,i} &= x'^{[s-1]}_{c,i} \oplus x^{[s]}_{d,i} \oplus \Theta_i(x'^{[s-1]}_{c}, x^{[s]}_{d}) \\
x^{[s]}_{b,i} &= x'^{[s-1]}_{b,i} \oplus x^{[s]}_{c,i} \\
x^{[s]}_{a,i+8} &= x'^{[s-1]}_{a,i} \oplus x^{[s]}_{b,i} \oplus \Theta_i(x'^{[s-1]}_{a}, x^{[s]}_{b})
\end{aligned} \tag{9.5}$$

Inverting these equations, we get:

$$\begin{aligned}
x'^{[s-1]}_{a,i} &= x^{[s]}_{a,i+8} \oplus x^{[s]}_{b,i} \oplus \Theta_i(x'^{[s-1]}_{a}, x^{[s]}_{b}) && (9.6)\\
x'^{[s-1]}_{b,i} &= x^{[s]}_{b,i} \oplus x^{[s]}_{c,i} && (9.7)\\
x'^{[s-1]}_{c,i} &= x^{[s]}_{c,i} \oplus x^{[s]}_{d,i} \oplus \Theta_i(x'^{[s-1]}_{c}, x^{[s]}_{d}) && (9.8)\\
x'^{[s-1]}_{d,i} &= x^{[s]}_{d,i+27} \oplus x^{[s]}_{e,i} \oplus \Theta_i(x'^{[s-1]}_{d}, x^{[s]}_{e}) && (9.9)\\
x^{[s-1]}_{e,i} &= \mathcal{L}^{[s]}_{e,i} \oplus \Theta_i(x'^{[s-1]}_{a}, x^{[s]}_{b}) && (9.10)\\
x^{[s-1]}_{a,i} &= \mathcal{L}^{[s]}_{a,i} \oplus \Theta_i(x'^{[s-1]}_{a}, x^{[s]}_{b}) \oplus \Theta_i(x^{[s-1]}_{a}, x'^{[s-1]}_{b}) && (9.11)\\
x^{[s-1]}_{b,i} &= \mathcal{L}^{[s]}_{b,i} \oplus \Theta_i(x'^{[s-1]}_{c}, x^{[s]}_{d}) \oplus \Theta_i(x^{[s-1]}_{b}, x'^{[s-1]}_{c}) && (9.12)\\
x^{[s-1]}_{c,i} &= \mathcal{L}^{[s]}_{c,i} \oplus \Theta_i(x'^{[s-1]}_{c}, x^{[s]}_{d}) \oplus \Theta_i(x'^{[s-1]}_{d}, x^{[s]}_{e}) && (9.13)\\
x^{[s-1]}_{d,i} &= \mathcal{L}^{[s]}_{d,i} \oplus \Theta_i(x'^{[s-1]}_{d}, x^{[s]}_{e}) \oplus \Theta_i(x'^{[s-1]}_{a}, x^{[s]}_{b}) \oplus \Theta_i(x^{[s-1]}_{d}, x^{[s-1]}_{e}) && (9.14)
\end{aligned}$$

where

$$\begin{aligned}
\mathcal{L}^{[s]}_{a,i} &= x^{[s]}_{a,i+8} \oplus x^{[s]}_{c,i} && (9.15)\\
\mathcal{L}^{[s]}_{b,i} &= x^{[s]}_{b,i+10} \oplus x^{[s]}_{c,i+10} \oplus x^{[s]}_{c,i} \oplus x^{[s]}_{d,i} && (9.16)\\
\mathcal{L}^{[s]}_{c,i} &= x^{[s]}_{c,i} \oplus x^{[s]}_{d,i} \oplus x^{[s]}_{d,i+27} \oplus x^{[s]}_{e,i} && (9.17)\\
\mathcal{L}^{[s]}_{d,i} &= x^{[s]}_{d,i+27} \oplus x^{[s]}_{a,i+8} \oplus x^{[s]}_{b,i} && (9.18)\\
\mathcal{L}^{[s]}_{e,i} &= x^{[s]}_{e,i} \oplus x^{[s]}_{a,i+8} \oplus x^{[s]}_{b,i} && (9.19)
\end{aligned}$$

We start with simple linear approximations for $x^{[s-1]}_{a,i}$ to $x^{[s-1]}_{e,i}$.

Lemma 9.1 It holds that $x^{[s-1]}_{l,0} = \mathcal{L}^{[s]}_{l,0}$, for $l \in \{a, b, c, d, e\}$.

proof.

This result follows directly from Eqs. (9.10)-(9.14) by using the fact that $\Theta_0(x, y) = 0$. ■

Lemma 9.2 The following holds for $i > 0$

$$x_{e,i}^{[s-1]} = \mathcal{L}_{e,i}^{[s]} \oplus x_{b,i-1}^{[s]}, \quad \text{w.p. } \frac{1}{2}\left(1+\frac{1}{2}\right), \tag{9.20}$$

$$x_{a,i}^{[s-1]} = \mathcal{L}_{a,i}^{[s]} \oplus x_{c,i-1}^{[s]}, \quad \text{w.p. } \frac{1}{2}\left(1+\frac{1}{2^2}\right), \tag{9.21}$$

$$x_{b,i}^{[s-1]} = \mathcal{L}_{b,i}^{[s]} \oplus x_{c,i-1}^{[s]} \oplus x_{d,i-1}^{[s]}, \quad \text{w.p. } \frac{1}{2}\left(1+\frac{1}{2^2}\right), \tag{9.22}$$

$$x_{c,i}^{[s-1]} = \mathcal{L}_{c,i}^{[s]} \oplus x_{d,i-1}^{[s]} \oplus x_{e,i-1}^{[s]}, \quad \text{w.p. } \frac{1}{2}\left(1+\frac{1}{2^2}\right), \tag{9.23}$$

$$x_{d,i}^{[s-1]} = \mathcal{L}_{d,i}^{[s]} \oplus \mathcal{L}_{e,i-1}^{[s]} \oplus x_{e,i-1}^{[s]}, \quad \text{w.p. } \frac{1}{2}\left(1+\frac{1}{2^3}\right). \tag{9.24}$$

proof.

For $x_{a,i}^{[s-1]}, x_{c,i}^{[s-1]}$ and $x_{e,i}^{[s-1]}$, this result follows directly from Eq. (6.14) and the Piling-up Lemma. For $x_{b,i}^{[s-1]}$, from Eq. (9.12) and using Eq. (6.14) on the last term we get

$$x_{b,i}^{[s-1]} = \mathcal{L}_{b,i}^{[s]} \oplus \Theta_i(x_c'^{[s-1]}, x_d^{[s]}) \oplus x_{c,i-1}'^{[s-1]}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Using Eq. (9.8)

$$x_{b,i}^{[s-1]} = \mathcal{L}_{b,i}^{[s]} \oplus \Theta_i(x_c'^{[s-1]}, x_d^{[s]}) \oplus x_{c,i-1}^{[s]} \oplus x_{d,i-1}^{[s]} \oplus \Theta_{i-1}(x_c'^{[s-1]}, x_d^{[s]}).$$

Finally, using Eq. (6.15) and the Piling-up Lemma we get

$$x_{b,i}^{[s-1]} = \mathcal{L}_{b,i}^{[s]} \oplus x_{c,i-1}^{[s]} \oplus x_{d,i-1}^{[s]},$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Next, for $x_{d,i}^{[s-1]}$, from Eq. (9.14) and using Eq. (6.14) to replace $\Theta_i(x_d'^{[s-1]}, x_e^{[s]})$ and $\Theta_i(x_d^{[s-1]}, x_e^{[s-1]})$ we get

$$x_{d,i}^{[s-1]} = \mathcal{L}_{d,i}^{[s]} \oplus x_{e,i-1}^{[s]} \oplus \Theta_i(x_a'^{[s-1]}, x_b^{[s]}) \oplus x_{e,i-1}^{[s-1]}.$$

Then, using Eq. (9.10) we get

$$x_{d,i}^{[s-1]} = \mathcal{L}_{d,i}^{[s]} \oplus x_{e,i-1}^{[s]} \oplus \Theta_i(x_a'^{[s-1]}, x_b^{[s]}) \oplus \mathcal{L}_{e,i-1}^{[s]} \oplus \Theta_{i-1}(x_a'^{[s-1]}, x_b^{[s]}).$$

Finally, using Eq. (6.15) and the Piling-up Lemma we get

$$x_{d,i}^{[s-1]} = \mathcal{L}_{d,i}^{[s]} \oplus x_{e,i-1}^{[s]} \oplus \mathcal{L}_{e,i-1}^{[s]},$$

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

LEMMA 9.3 For two active input bits in subround $s-1$ and multiple active output bits in subround s, the following holds for $i > 1$

$$x_{\lambda,i}^{[s-1]} \oplus x_{\lambda,i-1}^{[s-1]} = \mathcal{L}_{\lambda,i}^{[s]} \oplus \mathcal{L}_{\lambda,i-1}^{[s]}, \text{ w.p. } \frac{1}{2}\left(1+\frac{1}{2^{\sigma}}\right),$$

where $(\lambda, \sigma) \in \{(a,2),(b,2),(c,2),(d,3),(e,1)\}$.

proof.

This proof follows directly from Eqs. (9.10)-(9.14) using the approximation of Eq. (6.15) and the Piling-up Lemma. ■

We also derive some additional lemmas to deal with special cases:

LEMMA 9.4 For two active input bits in subround $s-1$ and multiple active output bits in subround s, the following holds for $i > 0$

$$x_{e,i}^{[s-1]} \oplus x_{a,i}^{[s-1]} = \mathcal{L}_{e,i}^{[s]} \oplus \mathcal{L}_{a,i}^{[s]} \oplus x_{b,i-1}^{[s]} \oplus x_{c,i-1}^{[s]}, \text{ w.p. } \frac{1}{2}\left(1+\frac{1}{2}\right).$$

proof.

This proof follows directly from Eqs. (9.10) and (9.11) canceling out the term $\Theta_i(x_a'^{[s-1]}, x_b^{[s]})$, using the approximation of Eq. (6.14) and the Piling-up Lemma. ■

LEMMA 9.5 For three active input bits in subround $s-1$ and multiple active output bits in subround s, the following holds for $i > 1$

$$x_{e,i}^{[s-1]} \oplus x_{e,i-1}^{[s-1]} \oplus x_{d,i}^{[s-1]} = \mathcal{L}_{e,i}^{[s]} \oplus \mathcal{L}_{d,i}^{[s]} \oplus x_{e,i-1}^{[s]}, \text{ w.p. } \frac{1}{2}\left(1+\frac{1}{2^2}\right).$$

proof.

From Eq. (9.14) notice that using Eq. (6.14) we can write

$$\begin{aligned} x_{d,i}^{[s-1]} &= \mathcal{L}_{d,i}^{[s]} \oplus \Theta_i(x_d'^{[s-1]}, x_e^{[s]}) \oplus \Theta_i(x_a'^{[s-1]}, x_b^{[s]}) \oplus \Theta_i(x_d^{[s-1]}, x_e^{[s-1]}) = \\ &\mathcal{L}_{d,i}^{[s]} \oplus \Theta_i(x_d'^{[s-1]}, x_e^{[s]}) \oplus \Theta_i(x_a'^{[s-1]}, x_b^{[s]}) \oplus x_{e,i-1}^{[s-1]}, \text{ w.p. } \tfrac{1}{2}\left(1+\tfrac{1}{2}\right). \end{aligned}$$

Thus, we have

$$x_{e,i}^{[s-1]} \oplus x_{e,i-1}^{[s-1]} \oplus x_{d,i}^{[s-1]} = x_{e,i}^{[s-1]} \oplus \mathcal{L}_{d,i}^{[s]} \oplus \Theta_i(x_d'^{[s-1]}, x_e^{[s]}) \oplus \Theta_i(x_a'^{[s-1]}, x_b^{[s]}).$$

Next, using Eq. (9.10) and canceling out equal terms, we get

$$x_{e,i}^{[s-1]} \oplus x_{e,i-1}^{[s-1]} \oplus x_{d,i}^{[s-1]} = \mathcal{L}_{e,i}^{[s]} \oplus \mathcal{L}_{d,i}^{[s]} \oplus \Theta_i(x_d'^{[s-1]}, x_e^{[s]}).$$

Finally, using Eq. (6.14) and the Piling-up Lemma, completes the proof. ■

LEMMA 9.6 For three active input bits in subround $s-1$ and multiple active output bits in subround s, the following holds for $i > 1$

$$x_{e,i}^{[s-1]} \oplus x_{d,i}^{[s-1]} \oplus x_{d,i\pm1}^{[s-1]} = \mathcal{L}_{e,i}^{[s]} \oplus \mathcal{L}_{d,i}^{[s]} \oplus \mathcal{L}_{d,i\pm1}^{[s]} \oplus x_{b,i-1\pm1}^{[s]}, \text{ w.p. } \frac{1}{2}\left(1+\frac{1}{2^3}\right).$$

proof.

From Eqs. (9.10) and (9.14) we can cancel out the term $\Theta_i(x_a'^{[s-1]}, x_b^{[s]})$ and get

$$x_{e,i}^{[s-1]} \oplus x_{d,i}^{[s-1]} \oplus x_{d,i\pm1}^{[s-1]} = \mathcal{L}_{e,i}^{[s]} \oplus \mathcal{L}_{d,i}^{[s]} \oplus \Theta_i(x_d'^{[s-1]}, x_e^{[s]}) \oplus \Theta_i(x_d^{[s-1]}, x_e^{[s-1]}) \oplus$$
$$\mathcal{L}_{d,i\pm1}^{[s]} \oplus \Theta_{i\pm1}(x_d'^{[s-1]}, x_e^{[s]}) \oplus \Theta_{i\pm1}(x_a'^{[s-1]}, x_b^{[s]}) \oplus \Theta_{i\pm1}(x_d^{[s-1]}, x_e^{[s-1]}).$$

Using Eq. (6.15) to approximate $\Theta_i(x_d'^{[s-1]}, x_e^{[s]}) \oplus \Theta_{i\pm1}(x_d'^{[s-1]}, x_e^{[s]})$ and $\Theta_i(x_d^{[s-1]}, x_e^{[s-1]}) \oplus \Theta_{i\pm1}(x_d^{[s-1]}, x_e^{[s-1]})$, Eq. (6.14) to approximate $\Theta_{i\pm1}(x_a'^{[s-1]}, x_b^{[s]})$, and the Piling-up Lemma completes the proof. ■

LEMMA 9.7 For four active input bits in subround $s-1$ and multiple active output bits in subround s, the following holds for $i > 1$

$$x_{b,i}^{[s-1]} \oplus x_{c,i}^{[s-1]} \oplus x_{b,i-1}^{[s-1]} \oplus x_{c,i-1}^{[s-1]} = \mathcal{L}_{b,i}^{[s]} \oplus \mathcal{L}_{c,i}^{[s]} \oplus \mathcal{L}_{b,i-1}^{[s]} \oplus \mathcal{L}_{c,i-1}^{[s]}, \text{ w.p. } \frac{1}{2}\left(1+\frac{1}{2^2}\right).$$

proof.

