Few Quantum Cryptanalysis Techniques

by

SANCHITA SAHA

under the supervision of

Dr. Debajyoti Bera

submitted

in partial fulfilment of the requirements for the degree of

MASTER OF TECHNOLOGY

to

COMPUTER SCIENCE AND ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI - 110020

May, 2021

CERTIFICATE

This is to certify that the thesis titled "**Few Quantum Cryptanalysis Technique**", submitted by **Sanchita Saha**, to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

May, 2023

Dr. Debajyoti Bera
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to my M.Tech. Thesis supervisor, Dr. Debajyoti Bera, for his invaluable guidance and support throughout this research work. He has been a constant source of inspiration and motivation for me. I want to thank my committee members, Dr. Santanu Sarkar and Dr. Subhamoy Maitra, for engaging with my thesis diligently and enthusiastically and providing insightful feedback. The earlier discussions on the cryptanalysis of ChaCha cipher with Dr. Maitra, at various stages of the work led me to look deeper into the algorithms and approaches used.

I am grateful to Dr. Subhabrata Samajder for his valuable insights on the possible extensions and applications of the current work. His guidance helped me better understand the cryptographic challenges and solutions.

I thank my family and friends for their unconditional love and encouragement. They have always stood by me in times of difficulty and joy. I want to acknowledge the help and cooperation of my labmates, especially Mr. Sagnik Chatterjee and Mr. Tharrmashastha SAPV, who have shared their insights and expertise with me. They have always provided constructive feedback and suggestions for improving my work. I am also indebted to the IIITD administration and IIITD Computer Science and Engineering department for providing me with the necessary facilities and resources for conducting this research. I thank Ms. Priti Patwal for her immense support and cooperation.

I am grateful to all the people who have directly or indirectly contributed to this thesis. I appreciate their time and effort.

ABSTRACT

Quantum algorithms like Shor's and Grover's algorithms have proven to break several cryptographic schemes theoretically. Shor's algorithm, which can factor numbers exponentially faster than classical computers, can break asymmetric cryptosystems like RSA. Grover's algorithm compromises symmetric key ciphers, as it provides asymptotic quadratic speedup for unordered search. However, given the current resource limitations, whether these techniques will be practically realizable is debatable. In this thesis, we aim to show some evidence that quantum can aid cryptanalysis practically.

Several studies have been conducted on quantum cryptanalysis of block ciphers like AES etc., but not enough on stream ciphers. ChaCha cipher is a stream cipher with gaining popularity. It has been included in TLS 1.3. We designed a quantum version of the ChaCha cipher with reduced depth. The best classical attack on ChaCha7 is a 2^{214} operation attack. Most classical attacks on ChaCha use the notion of Input Difference (ID), Output Difference (OD), and Probabilistic Neutral Bits (PNBs). We proposed two quantum algorithms, one to get the best ID-OD pair and the other to get the PNBs. However, the oracle used for PNBs is massive, and amplitude estimation on it would severely blow up the circuit. For a workaround to this, we started exploring other techniques like linear cryptanalysis.

Linear cryptanalysis uses linear expressions between input or plaintext (n bits) and output or ciphertext (m bits) bits to attack ciphers. Classically, a linear approximation table of size $2^n \times 2^m$ is used to count the probability of occurrence of each linear expression exhaustively. We developed a quantum algorithm to reduce space and time for the same. Our circuit generates a superposition of all possible linear combinations with amplitudes proportional to respective linear probability biases. Applying multi-distribution amplitude estimation to these states, followed by marking states above a predefined threshold, and retrieving all the marked states, gives us a list of high-probability linear expressions of the cipher. These linear expressions could then be used to launch attacks on the ciphers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Classical cryptography relies on mathematical operations that are hard to invert, such as modular exponentiation and hashing. However, with the advent of quantum computing, some of these operations have become vulnerable to quantum algorithms that can solve them faster than classical ones. Instances of these are Grover's algorithm that gives asymptotic quadratic speedup for unordered search; and Shor's algorithm which gives exponential speedup in factorising numbers. At the same time, if the added quantum computing capabilities are harnessed appropriately, we might be able to develop better quantum ciphers. In this paper, we explore the implications of quantum computing for both encryption of information and cryptanalysis of encryption schemes (ciphers).