From Eqs. (9.12) and (9.13) we can cancel out the terms $\Theta_i(x_c'^{[s-1]}, x_d^{[s]})$ and $\Theta_{i-1}(x_c'^{[s-1]}, x_d^{[s]})$.

$\mathcal{ID}$	$\mathcal{OD}$	Correlation
$\Delta X_5^{(0)} = 2^{18}$	$\Delta X_{15}^{(2)} = 2^7$	-0.00379
$\Delta X_5^{(0)} = 2^{18}$	$\Delta X_{10}^{(2)} = 2^7$	-0.00221
$\Delta X_5^{(0)} = 2^{11}$	$\Delta X_{15}^{(2)} = 1$	-0.00139
$\Delta X_5^{(0)} = 2^{11}$	$\Delta X_{10}^{(2)} = 1$	-0.00053

Table 9.1 – Some of the best single bit differentials for 2 rounds of Forró.

Thus, we get

$$x_{b,i}^{[s-1]} \oplus x_{c,i}^{[s-1]} \oplus x_{b,i-1}^{[s-1]} \oplus x_{c,i-1}^{[s-1]} = \mathcal{L}_{b,i}^{[s]} \oplus \Theta_i(x_b^{[s-1]}, x_c'^{[s-1]}) \oplus \mathcal{L}_{c,i}^{[s]} \oplus \\ \Theta_i(x_d'^{[s-1]}, x_e^{[s]}) \oplus \mathcal{L}_{b,i-1}^{[s]} \oplus \Theta_{i-1}(x_b^{[s-1]}, x_c'^{[s-1]}) \oplus \mathcal{L}_{c,i-1}^{[s]} \oplus \Theta_{i-1}(x_d'^{[s-1]}, x_e^{[s]}).$$

Applying Eq. (6.15) and the Piling-up Lemma completes the proof.

■

9.2.2 Distinguishers

We constructed distinguishers for Forró by following the best techniques used against ChaCha in the literature [24, 72]. More precisely, we looked for single bit differentials ranging 2 and 3 rounds of Forró. To do so, we tested all possible single bit input differences (128 possibilities) combined with every possible single bit output difference (512 possibilities). Hence, we tested a total of 2^{15} differentials. In each case, we estimated the correlation experimentally with a total of 2^{34} random samples. We present some examples in Table 9.1.

Next, we implemented a program capable of expanding the linear equations of Forró automatically using Lemmas 9.1, 9.2 and 9.3. With this program we constructed distinguishers against Forró for every single differential that had a statistically significant correlation. Of this study, we concluded in the best distinguisher for 3, 4, 5 and 5.25 rounds of Forró. We could not find any distinguishers against 5.5 rounds of Forró or more. In the following, we theoretically demonstrate these distinguishers.

9.2.2.1 Distinguisher against 3 rounds of Forró.

In this case, consider that single bit differential with $\mathcal{OD} = \Delta X_{15}^{(2)} = 1$ presented in Table 9.1. Thus, we have $\varepsilon_d = 0.00139$. For the linear part, we have to expand the bit

$x^{(2)}_{15,0} = x^{[8]}_{15,0}$. Note, from Eqs (9.2) and (9.3) that we have

$$(a,b,c,d,e) = \begin{cases} (0,4,8,12,3), & s = 1,9,17,... \\ (1,5,9,13,0), & s = 2,10,18,... \\ (2,6,10,14,1), & s = 3,11,19,... \\ (3,7,11,15,2), & s = 4,12,20,... \\ (0,5,10,15,3), & s = 5,13,21,... \\ (1,6,11,12,0), & s = 6,14,22,... \\ (2,7,8,13,1), & s = 7,15,23,... \\ (3,4,9,14,2), & s = 8,16,24,... \end{cases} \tag{9.25}$$

Therefore, considering that for subrounds 9, 10 and 11 we do not update the word X_{15}, we have

$$x^{[8]}_{15,0} = x^{[9]}_{15,0} = x^{[10]}_{15,0} = x^{[11]}_{15,0}.$$

Then, in subround 12, we have that $(a,b,c,d,e) = (3,7,11,15,2)$. Thus, X_{15} is of type X_d and using Lemma 9.1 we have $x^{[11]}_{15,0} = x^{[12]}_{15,27} \oplus x^{[12]}_{3,8} \oplus x^{[12]}_{7,0}$, with probability 1. Clearly, $\varepsilon_L = 1$, then the complexity of the differential-linear distinguisher for 3 rounds of Forró is $\frac{1}{\varepsilon_d^2} \approx 2^{18.9814}$.

9.2.2.2 Distinguisher against 4 rounds of Forró.

In this case, consider that single bit differential with $\mathcal{OD} = \Delta X^{(2)}_{10} = 1$ presented in Table 9.1. Thus, we have $\varepsilon_d = 0.00053$. For the linear part, we have to expand the bit $x^{(2)}_{10,0} = x^{[8]}_{10,0}$, which results in the following Lemma:

LEMMA 9.8 The following linear approximation holds with probability $\frac{1}{2}\left(1 + \frac{1}{2^5}\right)$

$$x^{[8]}_{10,0} = x^{[16]}_1[8] \oplus x^{[16]}_2[16] \oplus x^{[16]}_3[2,3,24] \oplus x^{[16]}_4[0,15,16,26,27] \oplus$$
$$x^{[16]}_7[7,8] \oplus x^{[16]}_9[0] \oplus x^{[16]}_{10}[0] \oplus x^{[16]}_{11}[0] \oplus x^{[16]}_{14}[22,27] \oplus x^{[16]}_{15}[0,27].$$

proof.

See Appendix A.2. ■

COMPUTATIONAL RESULT 9.1 The linear approximation of Lemma 9.8 holds computationally with $\varepsilon_{L_0} = 0.0476 \approx 2^{-4.39}$. This correlation was verified using 2^{38} random samples.

We conclude that the complexity of the differential-linear distinguisher for 4 rounds of

Forró is $\frac{1}{\varepsilon_d^2 \varepsilon_{L_0}^4} \approx 2^{36.55}$.

9.2.2.3 Distinguisher against 5 rounds of Forró.

For 5 rounds, we keep expanding the final equation of Lemma 9.8, which results in the following Lemma:

LEMMA 9.9 The following linear approximation holds with probability $\frac{1}{2}\left(1+\frac{1}{2^{33}}\right)$

$$\begin{aligned} x_{10,0}^{[8]} = \; & x_0^{[20]}[10,11] \oplus x_1^{[20]}[0,8,16,18,19] \oplus x_2^{[20]}[0,2,3,16,26,27,29,30] \oplus \\ & x_3^{[20]}[0,5,6,8,24] \oplus x_4^{[20]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[20]}[0,10,11] \oplus \\ & x_6^{[20]}[7,8,15,16,18,19,21,22,26,27] \oplus x_7^{[20]}[0,2,3,15,16,17,18,29,30] \oplus \\ & x_8^{[20]}[0,4,5,10,15,16,25,27] \oplus x_9^{[20]}[0,7,8] \oplus x_{10}^{[20]}[0,15,16] \oplus \\ & x_{11}^{[20]}[0,2,3,7,8,17,18,23,24] \oplus x_{12}^{[20]}[0,15,16,26,27] \oplus x_{13}^{[20]}[0,27] \oplus \\ & x_{14}^{[20]}[0,17,22,27] \oplus x_{15}^{[20]}[0,7,8,22] \end{aligned}$$

proof.

See Appendix A.2. ■

COMPUTATIONAL RESULT 9.2 The linear approximations of Eqs. (A.9)-(A.12) hold computationally with $\varepsilon_{L_1} = 0.0278$, $\varepsilon_{L_2} = 0.1667$, $\varepsilon_{L_3} = 0.0046$ and $\varepsilon_{L_4} = 0.0046$, respectively. This correlation was verified using 2^{38} random samples.

We conclude that the complexity of the differential-linear distinguisher for 5 rounds of Forró is $\frac{1}{\varepsilon_d^2(\varepsilon_{L_0}\varepsilon_{L_1}\varepsilon_{L_2}\varepsilon_{L_3}\varepsilon_{L_4})^4} \approx 2^{129.68}$.

9.2.2.4 Distinguisher against 5.25 rounds of Forró.

For 5.25 rounds, we expand the final equation of Lemma 9.9, which results in the following Lemma:

LEMMA 9.10 The following linear approximation holds with probability $\frac{1}{2}\left(1+\frac{1}{2^{47}}\right)$

$$\begin{aligned} x_{10,0}^{[8]} = \; & x_0^{[21]}[0,13,14,15,18,19,29,30] \oplus x_1^{[21]}[0,8,16,18,19] \oplus \\ & x_2^{[21]}[0,2,3,16,26,27,29,30] \oplus x_3^{[21]}[5,6,8,15,16,24] \oplus \\ & x_4^{[21]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[21]}[5,7,10,20,22,23,24] \oplus \\ & x_6^{[21]}[7,8,15,16,18,19,21,22,26,27] \oplus x_7^{[21]}[0,2,3,15,16,17,18,29,30] \oplus \\ & x_8^{[21]}[0,4,5,10,15,16,25,27] \oplus x_9^{[21]}[0,7,8] \oplus x_{10}^{[21]}[10,15,16,20,21] \oplus \\ & x_{11}^{[21]}[0,2,3,7,8,17,18,23,24] \oplus x_{12}^{[21]}[0,15,16,26,27] \oplus x_{13}^{[21]}[0,27] \oplus \\ & x_{14}^{[21]}[0,17,22,27] \oplus x_{15}^{[21]}[2,3,15,16,17] \end{aligned}$$

proof.

In subround 21, we have $(a,b,c,d,e) = (0,5,10,15,3)$. Thus, from Eq. (A.12) we have to expand the terms $x_0^{[20]}[10,11]$, $x_3^{[20]}[0,5,6,8,24]$, $x_5^{[20]}[0,10,11]$, $x_{10}^{[20]}[0,15,16]$ and $x_{15}^{[20]}[0,7,8,22]$. Here, we use Lemma 9.1 to expand $x_{3,0}^{[20]}$, $x_{5,0}^{[20]}$, $x_{10,0}^{[20]}$ and $x_{15,0}^{[20]}$ with probability 1. Then, we use Lemma 9.6 to expand $x_{15}^{[20]}[7,8] \oplus x_{3,8}^{[20]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$. Next, using Lemma 9.3 we can expand $x_0^{[20]}[10,11]$, $x_5^{[20]}[10,11]$ and $x_{10}^{[20]}[15,16]$ with probabilities $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$, and $x_3^{[20]}[5,6]$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Additionally, with Lemma 9.2 we can expand $x_{3,24}^{[20]}$ and $x_{15,22}^{[20]}$ with probabilities $\frac{1}{2}\left(1+\frac{1}{2}\right)$ and $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$, respectively. Thus, using the Piling-up Lemma we have that

$$\begin{aligned} & x_0^{[20]}[10,11] \oplus x_1^{[20]}[0,8,16,18,19] \oplus x_2^{[20]}[0,2,3,16,26,27,29,30] \oplus x_3^{[20]}[0,5,6, \\ & 8,24] \oplus x_4^{[20]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[20]}[0,10,11] \oplus x_6^{[20]}[7,8,15,16,18, \\ & 19,21,22,26,27] \oplus x_7^{[20]}[0,2,3,15,16,17,18,29,30] \oplus x_8^{[20]}[0,4,5,10,15,16,25, \\ & 27] \oplus x_9^{[20]}[0,7,8] \oplus x_{10}^{[20]}[0,15,16] \oplus x_{11}^{[20]}[0,2,3,7,8,17,18,23,24] \oplus x_{12}^{[20]}[0,15, \\ & 16,26,27] \oplus x_{13}^{[20]}[0,27] \oplus x_{14}^{[20]}[0,17,22,27] \oplus x_{15}^{[20]}[0,7,8,22] = x_0^{[21]}[0,13,14, \\ & 15,18,19,29,30] \oplus x_1^{[21]}[0,8,16,18,19] \oplus x_2^{[21]}[0,2,3,16,26,27,29,30] \oplus \\ & x_3^{[21]}[5,6,8,15,16,24] \oplus x_4^{[21]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[21]}[5,7,10,20,22, \\ & 23,24] \oplus x_6^{[21]}[7,8,15,16,18,19,21,22,26,27] \oplus x_7^{[21]}[0,2,3,15,16,17,18, \\ & 29,30] \oplus x_8^{[21]}[0,4,5,10,15,16,25,27] \oplus x_9^{[21]}[0,7,8] \oplus x_{10}^{[21]}[10,15,16,20,21] \oplus \\ & x_{11}^{[21]}[0,2,3,7,8,17,18,23,24] \oplus x_{12}^{[21]}[0,15,16,26,27] \oplus x_{13}^{[21]}[0,27] \oplus \\ & x_{14}^{[21]}[0,17,22,27] \oplus x_{15}^{[21]}[2,3,15,16,17] \end{aligned} \tag{9.26}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^{14}}\right)$.

■

Computational Result 9.3 The linear approximations of Eq. (9.26) hold computationally with $\varepsilon_{L_5} = 0.000284$. This correlation was verified using 2^{38} random samples.

Using Computational Result 9.3 we can estimate the complexity of the differential-linear distinguisher for 5.25 rounds of Forró as $\frac{1}{\varepsilon_d^2(\varepsilon_{L_0}\varepsilon_{L_1}\varepsilon_{L_2}\varepsilon_{L_3}\varepsilon_{L_4}\varepsilon_{L_5})^4} \approx 2^{176.81}$. We also expanded the linear approximation to subround 22 but it did not lead to a significant attack, more precisely, the attack for 5.5 rounds would have complexity of 2^{296}.

9.2.3 Attacks Using PNBs

In this section we use the techniques developed by [19] and later improved by [24] to attack Forró. We tested several different attacks for different values of γ for all differentials presented in Table 9.1. With this approach, the best attack we found against 5 rounds of Forró uses 2 rounds forward and 3 rounds backwards. The attack uses the differential $(\Delta_{10,0}^{(2)}|\Delta_{5,11}^{(0)})$, thus, from Table 9.1 we get $\varepsilon_d = -0.00053$. Using $\gamma = 0.25$ we get a total of 155 PNBs. From that, we estimated $\varepsilon_a = 0.000068$ which leads to an attack with data complexity of 2^{57} and time complexity of 2^{158}.

9.3 PERFORMANCE

Forró by design has less operations than ChaCha, the implication being that on embedded devices with limited concurrency capabilities, such as ARM processors, Forró naturally has better performance[1]. In more advanced processors with speculative execution and out-of-order execution, such as modern x86, ChaCha has the advantage that all quarter round operations in a round are independent, while Forró, because of the Pollination, has a serial dependency between the quarter round functions. In order to work around this apparent limitation on concurrency, a few points need to be explained.

In order to pipeline instructions, the processor detects (or speculates) instructions that don't have dependencies on each others output and are nearby to anticipate them, so while one executes, the other can be fetching, for example. In ChaCha, the QRF is applied independently inside a round, and pipelining occurs without much impediment. In Forro, because of Pollination, every operation in a round has a dependency on the previous output, causing a serial data dependency. Meaning that the processor can't detect independent instructions to pipeline, or if it guesses the instructions are likely to not retire.

[1]The results presented in this section were achieved with the contribution of my friend Iago Passos

Network	**MTU (bytes)**
16 Mbps Token Ring	17914
4 Mbps Token Ring	4464
FDDI	4352
Ethernet	1500
IEEE 802.3/802.2	1492
PPPoE (WAN Miniport)	1480
X.25	576

Table 9.2 – Typical MTU of different types of networks.

However, just like ChaCha, in order to get the next 512 bits of keystream, the algorithm needs to be executed from the start with an increment on the counter. This execution is completely independent of the previous one. Unfortunately, the processor doesn't have the foresight to anticipate that, since the code for it is far into the future, but that can be bypassed.

To take full advantage of pipelining, whenever possible, we implement it so that 2 executions of Forró are in the same loop, which we refer as the parallel version of the implementation . This strategy prevents us from losing too much performance in modern architectures, and still be able to leverage the fact we have less instructions in order to beat ChaCha's performance.

In the following tables we compare the performance of Forro against ChaCha and Salsa. From the tables we can see that Forro is faster in constrained devices such as ARM-v7, being a good choice for IOT devices. We note that for these results all other processes from the machine were turn off to not affect the results.

	576 bytes		1480 bytes		1492 bytes		1500 bytes	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
Salsa	2948	0	8018	1	7878	1	7856	1
ChaCha12	1910	0	5238	1	5132	1	5144	1
ChaCha20	2878	0	7850	1	7680	1	7720	1
ChaCha20 (Parallel)	3034	0	7430	1	7586	1	7282	1
Forro10	3522	0	9640	1	9642	1	9438	1
Forro12	4174	0	11406	2	11398	2	11178	2
Forro14	4818	0	13172	2	13176	2	12894	2
Forro10 (Parallel)	2360	0	5754	1	5872	1	5642	1
Forro12 (Parallel)	2756	0	6734	1	6874	1	6600	1
Forro14 (Parallel)	3190	0	7778	1	7950	1	7624	1

	4096 bytes		4352 bytes		4464 bytes		17914 bytes	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
Salsa	20542	4	23142	4	23382	4	91556	18
ChaCha12	13462	2	15134	3	15306	3	59992	12
ChaCha20	20118	4	21890	4	23078	4	92084	18
ChaCha20 (Parallel)	19362	3	21006	4	22172	4	88136	17
Forro10	25134	5	26698	5	28110	5	114610	23
Forro12	29756	6	32994	6	33280	6	132970	27
Forro14	34472	7	36588	7	39310	8	153938	31
Forro10 (Parallel)	15326	3	16280	3	16842	3	65482	13
Forro12 (Parallel)	17536	3	18988	3	20082	4	76578	15
Forro14 (Parallel)	20748	4	22062	4	23194	4	90766	18

Table 9.3 – Timings of Salsa, ChaCha and Forró's reference implementations on an x86_64. Values are the median of 10001 measurements. And the size of the packet is based on common network MTU sizes following Table 9.2.

	576 bytes		1480 bytes		1492 bytes		1500 bytes	
Algorithm	**Cycles**	μ**s**	**Cycles**	μ**s**	**Cycles**	μ**s**	**Cycles**	μ**s**
Salsa	850	0	2814	0	2814	0	2808	0
ChaCha12	494	0	1644	0	1644	0	1656	0
ChaCha20	752	0	2522	0	2526	0	2536	0
ChaCha20 (Parallel)	1326	0	2538	0	2544	0	2548	0
Forro10	1010	0	3258	0	3256	0	3260	0
Forro12	1188	0	3856	0	3866	0	3870	0
Forro14	1396	0	4522	0	4524	0	4522	0
Forro10 (Parallel)	1246	0	2718	0	2718	0	2728	0
Forro12 (Parallel)	1472	0	3186	0	3192	0	3198	0
Forro14 (Parallel)	1676	0	3644	0	3644	0	3648	0

	4096 bytes		4352 bytes		4464 bytes		17914 bytes	
Algorithm	**Cycles**	μ**s**	**Cycles**	μ**s**	**Cycles**	μ**s**	**Cycles**	μ**s**
Salsa	4418	0	4940	1	5772	1	20740	4
ChaCha12	2688	0	2956	0	3480	0	12150	2
ChaCha20	3934	0	4434	0	5264	1	18100	3
ChaCha20 (Parallel)	4480	0	5002	1	5868	1	20302	4
Forro10	4548	0	5058	1	6178	1	21230	4
Forro12	5406	1	6032	1	7244	1	25294	5
Forro14	6244	1	6966	1	8422	1	29366	5
Forro10 (Parallel)	3558	0	4090	0	5164	1	17098	3
Forro12 (Parallel)	4142	0	4782	0	6054	1	19586	3
Forro14 (Parallel)	4826	0	5568	1	6928	1	22790	4

Table 9.4 – Timings of Salsa, ChaCha and Forró's SIMD implementations on an x86_64. Values are the median of 10001 measurements. And the size of the packet is based on common network MTU sizes following Table 9.2.

	576 bytes		1480 bytes		1492 bytes		1500 bytes	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
Salsa	12540	19	31714	48	31808	48	31824	48
ChaCha12	10011	15	24850	38	24970	38	24964	38
ChaCha20	13424	20	33972	52	34030	52	34050	52
ChaCha20 (Parallel)	22042	33	52336	80	52340	80	52258	80
Forro10	9181	14	22709	34	22810	35	22778	35
Forro12	10212	15	25474	39	25547	39	25554	39
Forro14	11082	17	27798	42	27886	42	27889	42
Forro10 (Parallel)	10252	15	24373	37	24167	37	23870	36
Forro12 (Parallel)	11150	17	26250	40	25988	39	25755	39
Forro14 (Parallel)	12194	18	28836	44	28970	44	28332	80

	4096 bytes		4352 bytes		4464 bytes		17914 bytes	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
Salsa	83689	128	89108	137	91369	140	364660	560
ChaCha12	65263	100	69333	106	71337	109	283103	435
ChaCha20	89495	137	95363	146	97854	150	388863	598
ChaCha20 (Parallel)	138284	212	147484	226	151109	232	605445	931
Forro10	59816	92	63308	97	65075	100	257782	396
Forro12	67098	103	71102	109	73100	112	290095	446
Forro14	73230	112	77833	119	79928	122	317966	489
Forro10 (Parallel)	62365	95	66096	101	69852	107	277038	426
Forro12 (Parallel)	69317	106	73383	112	74569	114	297071	456
Forro14 (Parallel)	76236	117	80906	124	83172	127	330549	509

Table 9.5 – Timings of Salsa, ChaCha and Forró's reference implementations on an ARMv7. Values are the median of 10001 measurements. And the size of the packet is based on common network MTU sizes following Table 9.2.

	576 bytes		**1480 bytes**		**1492 bytes**		**1500 bytes**	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
ChaCha12	6269	9	15278	23	15358	23	15368	23
ChaCha20	8637	13	21925	33	21996	33	22020	33
Forro10	6580	10	16154	24	16187	24	16186	24
Forro12	7311	11	18113	27	18208	28	18220	28
Forro14	8118	12	20321	31	20380	31	20417	31

	4096 bytes		**4352 bytes**		**4464 bytes**		**17914 bytes**	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
ChaCha12	35652	54	37803	58	39651	60	154255	237
ChaCha20	51914	79	55108	84	57738	88	226416	348
Forro10	38409	59	40747	62	42498	65	166003	255
Forro12	43589	67	46292	71	48226	74	189334	291
Forro14	49575	76	52658	81	54764	84	215532	331

Table 9.6 – Timings of ChaCha and Forró's NEON implementations on an ARMv7. Values are the median of 10001 measurements. And the size of the packet is based on common network MTU sizes following Table 9.2.

	576 bytes		**1480 bytes**		**1492 bytes**		**1500 bytes**	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
Salsa	3542	1	9285	5	9290	5	9293	5
ChaCha12	3279	1	8617	4	8638	4	8627	4
ChaCha20	5011	2	13211	7	13219	7	13221	7
ChaCha20 (Parallel)	5703	3	13627	7	13635	7	13637	7
Forro10	4915	2	12949	7	12876	7	12961	7
Forro12	5742	3	15252	8	15214	8	15216	8
Forro14	6630	3	17544	9	17566	9	17569	9
Forro10 (Parallel)	3588	1	8562	4	8559	4	8584	4
Forro12 (Parallel)	4179	2	9967	5	9975	5	9977	5
Forro14 (Parallel)	4708	2	11216	6	11227	6	11259	6

	4096 bytes		**4352 bytes**		**4464 bytes**		**17914 bytes**	
Algorithm	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs	**Cycles**	μs
Salsa	24622	13	26156	14	26928	14	108281	60
ChaCha12	22844	12	24268	13	24984	13	100300	55
ChaCha20	35100	19	37289	20	38386	21	153589	85
ChaCha20 (Parallel)	36214	20	147484	21	39599	21	159359	88
Forro10	34411	19	36330	20	37635	20	150649	83
Forro12	40561	22	43091	23	44355	24	176641	98
Forro14	46700	25	49614	27	51075	28	204260	113
Forro10 (Parallel)	22677	12	24086	13	24794	13	99525	55
Forro12 (Parallel)	26454	14	28102	15	28924	16	115610	64
Forro14 (Parallel)	29804	16	31666	17	32530	18	130157	72

Table 9.7 – Timings of Salsa, ChaCha and Forró's reference implementations on an ARMv8, working with 64-bit words. Values are the median of 10001 measurements. And the sizes are based on common network MTU sizes following Table 9.2.

10 CONCLUSIONS AND FUTURE WORKS

In this thesis, we proposed several contributions to symmetric cryptography. First, we developed novel techniques to measure the diffusion of cryptographic algorithms. We presented a framework to compute continuous generalizations of cryptographic algorithms. In addition, we present new metrics to measure the avalanche effect, the diffusion, and the influence of input bits to sections of the output. In particular, the Diffusion Factor can be used to compare the diffusion of secure algorithms without the need of reducing the number of rounds or even consider small subset of bits.

Then, we proposed a new framework, named *ColoreD*, useful to design and to compare the security of cryptographic algorithms against differential cryptanalysis. We showed that *ColoreD* allows us to consider continuous differences, instead of just binary (black and white) differences. The framework includes the Continuous Differential Cryptanalysis (CDC), a new theoretical type of attacks in which we are able to consider statistical properties generated by very small continuous differences. Additionally, we provided some examples of how to use *ColoreD* and a full analysis and comparison of AES and PRESENT, leading to one interesting conclusion that PRESENT would need at least 37 rounds to be as secure as AES by these metrics.

This thesis also provide several new advances to the cryptanalysis of ARX ciphers. In particular, we provided a new way to derive linear approximations for ChaCha, and described the first differential-linear distinguisher against ChaCha with complexity 2^{214}. In addition, using the proposed BLE, we improved attacks against Salsa. More precisely, we presented the first distinguishers against 7 and 8 rounds of Salsa with complexities 2^{109} and 2^{216}, and improved key recovery attacks achieving a complexity of 2^{212} for 8 rounds when the best know attack so far had complexity of $2^{244.9}$.

Using the knowledge gained in cryptanalysis of ARX, we proposed a new stream cipher called Forró. We showed that Forró can achieve the same security of ChaCha with fewer operations. Because of that, Forró can achieve faster performance in certain platforms, specially in constrained devices.

Last but not least, another important result from this thesis is the introduction of three new open-source solutions encompassing more than 30 thousand lines of code. The solutions are:

1. Cryptdances, a tool to perform cryptanalysis of ChaCha, Salsa and Forro in high performance environments. Available in <https://github.com/MurCoutinho/cryptDances>.
2. The Fundamental Probabilistic Cryptographic Operations (FPCO) library, which can

be used to implement a wide range of cryptographic algorithms, in particular ARX or AND-RX algorithms, with continuous probabilistic operations. Using the FPCO library is possible to study the confusion and diffusion of cryptographic algorithms or implement metrics such as the Continuous Avalanche Factor (CAF) and the Magnitude Factor (MF), useful to measure the avalanche effect. Available at <https://github.com/MurCoutinho/pda>.

3. Reference implementation for Forró and benchmarks for performance against ChaCha and Salsa. Available at <https://github.com/MurCoutinho/forro_cipher>.

For future works, it would be interesting to use CDA in a broader range of algorithms to better understand and compare their diffusion properties. Moreover, the use of continuous generalizations in cryptanalysis could be further investigated, in particular, using the CNM for identifying neutral bits or probabilistic neutral bits. Another idea would be to combine CDC with differential attacks based on neural networks [73]. In the theoretical side, it might be interesting to study the mathematical behavior of the continuous generalizations, in particular the curvature of the final function.

Also, the techniques developed in this paper may be used to improve cryptanalysis against other ARX primitives, such as Chaskey or the hash function Blake. Also, the security of Forro should be analyzed further, specially against other types of attacks, such as rotational cryptanalysis. Finally, the tool *CryptDances* can be used by researches to try to improve further attacks against Salsa, ChaCha, and Forró.

BIBLIOGRAPHY

1 DAUM, M. *Cryptanalysis of Hash functions of the MD4-family*. Tese (Doutorado) — Ruhr University Bochum, 2005. Disponível em: <http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/DaumMagnus/>.

2 SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948.

3 SHANNON, C. E. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, v. 28, n. 4, p. 656–715, 1949. Disponível em: <https://doi.org/10.1002/j.1538-7305.1949.tb00928.x>.

4 WEBSTER, A. F.; TAVARES, S. E. On the design of s-boxes. In: WILLIAMS, H. C. (Ed.). *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*. Springer, 1985. (Lecture Notes in Computer Science, v. 218), p. 523–534. Disponível em: <https://doi.org/10.1007/3-540-39799-X_41>.

5 PRENEEL, B.; GOVAERTS, R.; VANDEWALLE, J. Boolean functions satisfying higher order propagation criteria. In: DAVIES, D. W. (Ed.). *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*. Springer, 1991. (Lecture Notes in Computer Science, v. 547), p. 141–152. Disponível em: <https://doi.org/10.1007/3-540-46416-6_12>.