We focus on the ChaCha cipher, a stream cipher with increasing popularity in recent times. It is widely used in applications such as TLS and OpenSSL. Our interest lies in implementing an efficient and correct quantum circuit for the ChaCha cipher. For the next part we try to develop quantum subroutines that can could replace computationally expensive parts in classical cryptanalysis.

Our contributions in this work are: we have designed a quantum implementation of ChaCha cipher with reduced circuit depth. We gave a quantum algorithm to quantize a phase of the classical ChaCha cryptanalysis process. This phase is to make a choice of Input Difference - Output Difference pair based on high probability of occurrence. The third quantum algorithm we propose is for finding linear approximations for any cipher.

1.1 Organisation

Chapter 2 discusses the scheme of the ChaCha cipher, followed by an efficient quantum implementation of ChaCha. In chapter 3 we study the various classical techniques used for cryptanalysis of ChaCha. We emphasise on the concept of Input-Output difference and Probabilistic Neutral Bits, and give an quantum algorithm to find good ID-OD

pairs. In chapter 4 we give a more generic algorithm to find linear approximations of any cipher. We conclude finally in 5 by giving a summary of the results and discussing the possible future directions.

CHAPTER 2

Quantum ChaCha

2.1 ChaCha cipher

Stream ciphers are symmetric key ciphers, i.e., the same key is used for encryption and decryption. In stream ciphers, a given key generates a pseudorandom sequence called keystream, which is then used to encrypt the plaintext to ciphertext by some operation, usually XOR.

The ChaCha family of stream ciphers was developed by D. J. Bernstein (2008). It was proposed as a variant of Salsa20, also designed by Bernstein in 2005. Salsa20 was a finalist in the eSTREAM software-oriented profile (profile 1). The eSTREAM project was a multi-year effort, from 2004 to 2008, to promote the design of efficient and compact stream ciphers suitable for widespread adoption (ECRYPT, Accessed on 2023-05-14).

ChaCha has a state of 4×4 matrix of 32-bit words, initially comprising constants, key, counter, and nonce. It is an ARX (add, rotate, XOR) based cipher, which on software are fast operations. This makes it consistently faster than AES, which is the most common encryption method today. ChaCha is, in fact, faster or equally fast as Salsa20. Moreover, compared to Salsa20, it conjecturally has higher diffusion per round, providing higher cryptanalytic resistance.

It has been included in TLS 1.3 cipher suite. Google adopted ChaCha20 for symmetric encryption and Poly1305 for authentication (MAC) in OpenSSL and NSS in March 2013.

2.2 Structure of ChaCha

ChaCha consists of a 4×4 matrix of words, each of size 4 bytes (32 bits) [Figure: 2.1]. There are a total of 512 bits in a ChaCha cipher state.

2.2.1 Initial State:

The rows in the **initial state** are described as follows:

- First row: Consists of 4 predefined **Constant** (C), i.e. 128 bits, C_0, C_1, C_2, C_3. In the version proposed by Bernstein, the constants are specified as "expand 32-byte k". Whereas, the constants for 128-bit key version of ChaCha are $C0 = 0x61707865, C1 = 0x3120646e, C2 = 0x79622d36, C3 = 0x6b206574$.
- Second and third row: Consists of **Key** (K), 256 bits, $K_0, \ldots K_7$. This is the secret part of the cipher. It is only known to the sender and receiver.
  For **ChaCha128** it has a 128-bit key structure, 4 keywords make a copy of itself and fill up the matrix's second and third row.
- Fourth row: First word $(32$ bits) is **counter** (T). The counter starts from 0, and gets incremented after transmission of each message block of 512 bytes.
  Next three words $(96$ bits) are **Nonce** (V). Every time for a new message the nonce is randomly generated. The counter and nonce values are usually publicly know for each block of message.