6 CUSICK, T. W.; STANICA, P. *Cryptographic Boolean functions and applications*. [S.l.]: Academic Press, 2017.

7 SRINIVASAN, C.; LAKSHMY, K.; SETHUMADHAVAN, M. Measuring diffusion in stream ciphers using statistical testing methods. *Defence Science Journal*, Defence Scientific Information & Documentation Centre, v. 62, n. 1, p. 6, 2012.

8 BERNSTEIN, D. J. The salsa20 family of stream ciphers. In: ROBSHAW, M. J. B.; BILLET, O. (Ed.). *New Stream Cipher Designs - The eSTREAM Finalists*. Springer, 2008, (Lecture Notes in Computer Science, v. 4986). p. 84–97. Disponível em: <https://doi.org/10.1007/978-3-540-68351-3_8>.

9 ROBSHAW, M. J. B.; BILLET, O. (Ed.). *New Stream Cipher Designs - The eSTREAM Finalists*. Springer, 2008. v. 4986. (Lecture Notes in Computer Science, v. 4986). ISBN 978-3-540-68350-6. Disponível em: <https://doi.org/10.1007/978-3-540-68351-3>.

10 BERNSTEIN, D. J. Chacha, a variant of salsa20. In: *Workshop Record of SASC*. [S.l.: s.n.], 2008. v. 8, p. 3–5.

11 BERNSTEIN, D. J. The poly1305-aes message-authentication code. In: GILBERT, H.; HANDSCHUH, H. (Ed.). *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. Springer, 2005. (Lecture Notes in Computer Science, v. 3557), p. 32–49. Disponível em: <https://doi.org/10.1007/11502760_3>.

12 LANGLEY, A. et al. Chacha20-poly1305 cipher suites for transport layer security (TLS). *RFC*, v. 7905, p. 1–8, 2016. Disponível em: <https://doi.org/10.17487/RFC7905>.

13 MULLER, S. *Documentation and Analysis of the Linux Random Number Generator - Federal Office for Information Security (Germany's)*. 2019. <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/LinuxRNG/LinuxRNG_EN.pdf;jsessionid=6B0F8D7795B80F5EADA3DB3DB3E4043B.1_cid360?__blob=publicationFile&v=19>.

14 TORVALDS, L. *Linux kernel source tree*. 2016. <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=818e607b57c94ade9824dad63a96c2ea6b21baf3>.

15 IANIX. *ChaCha Usage & Deployment*. 2020. <https://ianix.com/pub/chacha-deployment.html>. Accessed: 2020-01-13.

16 BONEH, D.; SHOUP, V. A graduate course in applied cryptography. *Draft 0.5*, 2020.

17 CROWLEY, P. Truncated differential cryptanalysis of five rounds of salsa20. *IACR Cryptol. ePrint Arch.*, v. 2005, p. 375, 2005. Disponível em: <http://eprint.iacr.org/2005/375>.

18 FISCHER, S. et al. Non-randomness in estream candidates salsa20 and TSC-4. In: BARUA, R.; LANGE, T. (Ed.). *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*. Springer, 2006. (Lecture Notes in Computer Science, v. 4329), p. 2–16. Disponível em: <https://doi.org/10.1007/11941378_2>.

19 AUMASSON, J. et al. New features of latin dances: Analysis of salsa, chacha, and rumba. In: NYBERG, K. (Ed.). *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*. Springer, 2008. (Lecture Notes in Computer Science, v. 5086), p. 470–488. Disponível em: <https://doi.org/10.1007/978-3-540-71039-4_30>.

20 SHI, Z. et al. Improved key recovery attacks on reduced-round salsa20 and chacha. In: KWON, T.; LEE, M.; KWON, D. (Ed.). *Information Security and Cryptology - ICISC 2012 - 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers*. Springer, 2012. (Lecture Notes in Computer Science, v. 7839), p. 337–351. Disponível em: <https://doi.org/10.1007/978-3-642-37682-5_24>.

21 MAITRA, S.; PAUL, G.; MEIER, W. Salsa20 cryptanalysis: New moves and revisiting old styles. *IACR Cryptol. ePrint Arch.*, v. 2015, p. 217, 2015. Disponível em: <http://eprint.iacr.org/2015/217>.

22 MAITRA, S. Chosen IV cryptanalysis on reduced round chacha and salsa. *Discret. Appl. Math.*, v. 208, p. 88–97, 2016. Disponível em: <https://doi.org/10.1016/j.dam.2016.02.020>.

23 DEY, S.; SARKAR, S. Improved analysis for reduced round salsa and chacha. *Discret. Appl. Math.*, v. 227, p. 58–69, 2017. Disponível em: <https://doi.org/10.1016/j.dam.2017.04.034>.

24 CHOUDHURI, A. R.; MAITRA, S. Significantly improved multi-bit differentials for reduced round salsa and chacha. *IACR Trans. Symmetric Cryptol.*, v. 2016, n. 2, p. 261–287, 2016. Disponível em: <https://doi.org/10.13154/tosc.v2016.i2.261-287>.

25 COUTINHO, M.; NETO, T. C. S. New multi-bit differentials to improve attacks against chacha. *IACR Cryptol. ePrint Arch.*, v. 2020, p. 350, 2020. Disponível em: <https://eprint.iacr.org/2020/350>.

26 BEIERLE, C.; LEANDER, G.; TODO, Y. Improved differential-linear attacks with applications to ARX ciphers. In: MICCIANCIO, D.; RISTENPART, T. (Ed.). *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*. Springer, 2020. (Lecture Notes in Computer Science, v. 12172), p. 329–358. Disponível em: <https://doi.org/10.1007/978-3-030-56877-1_12>.

27 COUTINHO, M.; NETO, T. C. S. Improved linear approximations to ARX ciphers and attacks against chacha. *IACR Cryptol. ePrint Arch.*, v. 2021, p. 224, 2021. Disponível em: <https://eprint.iacr.org/2021/224>.

28 DEY, S. et al. Revamped differential-linear cryptanalysis on reduced round chacha. In: *Eurocrypt*. [S.l.]: Springer-Verlag, 2022.

29 BEAULIEU, R. et al. The SIMON and SPECK lightweight block ciphers. In: *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015. p. 175:1–175:6. Disponível em: <https://doi.org/10.1145/2744769.2747946>.

30 DAEMEN, J.; RIJMEN, V. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition*. Springer, 2020. (Information Security and Cryptography). ISBN 978-3-662-60768-8. Disponível em: <https://doi.org/10.1007/978-3-662-60769-5>.

31 BOGDANOV, A. et al. PRESENT: an ultra-lightweight block cipher. In: PAILLIER, P.; VERBAUWHEDE, I. (Ed.). *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*. Springer, 2007. (Lecture Notes in Computer Science, v. 4727), p. 450–466. Disponível em: <https://doi.org/10.1007/978-3-540-74735-2_31>.

32 BIHAM, E.; SHAMIR, A. Differential cryptanalysis of des-like cryptosystems. *J. Cryptol.*, v. 4, n. 1, p. 3–72, 1991. Disponível em: <https://doi.org/10.1007/BF00630563>.

33 LAI, X.; MASSEY, J. L.; MURPHY, S. Markov ciphers and differential cryptanalysis. In: DAVIES, D. W. (Ed.). *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*. Springer, 1991. (Lecture Notes in Computer Science, v. 547), p. 17–38. Disponível em: <https://doi.org/10.1007/3-540-46416-6_2>.

34 KNUDSEN, L. R.; ROBSHAW, M. *The Block Cipher Companion*. Springer, 2011. (Information Security and Cryptography). ISBN 978-3-642-17341-7. Disponível em: <https://doi.org/10.1007/978-3-642-17342-4>.

35 MATSUI, M. Linear cryptanalysis method for DES cipher. In: HELLESETH, T. (Ed.). *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*. Springer, 1993. (Lecture Notes in Computer Science, v. 765), p. 386–397. Disponível em: <https://doi.org/10.1007/3-540-48285-7_33>.

36 TARDY-CORFDIR, A.; GILBERT, H. A known plaintext attack of FEAL-4 and FEAL-6. In: FEIGENBAUM, J. (Ed.). *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. Springer, 1991. (Lecture Notes in Computer Science, v. 576), p. 172–181. Disponível em: <https://doi.org/10.1007/3-540-46766-1_12>.

37 WALLÉN, J. On the differential and linear properties of addition. *Master's thesis, Helsinki University of Technology, Laboratory for Theoretical Computer Science*, 2003.

38 CANTEAUT, A. Lecture notes on cryptographic boolean functions. *Inria, Paris, France*, 2016.

39 LANGFORD, S. K.; HELLMAN, M. E. Differential-linear cryptanalysis. In: DESMEDT, Y. (Ed.). *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*. Springer, 1994. (Lecture Notes in Computer Science, v. 839), p. 17–25. Disponível em: <https://doi.org/10.1007/3-540-48658-5_3>.

40 BLONDEAU, C.; LEANDER, G.; NYBERG, K. Differential-linear cryptanalysis revisited. *J. Cryptol.*, v. 30, n. 3, p. 859–888, 2017. Disponível em: <https://doi.org/10.1007/s00145-016-9237-5>.

41 LIPMAA, H.; MORIAI, S. Efficient algorithms for computing differential properties of addition. In: MATSUI, M. (Ed.). *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*. Springer, 2001. (Lecture Notes in Computer Science, v. 2355), p. 336–350. Disponível em: <https://doi.org/10.1007/3-540-45473-X_28>.

42 WALLÉN, J. Linear approximations of addition modulo 2^n. In: JOHANSSON, T. (Ed.). *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*. Springer, 2003. (Lecture Notes in Computer Science, v. 2887), p. 261–273. Disponível em: <https://doi.org/10.1007/978-3-540-39887-5_20>.

43 FORRÉ, R. The strict avalanche criterion: Spectral properties of boolean functions and an extended definition. In: GOLDWASSER, S. (Ed.). *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*. Springer, 1988. (Lecture Notes in Computer Science, v. 403), p. 450–468. Disponível em: <https://doi.org/10.1007/0-387-34799-2_31>.

44 ZHANG, X.-M.; ZHENG, Y. Gac—the criterion for global avalanche characteristics of cryptographic functions. In: *J. UCS The Journal of Universal Computer Science*. [S.l.]: Springer, 1996. p. 320–337.

45 DAWSON, E.; GUSTAFSON, H.; PETTITT, A. N. Strict key avalanche criterion. *Australas. J Comb.*, v. 6, p. 147–154, 1992. Disponível em: <http://ajc.maths.uq.edu.au/pdf/6/ocr-ajc-v6-p147.pdf>.

46 DINU, D. et al. Design strategies for ARX with provable bounds: Sparx and LAX. In: CHEON, J. H.; TAKAGI, T. (Ed.). *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*. [s.n.],

2016. (Lecture Notes in Computer Science, v. 10031), p. 484–513. Disponível em: <https://doi.org/10.1007/978-3-662-53887-6_18>.

47 AUMASSON, J.-P. et al. Sha-3 proposal blake. *Submission to NIST*, v. 92, 2008.

48 FIPS, N. 180-2: Secure hash standard (shs). *US Department of Commerce, National Institute of Standards and Technology (NIST)*, 2012.

49 ADAMS, C. M.; TAVARES, S. E. The structured design of cryptographically good s-boxes. *J. Cryptol.*, v. 3, n. 1, p. 27–41, 1990. Disponível em: <https://doi.org/10.1007/BF00203967>.

50 WU, C. K.; FENG, D. *Boolean Functions and Their Applications in Cryptography*. Springer, 2016. (Advances in Computer Science and Technology). ISBN 978-3-662-48863-8. Disponível em: <https://doi.org/10.1007/978-3-662-48865-2>.

51 BERNSTEIN, D.; LANGE, T. Ecrypt benchmarking of cryptographic systems. In: CHES 2009 WORKSHOP BENCHMARKING CRYPTOGRAPHIC HARDWARE. [S.l.], 2009.

52 COPPERSMITH, D. The data encryption standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, v. 38, n. 3, p. 243–250, 1994. Disponível em: <https://doi.org/10.1147/rd.383.0243>.

53 PRENEEL, B.; GOVAERTS, R.; VANDEWALLE, J. Differential cryptanalysis of hash functions based on block ciphers. In: DENNING, D. E. et al. (Ed.). *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. ACM, 1993. p. 183–188. Disponível em: <https://doi.org/10.1145/168588.168611>.

54 LAI, X. Higher order derivatives and differential cryptanalysis. In: *Communications and cryptography*. [S.l.]: Springer, 1994. p. 227–233.

55 KNUDSEN, L. R. Truncated and higher order differentials. In: PRENEEL, B. (Ed.). *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*. Springer, 1994. (Lecture Notes in Computer Science, v. 1008), p. 196–211. Disponível em: <https://doi.org/10.1007/3-540-60590-8_16>.

56 KNUDSEN, L. DEAL-a 128-bit block cipher. *complexity*, Citeseer, v. 258, n. 2, p. 216, 1998.

57 BIHAM, E.; BIRYUKOV, A.; SHAMIR, A. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: STERN, J. (Ed.). *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. Springer, 1999. (Lecture Notes in Computer Science, v. 1592), p. 12–23. Disponível em: <https://doi.org/10.1007/3-540-48910-X_2>.

58 WAGNER, D. A. The boomerang attack. In: KNUDSEN, L. R. (Ed.). *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*. Springer, 1999. (Lecture Notes in Computer Science, v. 1636), p. 156–170. Disponível em: <https://doi.org/10.1007/3-540-48519-8_12>.

59 KELSEY, J.; KOHNO, T.; SCHNEIER, B. Amplified boomerang attacks against reduced-round MARS and serpent. In: SCHNEIER, B. (Ed.). *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*. Springer, 2000. (Lecture Notes in Computer Science, v. 1978), p. 75–93. Disponível em: <https://doi.org/10.1007/3-540-44706-7_6>.

60 BIHAM, E.; DUNKELMAN, O.; KELLER, N. The rectangle attack - rectangling the serpent. In: PFITZMANN, B. (Ed.). *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*. Springer, 2001. (Lecture Notes in Computer Science, v. 2045), p. 340–357. Disponível em: <https://doi.org/10.1007/3-540-44987-6_21>.

61 DUNKELMAN, O. et al. The retracing boomerang attack. In: CANTEAUT, A.; ISHAI, Y. (Ed.). *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*. Springer, 2020. (Lecture Notes in Computer Science, v. 12105), p. 280–309. Disponível em: <https://doi.org/10.1007/978-3-030-45721-1_11>.

62 COUTINHO, M.; JÚNIOR, R. T. de S.; BORGES, F. Continuous diffusion analysis. *IEEE Access*, v. 8, p. 123735–123745, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3005504>.