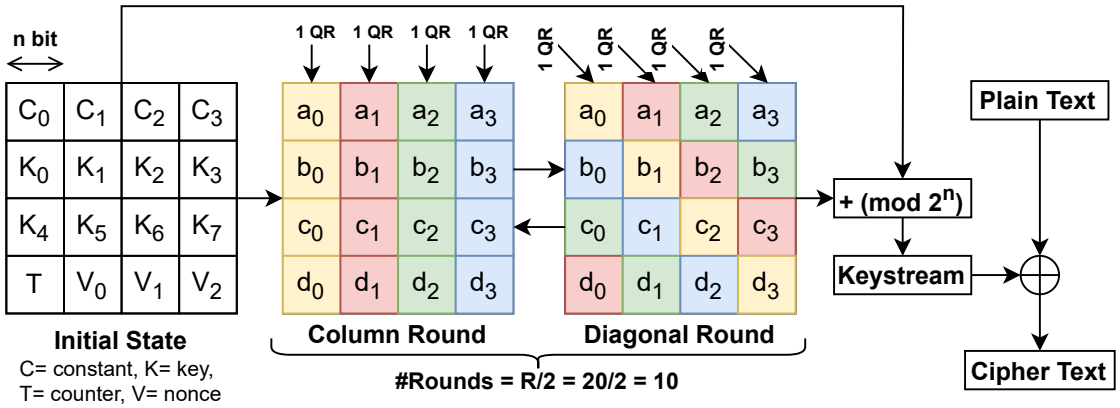

Figure 2.1: ChaCha cipher

2.2.2 Quarter-round

The scheme of the cipher consists of quarter-rounds. Each quarter-round consists of four ARX rounds: addition modulo 2^{32} (A), circular left rotation (R), XOR (X) operations. One quarter-round of the cipher on (a, b, c, d) can be described as follows:

$$\begin{aligned}
&\text{ARX-1} : a = a + b;\ d = d \oplus a;\ d = d \lll 16;\\
&\text{ARX-2} : c = c + d;\ b = b \oplus c;\ b = b \lll 12;\\
&\text{ARX-3} : a = a + b;\ d = d \oplus a;\ d = d \lll 8;\\
&\text{ARX-4} : c = c + d;\ b = b \oplus c;\ b = b \lll 7;
\end{aligned} \tag{2.1}$$

2.2.3 Column and Diagonal Round

ChaChaR consists of R rounds, i.e., $R/2$ column rounds and $R/2$ diagonal rounds, applied alternately. Each column round consists of 4 quarter-rounds, one on each column of the current matrix state. Similarly, each diagonal round consists of 4 quarter-rounds, one along each diagonal. A complete **column round** can be written as:

$$\begin{aligned}
&\text{quarter-round}(a_0, b_0, c_0, d_0)\\
&\text{quarter-round}(a_1, b_1, c_1, d_1)\\
&\text{quarter-round}(a_2, b_2, c_2, d_2)\\
&\text{quarter-round}(a_3, b_3, c_3, d_3)
\end{aligned} \tag{2.2}$$

A complete **diagonal round** can be written as:

$$\begin{aligned}
&\text{quarter-round}(a_0, b_1, c_2, d_3)\\
&\text{quarter-round}(a_1, b_2, c_3, d_0)\\
&\text{quarter-round}(a_2, b_3, c_0, d_1)\\
&\text{quarter-round}(a_3, b_0, c_1, d_2)
\end{aligned} \tag{2.3}$$

2.2.4 Final addition and encryption

After the R rounds, the matrix state is added (modulo 2^{32}) to the initial state. The final state is then serialized to get a keystream, which is XOR-ed with the plaintext to get ciphertext or vice versa. The final addition of the initial secret state makes the transform key-to-keystream-block non-invertible (Aumasson and Green, 2017).
Since the sender and receiver both have access to key (only known to them), counter and nonce (usually publicly shared), they can generate the same keystream, this ensures the message integrity.

2.3 Quantum ChaCha

The first step of attempting cryptanalysis of any cipher would be to construct a quantum version of the cipher. This construction can serve as an oracle model for cryptanalytic investigations. For ChaCha quantum version we want a circuit U (unitary) that implements the following:

$$U\left|X\right\rangle\left|0\right\rangle \longrightarrow \left|X\right\rangle\left|ChaCha(X)\right\rangle \tag{2.4}$$

where, $ChaCha(X)$ is ChaCha cipher as a function on X, the initial matrix state. The main objectives of any implementation should be reducing the number of qubits, reducing the depth and ensuring correctness.

Bathe *et al.* (2021) gave a quantum implementation of ChaCha cipher. For the modulo addition operation they used method of Takahashi *et al.* (2009), which uses no ancilla. For component-wise XOR they use CNOT gates, and shift operation is done by rewiring, not requiring any quantum gates.

In our implementation, we have used modulo 2^n addition method of Cuccaro *et al.* (2004*a*) using ripple carry adder. This method reduces the depth of the circuit, but uses additional ancilla of 1 bit. The algorithm for modulo 2^n addition is elaborated in Algorithm 1. For XOR and shift operation we used the same method as was used by Bathe *et al.* (2021).