63 BEIERLE, C. et al. Schwaemm and Esch: lightweight authenticated encryption and hashing using the Sparkle permutation family. NIST, 2019.

64 DEY, S.; ROY, T.; SARKAR, S. Revisiting design principles of salsa and chacha. *Adv. Math. Commun.*, v. 13, n. 4, p. 689–704, 2019. Disponível em: <https://doi.org/10.3934/amc.2019041>.

65 DING, L. Improved related-cipher attack on salsa20 stream cipher. *IEEE Access*, v. 7, p. 30197–30202, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2892647>.

66 CASTRO, J. C. H.; ESTÉVEZ-TAPIADOR, J. M.; QUISQUATER, J. On the salsa20 core function. In: NYBERG, K. (Ed.). *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*. Springer, 2008. (Lecture Notes in Computer Science, v. 5086), p. 462–469. Disponível em: <https://doi.org/10.1007/978-3-540-71039-4_29>.

67 ISHIGURO, T.; KIYOMOTO, S.; MIYAKE, Y. Latin dances revisited: New analytic results of salsa20 and chacha. In: QING, S. et al. (Ed.). *Information and Communications Security - 13th International Conference, ICICS 2011, Beijing, China, November 23-26, 2011. Proceedings*. Springer, 2011. (Lecture Notes in Computer Science, v. 7043), p. 255–266. Disponível em: <https://doi.org/10.1007/978-3-642-25243-3_21>.

68 MOUHA, N.; PRENEEL, B. A proof that the ARX cipher salsa20 is secure against differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, v. 2013, p. 328, 2013. Disponível em: <http://eprint.iacr.org/2013/328>.

69 TSUNOO, Y. et al. Differential cryptanalysis of Salsa20/8. In: *Workshop Record of SASC*. [S.l.: s.n.], 2007. v. 28.

70 COUTINHO, M.; NETO, T. S. New multi-bit differentials to improve attacks against ChaCha. *IACR Cryptol. ePrint Arch.*, v. 2020, p. 350, 2020.

71 AUMASSON, J.; BERNSTEIN, D. J. Siphash: A fast short-input PRF. In: GALBRAITH, S. D.; NANDI, M. (Ed.). *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*. Springer, 2012. (Lecture Notes in Computer Science, v. 7668), p. 489–508. Disponível em: <https://doi.org/10.1007/978-3-642-34931-7_28>.

72 COUTINHO, M. et al. Improving the security of chacha against differential-linear cryptanalysis. 2020.

73 GOHR, A. Improving attacks on round-reduced speck32/64 using deep learning. In: BOLDYREVA, A.; MICCIANCIO, D. (Ed.). *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*. Springer, 2019. (Lecture Notes in Computer Science, v. 11693), p. 150–179. Disponível em: <https://doi.org/10.1007/978-3-030-26951-7_6>.

74 ARANHA, D. F.; GOUVêA, C. P. L. *RELIC is an Efficient LIbrary for Cryptography*. 2013. <https://github.com/relic-toolkit/relic>.

A APPENDIX

A.1 LIBRARY

Is this section , we briefly describe our library called `libfpco` that implements continuous generalizations (available at <https://github.com/MurCoutinho/pda>).

A.1.1 Scope and goals

The goal of the proposed library is to implement all continuous generalizations developed in this thesis. Furthermore, the library should provide an easy API such that a programmer can implement a generalization of common symmetric cryptographic algorithms.

The API follows the convention that the name of the generalized operation will be:

```
"fpco_" + name of the function
```

For example, the continuous generalization of the XOR function is denoted `fpco_xor`.

To implement this API we require several internal functions. In particular, we require the implementation of the probabilistic operations and a set of structures and methods implementing a small number arithmetic. In the following sections, we present all theses in greater details.

A.1.2 Internal functions

A.1.2.1 Small numbers arithmetic

When executing a cryptographic algorithm using continuous generalizations the continuous deviation computed tends to get really small. This happens because the continuous operations are based on probability theory, therefore when a continuous deviation different than -1 and than 1 is defined as input it makes that uncertainty about each bit increases following the confusion and diffusion of the algorithm, i.e., the continuous generalization gets closer and closer to zero.

Hence, we need a way to work with very small numbers. For example, a `double` in `C` language may not have the necessary precision converging to zero as the number of rounds progresses. To do so, in `libfpco` we created a structure defining a small number:

```
typedef struct {
    double mantissa;
    bn_t exponent;
} sn_st;
```

Here, a small number is a number in the form $m * 10^{-y}$, where m is the mantissa and y the exponent. Also, $-1 < mantissa < 1$ and $exponent$ is represented as a big number. The purpose of `sn_st` is to create the possibility of tracking a number that is rapidly approaching zero. It is not the purpose of `sn_st` to be a double with more precision or decimal places.

There are a series of functions to work with a small number. Bellow we list all of them:

- `sn_print`: Prints a small number.
- `sn_print_array`: Prints a small number array.
- `sn_new`: Creates a new small number.
- `sn_new_array`: Creates a new small number array.
- `sn_free`: Frees the memory and destroy a small number.
- `sn_free_array`: Frees a small number array.
- `sn_cpy`: Copies small number.
- `sn_cpy_array`: Copies small number array.
- `sn_cmp`: Returns the result of a signed comparison between two small numbers.
- `sn_cmp_array`: Compares two small number arrays.
- `sn_cmp_magnitude`: Returns the result of a comparison between the magnitude or scale of two small number.
- `sn_cmp_magnitude_array`: Returns the result of a comparison between the magnitude or scale of two small number arrays.
- `sn_update_range`: Updates the range of the small number in such a way that sets -1<mantissa<1.
- `sn_rand`: Generates a random small number.
- `sn_rand_array`: Generates a random small number array.
- `sn_to_double`: Converts a small number to a double.
- `sn_to_double_array`: Converts a small number array to a double array.
- `double_to_sn`: Converts a double to a small number.

- `double_to_sn_array`: Converts a double array to a small number array.
- `sn_mul`: Multiplies two small numbers.
- `sn_add`: Adds two small numbers.
- `sn_abs`: Absolute value of small number.
- `sn_abs_array`: Absolute value of small number array.

It is important to note that the exponent is a big number that we implemented using the RELIC toolkit [74]. For example, below we present the code for the function `sn_add`:

```
void sn_add(sn_t c, sn_t a, sn_t b)
{
    sn_st **max, **min;
    sn_t aux;
    if(a->mantissa == 0)
    {
        sn_cpy(c, b);
        return;
    }
    if(b->mantissa == 0)
    {
        sn_cpy(c, a);
        return;
    }
    if (bn_cmp(a->exponent, b->exponent)==CMP_GT)
    {
        max = &a;
        min = &b;
    }
    else
    {
        max = &b;
        min = &a;
    }
    sn_new(aux);
    bn_sub(aux->exponent, (*max)->exponent, (*min)->exponent);
    if (bn_cmp(aux->exponent, fifteen)!= CMP_GT)
    {
        sn_cpy(aux, (*min));
        while (bn_cmp((*max)->exponent, aux->exponent)==CMP_GT)
        {
            aux->mantissa *= 0.1;
            bn_add(aux->exponent,aux->exponent,one);
        }
        aux->mantissa += (*max)->mantissa;
        sn_cpy(c, aux);
    }
    else
    {
        sn_cpy(c, (*max));
    }
    sn_free(aux);
}
```

A.1.2.2 Probabilistic operations

Using the small number arithmetic of the previous section, we implemented some useful functions to work with continuous deviations. In the library we implemented these operations with the alias `correlation`, thus here, we consider correlation as a synonym of the continuous deviation. In the following, we list such functions:

- `correlation_rand`: Generates a random correlation.
- `correlation_rand_array`: Generates a random correlation array.
- `correlation_rand_max`: Generates a random maximum correlation (1 or -1).
- `correlation_rand_max_array`: Generates a random maximum correlation array (1 or -1).
- `correlation_correct_boundaries`: Corrects the correlation to be between -1 and 1.
- `correlation_max`: Computes the max, in absolute terms, of two correlations.
- `prob_to_correlation`: Computes converts a probability (double) to a correlation (sn).
- `prob_to_correlation_array`: Computes converts a probability array (double) to a correlation array (sn).
- `correlation_to_prob`: Computes converts a correlation (sn) a probability (double).
- `correlation_to_prob_array`: Computes converts a correlation array (sn) a probability array (double).
- `prob_mult_through_correlation`: Multiply two probabilities represented as correlation.

- `prob_sum_through_correlation`: Adds two probabilities represented as correlation.
- `prob_complement_through_correlation`: Complements a probability represented as correlation.
- `correlation_array_to_u{8 or 16 or 32 or 64}`: Converts a integer (u8, u16, u32 or u64) to a correlation array.
- `u{8 or 16 or 32 or 64}_to_correlation_array`: Converts correlation array to a integer (u8, u16, u32 or u64).

In particular, we should explain the behaviors of the functions:

- `prob_mult_through_correlation`
- `prob_sum_through_correlation`
- `prob_complement_through_correlation`

First, let us look at the source code:

```
void prob_mult_through_correlation(sn_t c, sn_t a, sn_t b)
{
    sn_t aux;

    sn_new(aux);

    sn_mul(aux, a, b);
    sn_add(aux, aux, a);
    sn_add(aux, aux, b);
    sn_mul(aux, aux, half);
    sn_add(aux, aux, mhalf);

    sn_cpy(c, aux);
    correlation_correct_boundaries(c);

    sn_free(aux);
}


void prob_sum_through_correlation(sn_t c, sn_t a, sn_t b)
{
    sn_add(c, a, b);
    sn_add(c, c, snone);
    correlation_correct_boundaries(c);
}
```

```
void prob_complement_through_correlation(sn_t c, sn_t a)
{
    sn_cpy(c, a);
    c->mantissa = -c->mantissa;
}
```

These functions denote operations that are being performed thinking about probabilities, however, computing mathematically through the correlations. More concretely, let p_i be a probability and ε_i be its correlation, i.e., $p_i = (\varepsilon_i + 1)/2$ for $i = 1, 2, 3$. If we want to obtain the correlation ε_3 resulting from the multiplication $p_3 = p_1 p_2$, but we are given ε_1 and ε_2, then we can do it directly by applying the formula:

$$\varepsilon_3 = \frac{\varepsilon_1 \varepsilon_2 + \varepsilon_1 + \varepsilon_2}{2} - \frac{1}{2}.$$

This result is easy to verify by simply converting the equation $p_3 = p_1 p_2$ in terms of the correlations. This is the equation being computed inside of `prob_mult_through_correlation`.

Similarly, `prob_sum_through_correlation` is derived from the equation $p_3 = p_1 + p_2$, and `prob_complement_through_correlation` is derived from the equation $p_3 = 1 - p_1$.

A.1.3 Main API

Finally, the main API is given by the generalization of all mathematical operations presented in Section 4.3. In `libfpco`, we call these Fundamental Probabilistic Cryptographic Operations (FPCO). More precisely, we have the following functions.

- `fpco_core_init`: Initializes the use of the fpco operations.
- `fpco_core_finalize`: Finalizes the use of the fpco operations.
- `fpco_rot_right`: Computes the continuous generalization of Rotate right (see Definition 4.9).
- `fpco_rot_left`: Computes the continuous generalization of Rotate left (see Definition 4.9).
- `fpco_shift_right`: Computes the continuous generalization of Shift right (see Definition 4.9).

- `fpco_shift_left`: Computes the continuous generalization of Shift left (see Definition 4.9).
- `fpco_xor`: Computes the continuous generalization of XOR (see Definition 4.4).
- `fpco_and`: Computes the continuous generalization of AND (see Definition 4.5).
- `fpco_or`: Computes the continuous generalization of OR (see Definition 4.6).
- `fpco_not`: Computes the continuous generalization of NOT (see Definition 4.7).
- `fpco_add`: Computes the continuous generalization of addition (see Definition 4.10).
- `fpco_sub`: Computes the continuous generalization of subtraction (see Definition 4.10).
- `fpco_boolean_function`: Computes the continuous generalization of a generic Boolean function (see Theorem (4.1)).
- `fpco_sbox`: Computes the continuous generalization of a SBOX (see Definition 4.12).

In the following, we present the implementation of `fpco_add`, `fpco_sbox` and `fpco_boolean_fu` as examples.

```
int fpco_add(sn_t *c, sn_t *a, sn_t *b, int size)
{
    if(size>global_max_size_of_bias_arrays)
        return 1;
    if(global_max_size_of_bias_arrays<7)
        return -1;

    //aux[0] - used for the carry bit
    //aux[1] - retains the resulting bit
    //aux[2] - used for intermediary values
    sn_cpy(aux[0], snminusone);

    for (int i = 0; i<size; i++)
    {
        sn_mul(aux[1], a[i], b[i]);
        sn_mul(aux[1], aux[1], aux[0]);

        prob_complement_through_bias(aux[5], aux[1]);
        sn_cpy(aux[4], aux[0]);
        sn_cpy(aux[3], a[i]);
        sn_cpy(aux[2], b[i]);
        sort_for_secure_sum();
        sn_add(aux[2], aux[2], aux[3]);
        sn_add(aux[2], aux[2], aux[4]);
        sn_add(aux[2], aux[2], aux[5]);

        sn_mul(aux[2], aux[2], half);

        bias_correct_boundaries(aux[2]);
        bias_correct_boundaries(aux[1]);

        sn_cpy(c[i], aux[1]); //to allow c as the same pointer of a or b
        sn_cpy(aux[0], aux[2]);
    }
    return 0;
}
```

```
int fpco_boolean_function(sn_t out, sn_t *in, unsigned char *truth_table,
     int nbits)
{
    unsigned char flag;
    out->mantissa = 0;
    for(int x=0; x<pow(2,nbits); x++)
    {
        flag = 0;
        if(truth_table[x])
        {
            aux[0]->mantissa = 1;
            for(int i=0; i<nbits; i++)
            {
                if((x>>i)&1)
                {
                    if(in[i]->mantissa == -1)
                    {
                        flag = 1;
                        break;
                    }
                    aux[1]->mantissa = 1 + in[i]->mantissa;
                }
                else
                {
                    if(in[i]->mantissa == 1)
                    {
                        flag = 1;
                        break;
                    }
                    aux[1]->mantissa = 1 - in[i]->mantissa;
                }
                sn_mul(aux[0], aux[0], aux[1]);
            }
            if(flag)
                continue;
            out->mantissa += aux[0]->mantissa;
        }
    }
    out->mantissa = out->mantissa*pow(2, -(nbits-1)) -1;
    return 0;
}
```

```
int fpco_sbox(sn_t *out, sn_t *in, int *sbox, int nbits)
{
    unsigned char *truth_table = NULL;
    truth_table = (unsigned char *) malloc(sizeof(unsigned char) * pow(2,
   nbits));

    if(nbits>global_max_size_of_bias_arrays-2)
        return 1;

    sn_cpy_array(&aux[2], in, nbits);
    for(int i=0;i<nbits;i++)
    {
        for(int j=0;j<pow(2,nbits); j++)
            truth_table[j] = (sbox[j]>>i)&1;

        fpco_boolean_function(out[i], &aux[2], truth_table, nbits);
    }

    free(truth_table);
    return 0;
}

\subsection{Challenges}

\rr{}{The implementation of the library was not always straightforward.
   That is because we have some approximation problems when dealing with
   these small numbers. For example, when computing the diffusion factor,
    the values converge very fast to zero. In such case, it might happen
   that a value equal to 1 or -1 also converges to zero. But in this case
    these bits should behave as discrete bits. To avoid such cases we our
    code present some implementations that are not necessarily trivial.}{
   Comentario do Julio para que se apresentasse as dificuldade
   enfrentadas na implementacao}
```

A.2 PROOFS

For completeness, in this section, we prove several Lemmas from this thesis.

A.2.1 Proofs for Lemma 6.9

Eq. (6.16)

proof.

Using Eqs. (6.6) and (6.7), we can write

$$\begin{aligned} x_{b,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = & \mathcal{L}_{b,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \\ & \mathcal{L}_{c,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_c^{(m-1)}, x_d'^{(m-1)}). \end{aligned}$$

Using the approximation of Eq. (6.14), we can write $\Theta_i(x_c^{(m-1)}, x_d'^{(m-1)}) = x_{d,i-1}'^{(m-1)}$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Thus, using Eq. (6.4) and canceling out common factors we get

$$x_{b,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)},$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$, which concludes the proof. ■

Eqs. (6.17) **and** (6.18)

proof.

Using Eqs. (6.6) and (6.9), we can write

$$\begin{aligned} x_{a,i}^{(m-1)} \oplus x_{b,i}^{(m-1)} = & \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \\ & \Theta_i(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}). \end{aligned}$$

Cancelling out common factors, using the approximation of Eq. (6.14) and the Piling-up Lemma we can write

$$x_{a,i}^{(m-1)} \oplus x_{b,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus x_{b,i-1}'^{(m-1)} \oplus x_{b,i-1}^{(m-1)}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Now, we can replace $x_{b,i-1}'^{(m-1)}$ using Eq. (6.2) and $x_{b,i-1}^{(m-1)}$ using Lemma 6.3, which leads to

$$x_{a,i}^{(m-1)} \oplus x_{b,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus x_{b,i+6}^{(m)} \oplus x_{c,i-1}^{(m)} \oplus \mathcal{L}_{b,i-1}^{(m)} \oplus x_{d,i-2}^{(m)},$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$ by the Piling-up Lemma. We can also use Lemma 6.1 to obtain

$$x_{a,1}^{(m-1)} \oplus x_{b,1}^{(m-1)} = \mathcal{L}_{a,1}^{(m)} \oplus \mathcal{L}_{b,1}^{(m)} \oplus x_{b,7}^{(m)} \oplus x_{c,0}^{(m)} \oplus \mathcal{L}_{b,0}^{(m)},$$

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

Eqs. (6.19) **and** (6.20)

proof.

Combining Eq. (6.7) and Eq. (6.9), we have

$$\begin{aligned} x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = {} & \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \Theta_i(x_c^{(m-1)}, x_d^{\prime(m-1)}) \oplus \\ & \Theta_i(x_a^{\prime(m-1)}, x_b^{\prime(m-1)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}). \end{aligned}$$

Using the approximation of Eq. (6.14) and the Piling-up Lemma, we can write

$$x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{d,i-1}^{\prime(m-1)} \oplus x_{b,i-1}^{\prime(m-1)} \oplus x_{b,i-1}^{(m-1)}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$. Now, we can replace $x_{d,i-1}^{\prime(m-1)}$ using Eq. (6.4), $x_{b,i-1}^{\prime(m-1)}$ using Eq. (6.2) and $x_{b,i-1}^{(m-1)}$ using Lemma 6.3 if $i > 1$ or 6.1 if $i = 1$, which leads to

$$\begin{aligned} x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = {} & \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)} \oplus x_{b,i+6}^{(m)} \\ & \oplus x_{c,i-1}^{(m)} \oplus \mathcal{L}_{b,i-1}^{(m)} \oplus x_{d,i-2}^{(m)}, \end{aligned}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^4}\right)$ by the Piling-up Lemma or

$$\begin{aligned} x_{a,1}^{(m-1)} \oplus x_{c,1}^{(m-1)} = {} & \mathcal{L}_{a,1}^{(m)} \oplus \mathcal{L}_{c,1}^{(m)} \oplus x_{a,0}^{(m)} \oplus x_{d,8}^{(m)} \oplus x_{b,7}^{(m)} \\ & \oplus x_{c,0}^{(m)} \oplus \mathcal{L}_{b,0}^{(m)}, \end{aligned}$$

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

Eq. (6.21)

proof.

Using Eq. (6.8) and Eq. (6.9), we can write

$$x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}).$$

Using Eq. (6.14), we get

$$x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus x_{b,i-1}^{(m-1)},$$

and from Eq. (6.6)

$$x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \mathcal{L}_{b,i-1}^{(m)} \oplus \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}),$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Thus, using the approximation of Eq. (6.15) and the Piling-up Lemma, we can write

$$x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \mathcal{L}_{b,i-1}^{(m)},$$

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

Eq. (6.22)

proof.

Using Eq. (6.9) and Eq. (6.7), and canceling out common factors we get

$$\begin{aligned} x_{a,i-1}^{(m-1)} \oplus x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \; & \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \\ & \Theta_{i-1}(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}) \oplus \\ & \Theta_{i-1}(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \Theta_i(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \\ & \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \Theta_i(x_c^{(m-1)}, x_d'^{(m-1)}) \end{aligned}$$

Using the approximation of Eq. (6.15) and the Piling-up Lemma we obtain

$$\begin{aligned} x_{a,i-1}^{(m-1)} \oplus x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \; & \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \\ & \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_c^{(m-1)}, x_d'^{(m-1)}) \end{aligned}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Using Eq. (6.14) and Eq. (6.4) we get

$$\begin{aligned} x_{a,i-1}^{(m-1)} \oplus x_{a,i}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \; & \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \\ & x_{d,i-2}^{(m)} \oplus x_{a,i-1}^{(m)} \oplus x_{d,i+7}^{(m)} \end{aligned}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^4}\right)$. ■

Eq. (6.23)

proof.

Using Eq. (6.6) and Eq. (6.9), and canceling out common factors we can write

$$x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \\ \Theta_{i-1}(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}) \oplus \Theta_{i-1}(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \\ \Theta_i(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}).$$

Using the approximation of Eq. (6.15) and the Piling-up Lemma, we can write

$$\begin{aligned} x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} = \; & \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \\ & \oplus \mathcal{L}_{b,i}^{(m)} \oplus \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}). \end{aligned}$$

with probability $\frac{1}{2}\left(1 + \frac{1}{2^2}\right)$. Using the approximation of Eq. (6.14) we get

$$x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus x_{d,i-2}^{(m)}.$$

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

Eq. (6.24)

proof.

Using Eq. (6.8) and Eq. (6.9), and canceling out common factors we have

$$\begin{aligned} x_{b,i-1}^{(m-1)} \oplus x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \; & x_{b,i-1}^{(m-1)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \\ & \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \mathcal{L}_{d,i}^{(m)}. \end{aligned}$$

Using the approximation of Eq. (6.14), we have $\Theta_i(x_a^{(m-1)}, x_b^{(m-1)}) = x_{b,i-1}^{(m-1)}$ occurring with probability $\frac{1}{2}\left(1 + \frac{1}{2^2}\right)$. Thus,

$$x_{b,i-1}^{(m-1)} \oplus x_{a,i}^{(m-1)} \oplus x_{d,i}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}).$$

with probability $\frac{1}{2}\left(1 + \frac{1}{2}\right)$. Finally, using the approximation of Eq. (6.14) and the Piling-up Lemma, we get

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

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

Eq. (6.25)

proof.

Using Eq. (6.6) and Eq. (6.7), we can write

$$x_{b,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{b,i-1}^{(m)} \oplus \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}) \oplus \mathcal{L}_{b,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \Theta_{i-1}(x_c'^{(m-1)}, x_d^{(m)}) \oplus \Theta_{i-1}(x_c^{(m-1)}, x_d'^{(m-1)}) \oplus \mathcal{L}_{c,i}^{(m)} \oplus \Theta_i(x_c'^{(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_c^{(m-1)}, x_d'^{(m-1)}).$$

Canceling out common factors we get

$$\begin{aligned} x_{b,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \;& \mathcal{L}_{b,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)} \oplus \\ & \Theta_{i-1}(x_c^{(m-1)}, x_d'^{(m-1)}) \oplus \\ & \Theta_i(x_c^{(m-1)}, x_d'^{(m-1)}). \end{aligned}$$

Thus, using the approximation of Eq. (6.15) we get

$$x_{b,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} \oplus x_{c,i}^{(m-1)} = \mathcal{L}_{b,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \mathcal{L}_{c,i}^{(m)}.$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. ■

Eq. (6.26)

proof.

Using equations (6.6), (6.7), and (6.9)

$$x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \Theta_i(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \Theta_{i-1}(x_a'^{(m-1)}, x_b'^{(m-1)}) \oplus \Theta_{i-1}(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \Theta_{i-1}(x_c^{(m-1)}, x_d'^{(m-1)}).$$

Using the approximation of Eq. (6.15) and the Piling-up Lemma, we can write

$$\begin{aligned} x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = \;& \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \\ & \Theta_{i-1}(x_c^{(m-1)}, x_d'^{(m-1)}). \end{aligned}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Therefore, equations (6.14) and (6.4) give us

$$\begin{aligned} x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{b,i}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} = \;& \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{b,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \\ & x_{a,i-2}^{(m)} \oplus x_{d,i+6}^{(m)}. \end{aligned}$$

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

Eq. (6.27)

proof.

Using equations (6.7), (6.8), and (6.9), we can write

$$x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} \oplus x_{d,i}^{(m-1)} \oplus x_{d,i-1}^{(m-1)} = \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \mathcal{L}_{d,i-1}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \Theta_{i-1}(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_i(x_a^{(m-1)}, x_b^{(m-1)}) \oplus \Theta_{i-1}(x_c^{(m-1)}, x_d^{\prime(m-1)}).$$

Using the approximation of Eq. (6.15), we have

$$x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} \oplus x_{d,i}^{(m-1)} \oplus x_{d,i-1}^{(m-1)} = \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \mathcal{L}_{d,i-1}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus \Theta_i(x_c^{\prime(m-1)}, x_d^{(m)}) \oplus \Theta_{i-1}(x_c^{(m-1)}, x_d^{\prime(m-1)})$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Finally, by the Piling-up Lemma and using the approximation of Eq. (6.14) and Eq. (6.4), we get

$$x_{a,i}^{(m-1)} \oplus x_{a,i-1}^{(m-1)} \oplus x_{c,i-1}^{(m-1)} \oplus x_{d,i}^{(m-1)} \oplus x_{d,i-1}^{(m-1)} = \mathcal{L}_{a,i-1}^{(m)} \oplus \mathcal{L}_{a,i}^{(m)} \oplus \mathcal{L}_{c,i-1}^{(m)} \oplus \mathcal{L}_{d,i-1}^{(m)} \oplus \mathcal{L}_{d,i}^{(m)} \oplus x_{d,i-1}^{(m)} \oplus x_{a,i-2}^{(m)} \oplus x_{d,i+6}^{(m)}$$

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

A.2.2 Proof of Lemma 6.10

proof.

First, from Eq. (6.28), we can use Lemma 6.1 to replace $x_{1,0}^{(5)}$, $x_{3,0}^{(5)}$, $x_{9,0}^{(5)}$, $x_{13,0}^{(5)}$, $x_{15,0}^{(5)}$ by $\mathcal{L}_{1,0}^{(6)}$, $\mathcal{L}_{3,0}^{(6)}$, $\mathcal{L}_{9,0}^{(6)}$, $\mathcal{L}_{13,0}^{(6)}$, $\mathcal{L}_{15,0}^{(6)}$ with probability 1. Next, note that, since we are transitioning from round 5 to 6, we have

$$(a,b,c,d) \in \{(0,5,10,15),(1,6,11,12),(2,7,8,13),(3,4,9,14)\}.$$

We have already considered the case $(a,b,c,d) = (0,5,10,15)$. Then we still have 3 cases left to consider.

- **Case 1:** When $(a,b,c,d) = (1,6,11,12)$, we have the factors $x_{11,12}^{(5)}$, $x_{12,6}^{(5)}$, $x_{12,7}^{(5)}$. Then we can use Lemma 6.3 and Lemma 6.7 to get

$$\Pr\left(x_{11,12}^{(5)} = \mathcal{L}_{11,12}^{(6)} \oplus x_{1,11}^{(6)} \oplus x_{12,19}^{(6)} \oplus x_{12,11}^{(6)}\right) = \frac{1}{2}\left(1+\frac{1}{2^2}\right)$$

  and

$$\Pr\left(x_{12,7}^{(5)} \oplus x_{12,6}^{(5)} = \mathcal{L}_{12,7}^{(6)} \oplus \mathcal{L}_{12,6}^{(6)}\right) = \frac{1}{2}\left(1+\frac{1}{2}\right).$$

- **Case 2:** If $(a,b,c,d) = (2,7,8,13)$, we have the factors $x^{(5)}_{7,7}$, $x^{(5)}_{7,19}$, $x^{(5)}_{8,7}$, $x^{(5)}_{8,19}$, $x^{(5)}_{13,8}$ and we can use Lemma 6.3 and Eq. (6.16) of Lemma 6.9 to get

$$\Pr\left(x^{(5)}_{13,8} = \mathcal{L}^{(6)}_{13,8} \oplus x^{(6)}_{8,7} \oplus x^{(6)}_{7,14}\right) = \frac{1}{2}\left(1+\frac{1}{2}\right),$$

$$\Pr\left(x^{(5)}_{7,7} \oplus x^{(5)}_{8,7} = \mathcal{L}^{(6)}_{7,7} \oplus \mathcal{L}^{(6)}_{8,7} \oplus x^{(6)}_{2,6} \oplus x^{(6)}_{13,14}\right) = \frac{1}{2}\left(1+\frac{1}{2}\right),$$

$$\Pr\left(x^{(5)}_{7,19} \oplus x^{(5)}_{8,19} = \mathcal{L}^{(6)}_{7,19} \oplus \mathcal{L}^{(6)}_{8,19} \oplus x^{(6)}_{2,18} \oplus x^{(6)}_{13,26}\right) = \frac{1}{2}\left(1+\frac{1}{2}\right).$$

- **Case 3:** When considering $(a,b,c,d) = (3,4,9,14)$, we have $x^{(5)}_{4,26}$, and we can use Lemma 6.3 to obtain

$$\Pr\left(x^{(5)}_{4,26} = \mathcal{L}^{(6)}_{4,26} \oplus x^{(6)}_{14,26}\right) = \frac{1}{2}\left(1+\frac{1}{2}\right).$$

By the Piling-up Lemma, we have that all these changes result in a probability of $\frac{1}{2}\left(1+\frac{1}{2^8}\right)$. Expanding the linear terms using Eqs. (6.10)-(6.13) and canceling out common factors completes the proof.

■

A.2.3 Proof of Lemma 6.11

proof.

If we start from Lemma 6.10, then we want to expand the equation one more round. To do so, first note that since we are transitioning from round 6 to 7, we have $(a,b,c,d) \in \{(0,4,8,12),(1,5,9,13),(2,6,10,14),(3,7,11,15)\}$. Therefore, we can divide the factors of the equation in 4 distinct groups:

- Group I - $x^{(6)}_0[0,16], x^{(6)}_4[7,13,19], x^{(6)}_8[0,7,8,19,31]$, $x^{(6)}_{12}[0,11,12,19,20,30,31]$.
- Group II - $x^{(6)}_1[0,6,7,11,12,22,23]$, $x^{(6)}_5[7]$, $x^{(6)}_9[0,6,12,26]$, $x^{(6)}_{13}[0,14,15,24,26,27]$.
- Group III - $x^{(6)}_2[0,6,7,8,16,18,19,24], x^{(6)}_6[7,13,14,19]$, $x^{(6)}_{10}[0]$, $x^{(6)}_{14}[8,25,26]$.
- Group IV - $x^{(6)}_7[6,7,14,15,26]$, $x^{(6)}_{11}[6,7], x^{(6)}_{15}[24]$.

The procedure to expand and compute the correlation is similar to that in the proof of Lemma 6.10. To simplify the notation we will compute the probability given by the Piling-up Lemma by summing values k where the probability of a particular linear equation will be given by $\frac{1}{2}\left(1+\frac{1}{2^k}\right)$.

In Group I, the factors $x^{(6)}_{0,0}, x^{(6)}_{8,0}, x^{(6)}_{12,0}$ can be expanded using Lemma 6.1 with probability 1. Next, we can combine the following factors: $x^{(6)}_{4,7}, x^{(6)}_{8,7}, x^{(6)}_{8,8}$ using Lemma 6.8 ($k=2$); $x^{(6)}_{4,19}, x^{(6)}_{8,19}$ using Eq. (6.16) of Lemma 6.9 ($k=1$); $x^{(6)}_{12,11}, x^{(6)}_{12,12}$ using Lemma 6.7 with ($k=1$); $x^{(6)}_{12,19}, x^{(6)}_{12,20}$ using Lemma 6.7 with ($k=1$); $x^{(6)}_{12,30}, x^{(6)}_{12,31}$ using Lemma 6.7 with ($k=1$). Finally, it remains some single terms to be expanded: $x^{(6)}_{0,16}$ using Lemma 6.6 ($k=3$); $x^{(6)}_{4,13}$ using Lemma 6.3 ($k=1$); $x^{(6)}_{8,31}$ using Lemma 6.3 ($k=2$). By the Piling-up Lemma, we can combine these linear relations to obtain

$$\begin{aligned}&x_0^{(6)}[0,16]\oplus x_4^{(6)}[7,13,19]\oplus x_8^{(6)}[0,7,8,19,31]\oplus x_{12}^{(6)}[0,11,12,19,20,30,31]=\\&x_0^{(7)}[0,3,4,7,8,11,12,14,15,18,20,27,28]\oplus x_4^{(7)}[0,2,3,5,18,22,23,27]\oplus\\&x_8^{(7)}[8,11,13,20,25,27,28,30,31]\oplus x_{12}^{(7)}[3,4,6,11,13,22,23,24,26,27,30,31]\end{aligned} \tag{A.1}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^{12}}\right)$.

In Group II, the factors $x^{(6)}_{1,0}, x^{(6)}_{9,0}, x^{(6)}_{13,0}$ can be expanded using Lemma 6.1 with probability 1. Next, we can combine the following factors: $x^{(6)}_{1,6}, x^{(6)}_{1,7}, x^{(6)}_{5,7}, x^{(6)}_{9,6}$ using Eq. (6.26) of Lemma 6.9 ($k=3$); $x^{(6)}_{1,11}, x^{(6)}_{1,12}, x^{(6)}_{9,12}$ using Eq. (6.22) of Lemma 6.9 ($k=4$); $x^{(6)}_{1,22}, x^{(6)}_{1,23}$ using Lemma 6.7 ($k=3$); $x^{(6)}_{13,14}, x^{(6)}_{13,15}$ using Lemma 6.7 ($k=1$); $x^{(6)}_{13,26}, x^{(6)}_{13,27}$ using Lemma 6.7 ($k=1$). Finally, it remains some single terms to be expanded: $x^{(6)}_{9,26}$ using Lemma 6.3 ($k=2$); $x^{(6)}_{13,24}$ using Lemma 6.3 ($k=1$). By the Piling-up Lemma, we can combine these linear relations to obtain

$$\begin{aligned}&x_1^{(6)}[0,6,7,11,12,22,23]\oplus x_5^{(6)}[7]\oplus x_9^{(6)}[0,6,12,26]\oplus x_{13}^{(6)}[0,14,15,24,26,\\&27]=x_1^{(7)}[0,5,7,8,10,11,14,15,16,22,23,24,25,27,30,31]\oplus x_5^{(7)}[1,2,9,10,\\&13,14,18,21,22,25,29,30]\oplus x_9^{(7)}[2,3,6,7,14,15,18,27]\oplus x_{13}^{(7)}[1,2,6,7,8,10,\\&11,13,14,16,18,20,22,23,24,25,26]\end{aligned} \tag{A.2}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^{15}}\right)$.

In Group III, the factors $x^{(6)}_{2,0}$ and $x^{(6)}_{10,0}$ can be expanded using Lemma 6.1 with probability 1. Next, we can combine the following factors: $x^{(6)}_{2,6}, x^{(6)}_{2,7}$ using Lemma 6.7 ($k=3$); $x^{(6)}_{6,13}, x^{(6)}_{6,14}$ using Lemma 6.7 ($k=1$); $x^{(6)}_{14,25}, x^{(6)}_{14,26}$ using Lemma 6.7 ($k=1$); $x^{(6)}_{2,18}, x^{(6)}_{2,19}, x^{(6)}_{6,19}$ using Eq. (6.23) of Lemma 6.9 ($k=3$); $x^{(6)}_{2,8}, x^{(6)}_{6,7}, x^{(6)}_{14,8}$ using Eq. (6.24) of Lemma 6.9 ($k=2$). Finally, it remains some single terms to be expanded: $x^{(6)}_{2,16}$ using Lemma 6.6 ($k=3$); $x^{(6)}_{2,24}$ using Lemma 6.6 ($k=3$). By the Piling-up Lemma, we can combine these linear relations to obtain

$$x_2^{(6)}[0,6,7,8,16,18,19,24] \oplus x_6^{(6)}[7,13,14,19] \oplus x_{10}^{(6)}[0] \oplus x_{14}^{(6)}[8,25,26] = x_2^{(7)}[6,7,9,10,16,18,19,25,26] \oplus x_6^{(7)}[2,3,5,7,10,11,13,14,19,22,23,27,30,31] \oplus x_{10}^{(7)}[0,3,4,8,12,13,14,18,20,27,28,30] \oplus x_{14}^{(7)}[0,6,13,14,15,16,23,24] \tag{A.3}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^{16}}\right)$.

In Group IV, we can combine the following factors: $x_{7,14}^{(6)}, x_{7,15}^{(6)}$ using Lemma 6.7 ($k=1$); $x_{7,6}^{(6)}, x_{7,7}^{(6)}, x_{11,6}^{(6)}, x_{11,7}^{(6)}$ using Eq. (6.25) of Lemma 6.9 ($k=1$). It remains some single terms to be expanded: $x_{7,26}^{(6)}$ using Lemma 6.3 ($k=1$); $x_{15,24}^{(6)}$ using Lemma 6.3 ($k=1$). By the Piling-up Lemma, we can combine these linear relations to obtain

$$x_7^{(6)}[6,7,14,15,26] \oplus x_{11}^{(6)}[6,7] \oplus x_{15}^{(6)}[24] = x_3^{(7)}[6,7,8,24] \oplus x_7^{(7)}[1,2,13,25,26,30,31] \oplus x_{11}^{(7)}[6,14,15,18,19,23,24,27] \oplus x_{15}^{(7)}[16,25,26] \tag{A.4}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^{4}}\right)$.

Finally, using the Piling-up Lemma, we can combine the results from Lemma 6.10 and Eqs. (A.1)-(A.4), which leads to a correlation of $\varepsilon_L = 1/2^{8+12+15+16+4} = 2^{-55}$. $\square$

■

A.2.4 Proof of Lemma 7.4

proof.

If we start from Lemma 7.3, then we want to expand the equation one more round. To do so, first note that since we are transitioning from round 7 to 8, we have $(a,b,c,d) \in \{(0,1,2,3),(5,6,7,4),(10,11,8,9),(15,12,13,14)\}$. Therefore, we can divide the factors of the equation in 4 distinct groups:

- Group I - $x_0^{(7)}[0] \oplus x_2^{(7)}[12,13] \oplus x_3^{(7)}[17]$.
- Group II - $x_4^{(7)}[7,18,19] \oplus x_6^{(7)}[25,26] \oplus x_7^{(7)}[26,31]$.
- Group III - $x_8^{(7)}[13,14,19] \oplus x_{11}^{(7)}[31]$.
- Group IV - $x_{12}^{(7)}[0,14] \oplus x_{14}^{(7)}[12,13] \oplus x_{15}^{(7)}[16,17]$.

The procedure to expand and compute the correlation is similar to that in the proof of Lemma 7.3, expanding adjacent pairs with Lemma 7.2 and the rest individually with Lemma 7.1. To simplify the notation, we compute the probability given by the Piling-up Lemma by summing values k where the probability of a particular linear equation is given by $\frac{1}{2}\left(1 \pm \frac{1}{2^k}\right)$.

For Group I, we expand $x^{(7)}_{2,12} \oplus x^{(7)}_{2,13}$ using Lemma 7.2 ($k = 2$), $x^{(7)}_{0,0}$ using the expansion for $x^{(m-1)}_{a,i}$ ($k = 1$), and $x^{(7)}_{3,17}$ using the expansion for $x^{(m-1)}_{d,i}$ ($k = 1$). Therefore, we get

$$\begin{gathered} x_0^{(7)}[0] \oplus x_2^{(7)}[12,13] \oplus x_3^{(7)}[17] = x_0^{(8)}[0,3,4]\oplus \\ x_2^{(8)}[4,12,14,17,18] \oplus x_3^{(8)}[14,18], \end{gathered} \tag{A.5}$$

with probability $\frac{1}{2}(1 + \frac{1}{2^4})$.

For Group II, we expand $x^{(7)}_{4,18} \oplus x^{(7)}_{4,19}$ and $x^{(7)}_{6,25} \oplus x^{(7)}_{6,26}$ using Lemma 7.2 ($k = 1$ and $k = 3$, respectively), $x^{(7)}_{4,7}$ using the expansion for $x^{(m-1)}_{d,i}$ ($k = 1$), $x^{(7)}_{7,26}$ using the expansion for $x^{(m-1)}_{c,i}$ ($k = 2$) and $x^{(7)}_{7,31}$ using the expansion for $x^{(m-1)}_{c,i}$ ($k = 2$). Therefore, we get

$$\begin{gathered} x_4^{(7)}[7,18,19] \oplus x_6^{(7)}[25,26] \oplus x_7^{(7)}[26,31] = \\ x_4^{(8)}[0,1,4,7,31]\oplus \\ x_5^{(8)}[16,17,18,19,21,22]\oplus \\ x_6^{(8)}[17,22] \oplus x_7^{(8)}[0,1,4], \end{gathered} \tag{A.6}$$

with probability $\frac{1}{2}(1 + \frac{1}{2^9})$.

For Group III, we expand $x^{(7)}_{8,13} \oplus x^{(7)}_{8,14}$ using Lemma 7.2 ($k = 2$), $x^{(7)}_{8,19}$ using the expansion for $x^{(m-1)}_{c,i}$ ($k = 2$), and $x^{(7)}_{11,31}$ using the expansion for $x^{(m-1)}_{b,i}$ ($k = 3$). Therefore, we get

$$\begin{gathered} x_8^{(7)}[13,14,19] \oplus x_{11}^{(7)}[31] = \\ x_8^{(8)}[6,11,13,14,18,24] \oplus x_9^{(8)}[6,18,19]\oplus \\ x_{10}^{(8)}[4,5,9,10,23,24] \oplus x_{11}^{(8)}[4,5,11,31], \end{gathered} \tag{A.7}$$

with probability $\frac{1}{2}(1 + \frac{1}{2^7})$.

For Group IV, we expand $x^{(7)}_{15,16} \oplus x^{(7)}_{15,17}$ using Lemma 7.2 ($k = 1$), $x^{(7)}_{12,0}$ using the expansion for $x^{(m-1)}_{b,i}$ ($k = 3$), $x^{(7)}_{12,14}$ using the expansion for $x^{(m-1)}_{b,i}$ ($k = 3$), $x^{(7)}_{14,12}$ using the expansion for $x^{(m-1)}_{d,i}$ ($k = 1$), and $x^{(7)}_{14,13}$ using the expansion for $x^{(m-1)}_{d,13}$ ($k = 0$). Therefore, we get

$$\begin{gathered} x_{12}^{(7)}[0,14] \oplus x_{14}^{(7)}[12,13] \oplus x_{15}^{(7)}[16,17] = \\ x_{12}^{(8)}[11,12,14,25,26,30,31]\oplus \\ x_{13}^{(8)}[0,7,12,21,26,30]\oplus \\ x_{14}^{(8)}[12,13,21,25,30,31]\oplus \\ x_{15}^{(8)}[6,7,16,17,24,25], \end{gathered} \tag{A.8}$$

with probability $\frac{1}{2}(1 + \frac{1}{2^8})$. Finally, using the Piling-up Lemma, we can combine the results from Lemma 7.3 and Eqs. (A.5)-(A.8), which leads to a correlation of $\varepsilon_L =$

$1/2^{6+4+9+7+8} = 2^{-34}$. ■

A.2.5 Proof of Lemma 9.8

proof.

From Eq. (9.25), we have that for subrounds 9 and 10 we do not update the word X_{10}, then we get $x^{[8]}_{10,0} = x^{[9]}_{10,0} = x^{[10]}_{10,0}$. Now, in subround 11, we have that $(a,b,c,d,e) = (2,6,10,14,1)$. Thus, X_{10} is of type X_c and using Lemma 9.1 we have $x^{[10]}_{10,0} = x^{[11]}_{1,0} \oplus x^{[11]}_{10,0} \oplus x^{[11]}_{14,0} \oplus x^{[11]}_{14,27}$, with probability 1.

In subround 12, words X_1, X_{10} and X_{14} are not expanded. Thus, we get

$$x^{[11]}_{1,0} \oplus x^{[11]}_{10,0} \oplus x^{[11]}_{14,0} \oplus x^{[11]}_{14,27} = x^{[12]}_{1,0} \oplus x^{[12]}_{10,0} \oplus x^{[12]}_{14,0} \oplus x^{[12]}_{14,27}.$$

In subround 13, we have $(a,b,c,d,e) = (0,5,10,15,3)$, and X_{10} is of type X_c. Again, using Lemma 9.1 we have

$$x^{[12]}_{1,0} \oplus x^{[12]}_{10,0} \oplus x^{[12]}_{14,0} \oplus x^{[12]}_{14,27} = \\ x^{[13]}_{1,0} \oplus x^{[13]}_{3,0} \oplus x^{[13]}_{10,0} \oplus x^{[13]}_{14,0} \oplus x^{[13]}_{14,27} \oplus x^{[13]}_{15,0} \oplus x^{[13]}_{15,27},$$

with probability 1.

In subround 14, we have $(a,b,c,d,e) = (1,6,11,12,0)$, and X_1 is of type X_a. Using Lemma 9.1 we have

$$x^{[13]}_{1,0} \oplus x^{[13]}_{3,0} \oplus x^{[13]}_{10,0} \oplus x^{[13]}_{14,0} \oplus x^{[13]}_{14,27} \oplus x^{[13]}_{15,0} \oplus x^{[13]}_{15,27} = \\ x^{[14]}_{1,8} \oplus x^{[14]}_{3,0} \oplus x^{[14]}_{10,0} \oplus x^{[14]}_{11,0} \oplus x^{[14]}_{14,0} \oplus x^{[14]}_{14,27} \oplus x^{[14]}_{15,0} \oplus x^{[14]}_{15,27}$$

with probability 1.

In subround 15, we have $(a,b,c,d,e) = (2,7,8,13,1)$, and X_1 is of type X_e. Using Lemma 9.2 we have

$$x^{[14]}_{1,8} \oplus x^{[14]}_{3,0} \oplus x^{[14]}_{10,0} \oplus x^{[14]}_{11,0} \oplus x^{[14]}_{14,0} \oplus x^{[14]}_{14,27} \oplus x^{[14]}_{15,0} \oplus x^{[14]}_{15,27} = x^{[15]}_{1,8} \oplus \\ x^{[15]}_{2,16} \oplus x^{[15]}_{3,0} \oplus x^{[15]}_{7,7} \oplus x^{[15]}_{7,8} \oplus x^{[15]}_{10,0} \oplus x^{[15]}_{11,0} \oplus x^{[15]}_{14,0} \oplus x^{[15]}_{14,27} \oplus x^{[15]}_{15,0} \oplus x^{[15]}_{15,27},$$

with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$.

Finally, in subround 16, we have $(a,b,c,d,e) = (3,4,9,14,2)$. Then, we have to expand the terms $x^{[15]}_{2,16}, x^{[15]}_{3,0}, x^{[15]}_{14,0}$ and $x^{[15]}_{14,27}$. Using Lemma 9.1 we can expand $x^{[15]}_{3,0}$ and $x^{[15]}_{14,0}$ with probability 1, and using Lemma 9.2 we can expand $x^{[15]}_{2,16}$ and $x^{[15]}_{14,27}$ with probabilities $\frac{1}{2}\left(1+\frac{1}{2}\right)$ and $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$, respectively. Therefore, by the Piling-up Lemma, we have

$$x_1^{[15]}[8] \oplus x_2^{[15]}[16] \oplus x_3^{[15]}[0] \oplus x_7^{[15]}[7,8] \oplus x_{10}^{[15]}[0] \oplus x_{11}^{[15]}[0] \oplus x_{14}^{[15]}[0,27] \oplus x_{15}^{[15]}[0,27] = x_1^{[16]}[8] \oplus x_2^{[16]}[16] \oplus x_3^{[16]}[2,3,24] \oplus x_4^{[16]}[0,15,16,26,27] \oplus x_7^{[16]}[7,8] \oplus x_9^{[16]}[0] \oplus x_{10}^{[16]}[0] \oplus x_{11}^{[16]}[0] \oplus x_{14}^{[16]}[22,27] \oplus x_{15}^{[16]}[0,27]$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^4}\right)$. Aggregating the correlation with the Piling-up Lemma completes the proof. ■

A.2.6 Proof of Lemma 9.9

proof.

In subround 17, we have $(a,b,c,d,e) = (0,4,8,12,3)$. Thus, we have to expand the terms $x_3^{[16]}[2,3,24]$ and $x_4^{[16]}[0,15,16,26,27]$. Here, we use Lemma 9.1 to expand $x_{4,0}^{[16]}$ with probability 1, and Lemma 9.3 to expand pairs $x_3^{[16]}[2,3]$, $x_4^{[16]}[15,16]$ and $x_4^{[16]}[26,27]$, with probabilities $\frac{1}{2}\left(1+\frac{1}{2}\right)$, $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$ and $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$, respectively. Additionally, with Lemma 9.2 we can expand $x_{3,24}^{[16]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Thus, from the Piling-up Lemma we have that

$$x_1^{[16]}[8] \oplus x_2^{[16]}[16] \oplus x_3^{[16]}[2,3,24] \oplus x_4^{[16]}[0,15,16,26,27] \oplus x_7^{[16]}[7,8] \oplus x_9^{[16]}[0] \oplus x_{10}^{[16]}[0] \oplus x_{11}^{[16]}[0] \oplus x_{14}^{[16]}[22,27] \oplus x_{15}^{[16]}[0,27] = x_0^{[17]}[0,10,11] \oplus x_1^{[17]}[8] \oplus x_2^{[17]}[16] \oplus x_3^{[17]}[2,3,24] \oplus x_4^{[17]}[2,3,4,5,10,23,24,25,26] \oplus x_7^{[17]}[7,8] \oplus x_8^{[17]}[0,4,5,10,15,16,25,27] \oplus x_9^{[17]}[0] \oplus x_{10}^{[17]}[0] \oplus x_{11}^{[17]}[0] \oplus x_{12}^{[17]}[0,15,16,26,27] \oplus x_{14}^{[17]}[22,27] \oplus x_{15}^{[17]}[0,27], \tag{A.9}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^6}\right)$.

In subround 18, we have $(a,b,c,d,e) = (1,5,9,13,0)$. Thus, we have to expand the terms $x_0^{[17]}[0,10,11]$, $x_{1,8}^{[17]}$ and $x_{9,0}^{[17]}$. Here, we use Lemma 9.1 to expand $x_{0,0}^{[17]}$ and $x_{9,0}^{[17]}$ with probability 1, and Lemma 9.3 to expand the pair $x_0^{[17]}[10,11]$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Additionally, with Lemma 9.2 we can expand $x_{1,8}^{[17]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Thus, from the Piling-up Lemma we have that

$$x_0^{[17]}[0,10,11] \oplus x_1^{[17]}[8] \oplus x_2^{[17]}[16] \oplus x_3^{[17]}[2,3,24] \oplus x_4^{[17]}[2,3,4,5,10,23,24,25,26] \oplus x_7^{[17]}[7,8] \oplus x_8^{[17]}[0,4,5,10,15,16,25,27] \oplus x_9^{[17]}[0] \oplus x_{10}^{[17]}[0] \oplus x_{11}^{[17]}[0] \oplus x_{12}^{[17]}[0,15,16,26,27] \oplus x_{14}^{[17]}[22,27] \oplus x_{15}^{[17]}[0,27] = x_0^{[18]}[10,11] \oplus x_1^{[18]}[8,16,18,19] \oplus x_2^{[18]}[16] \oplus x_3^{[18]}[2,3,24] \oplus x_4^{[18]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[18]}[0,10,11] \oplus x_7^{[18]}[7,8] \oplus x_8^{[18]}[0,4,5,10,15,16,25,27] \oplus x_9^{[18]}[0,7,8] \oplus x_{10}^{[18]}[0] \oplus x_{11}^{[18]}[0] \oplus x_{12}^{[18]}[0,15,16,26,27] \oplus x_{13}^{[18]}[0,27] \oplus x_{14}^{[18]}[22,27] \oplus x_{15}^{[18]}[0,27] \tag{A.10}$$

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

Next, in subround 19, we have $(a,b,c,d,e)=(2,6,10,14,1)$. Thus, we have to expand the terms $x_1^{[18]}[8,16,18,19]$, $x_2^{[18]}[16]$, $x_{10}^{[18]}[0]$ and $x_{14}^{[18]}[22,27]$. Here, we use Lemma 9.1 to expand $x_{10,0}^{[18]}$ with probability 1. Then, we use Lemma 9.4 to expand $x_{1,16}^{[18]} \oplus x_{2,16}^{[18]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Using Lemma 9.3 to expand $x_1^{[18]}[18,19]$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$. Additionally, with Lemma 9.2 we can expand $x_{1,8}^{[18]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$, and $x_{14,22}^{[18]}$ and $x_{14,23}^{[18]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2^3}\right)$. Thus, using the Piling-up Lemma we have that

$$\begin{aligned}
&x_0^{[18]}[10,11] \oplus x_1^{[18]}[8,16,18,19] \oplus x_2^{[18]}[16] \oplus x_3^{[18]}[2,3,24] \oplus x_4^{[18]}[2,3,4,5,10,23,\\
&24,25,26] \oplus x_5^{[18]}[0,10,11] \oplus x_7^{[18]}[7,8] \oplus x_8^{[18]}[0,4,5,10,15,16,25,27] \oplus\\
&x_9^{[18]}[0,7,8] \oplus x_{10}^{[18]}[0] \oplus x_{11}^{[18]}[0] \oplus x_{12}^{[18]}[0,15,16,26,27] \oplus x_{13}^{[18]}[0,27] \oplus\\
&x_{14}^{[18]}[22,27] \oplus x_{15}^{[18]}[0,27] = x_0^{[19]}[10,11] \oplus x_1^{[19]}[0,8,16,18,19] \oplus x_2^{[19]}[2,3,16,\\
&26,27,29,30] \oplus x_3^{[19]}[2,3,24] \oplus x_4^{[19]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[19]}[0,10,11] \oplus\\
&x_6^{[19]}[7,8,15,16,18,19,21,22,26,27] \oplus x_7^{[19]}[7,8] \oplus x_8^{[19]}[0,4,5,10,15,16,\\
&25,27] \oplus x_9^{[19]}[0,7,8] \oplus x_{10}^{[19]}[0,15,16] \oplus x_{11}^{[19]}[0] \oplus x_{12}^{[19]}[0,15,16,26,27] \oplus\\
&x_{13}^{[19]}[0,27] \oplus x_{14}^{[19]}[0,17,22,27] \oplus x_{15}^{[19]}[0,27]
\end{aligned} \tag{A.11}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^9}\right)$.

Finally, in subround 20, we have $(a,b,c,d,e)=(3,7,11,15,2)$. Thus, we have to expand the terms $x_2^{[19]}[2,3,16,26,27,29,30]$, $x_3^{[19]}[2,3,24]$, $x_7^{[19]}[7,8]$, $x_{11}^{[19]}[0]$ and $x_{15}^{[19]}[0,27]$. Here, we use Lemma 9.1 to expand $x_{11,0}^{[19]}$ and $x_{15,0}^{[19]}$ with probability 1. Then, we use Lemma 9.5 to expand $x_2^{[19]}[26,27] \oplus x_{15,27}^{[19]}$ with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Next, we use Lemma 9.4 to expand $x_2^{[19]}[2,3] \oplus x_3^{[19]}[2,3]$ with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Using Lemma 9.3 we can expand $x_2^{[19]}[29,30]$ and $x_7^{[19]}[7,8]$ with probabilities $\frac{1}{2}\left(1+\frac{1}{2}\right)$ and $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$, respectively. Additionally, with Lemma 9.2 we can expand $x_2^{[19]}[16]$ with probability $\frac{1}{2}\left(1+\frac{1}{2}\right)$, and $x_3^{[19]}[24]$ with probability $\frac{1}{2}\left(1+\frac{1}{2^2}\right)$. Thus, using the Piling-up Lemma we have that

$$x_0^{[19]}[10,11] \oplus x_1^{[19]}[0,8,16,18,19] \oplus x_2^{[19]}[2,3,16,26,27,29,30] \oplus x_3^{[19]}[2,3,24] \oplus x_4^{[19]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[19]}[0,10,11] \oplus x_6^{[19]}[7,8,15,16,18,19,21,22,26,27] \oplus x_7^{[19]}[7,8] \oplus x_8^{[19]}[0,4,5,10,15,16,25,27] \oplus x_9^{[19]}[0,7,8] \oplus x_{10}^{[19]}[0,15,16] \oplus x_{11}^{[19]}[0] \oplus x_{12}^{[19]}[0,15,16,26,27] \oplus x_{13}^{[19]}[0,27] \oplus x_{14}^{[19]}[0,17,22,27] \oplus x_{15}^{[19]}[0,27] = x_0^{[20]}[10,11] \oplus x_1^{[20]}[0,8,16,18,19] \oplus x_2^{[20]}[0,2,3,16,26,27,29,30] \oplus x_3^{[20]}[0,5,6,8,24] \oplus x_4^{[20]}[2,3,4,5,10,23,24,25,26] \oplus x_5^{[20]}[0,10,11] \oplus x_6^{[20]}[7,8,15,16,18,19,21,22,26,27] \oplus x_7^{[20]}[0,2,3,15,16,17,18,29,30] \oplus x_8^{[20]}[0,4,5,10,15,16,25,27] \oplus x_9^{[20]}[0,7,8] \oplus x_{10}^{[20]}[0,15,16] \oplus x_{11}^{[20]}[0,2,3,7,8,17,18,23,24] \oplus x_{12}^{[20]}[0,15,16,26,27] \oplus x_{13}^{[20]}[0,27] \oplus x_{14}^{[20]}[0,17,22,27] \oplus x_{15}^{[20]}[0,7,8,22] \tag{A.12}$$

with probability $\frac{1}{2}\left(1+\frac{1}{2^{10}}\right)$.

To conclude, we compute the correlation by using the Piling-up Lemma and aggregating the correlations of Lemma 9.8 and Eqs. (A.9)-(A.12), thus we get $\varepsilon_L = \frac{1}{2^{5+6+3+9+10}}$. ■