Comparative analysis of our implementation with Bathe *et al.* (2021) in terms of resource is given in Table 2.1. Since at most 4 addition occur in parallel, along the four column/diagonal at a time, thus we need 4 ancilla bit for each. And in the last addition phase, we have 16 independent additions. In the case where we use 4 ancilla, we do 4 of these 16 additions at a time. There is a reduction in depth by almost half from the prior work, due to the design choices. In the case where we use 16 ancilla, all the 16 additions are done parallely, this reduces the depth further to 5346. Besides the circuit depth, we report a reduction in the $CNOT$ and $Toffoli$ gates. The reduction in the gates and circuit depth is in tradeoff with the number of ancilla qubits.

Algorithm 1 Addition modulo 2^n (Cuccaro *et al.*, 2004*a*)

Input: Two n-bit numbers a, b initialized in quantum registers $A = a, B = b$
Output: Registers with value $A = a, B = (a + b) \mod 2^n$

$k = n - 1$
for $i = 1$ **to** $k - 1$: $B_i \oplus= A_i$
$X \oplus= A_1$
$X \oplus= A_0 B_0$; $A_1 \oplus= A_2$
$A_1 \oplus= X B_1$; $A_2 \oplus= A_3$
for $i = 2$ **to** $k - 3$:
    $A_i \oplus= A_{i-1} B_i$; $A_{i+1} \oplus= A_{i+2}$
$A_{k-2} \oplus= A_{k-3} B_{k-2}$; $Z \oplus= A_{k-1}$
$Z \oplus= A_{k-2} B_{k-1}$; **for** $i = 1$ **to** $k - 2$: NegateB_i
$B_1 \oplus= X$; **for** $i = 2$ **to** $k - 1$: $B_i \oplus= A_{i-1}$
$A_{k-2} \oplus= A_{k-3} B_{k-2}$
for $i = 2$ **down to** $k - 3$:
    $A_i \oplus= A_{i-1} B_i$; $A_{i+1} \oplus= A_{i+2}$; NegateB_{i+1}
$A_1 \oplus= X B_1$; $A_2 \oplus= A_3$; NegateB_2
$X \oplus= A_0 B_0$; $A_1 \oplus= A_2$; NegateB_1
$X \oplus= A_1$
$B_{n-1} \oplus= A_{n-1}$; **for** $i = 0$ **to** $k - 1$: $B_i \oplus= A_i$

Name	CNOT	Toffoli	Qubits	Depth
Bathe *et al.* (2021)	61984	20832	1024	12635
This work	61648	20496	1024+4 ancilla	6334
This work	61648	20496	1024+16 ancilla	5346

Table 2.1: Comparison of resource estimation for ChaCha20, i.e., for 20 rounds, and the final addition phase included

CHAPTER 3

Quantum approaches for ChaCha cryptanalysis

3.1 Related Works

Classically several works have been done on the cryptanalysis of the ChaCha cipher. The first significant attempt was a 2^{165} operation truncated differential cryptanalysis attack designed by Crowley (2005) on 5-round ChaCha. Subsequently, it was extended to 6-round ChaCha by Fischer (2008).

Aumasson *et al.* (2007) proposed a concept called 'Probabilistic Neutral Bits' (PNBs) in 2008. This uses a meet-in-the-middle technique, where they find a set of key bits that has less impact on the position of the output difference at a certain round when going backwards from the final state. Using this they attacked 7-round ChaCha256 and 6-round ChaCha128 (256 and 128 are the respective key sizes in bits). Shi *et al.* (2013) extended the approach by using column chaining distinguisher (CCD). The idea of using chosen IV cryptanalysis given by Maitra (2015) drastically reduced the complexity to 2^{239} for 7-round ChaCha. Choudhuri and Maitra (2016) gave a differential linear approach by considering multiple bit distinguisher of a few next rounds. Dey and Sarkar (2017) improved the set of PNBs. Beierle *et al.* (2020) proposed a attack against 7-round ChaCha256 with $2^{230.86}$ operations, using a generic framework for differential-linear attacks with a special focus on ARX ciphers. The best known attack till date against 7-round ChaCha256 is 2^{214} by Coutinho and Neto (2021). They proposed an improved differential-linear distinguisher against ChaCha.

3.2 Differential Cryptanalysis of ChaCha

In this section we discuss the generic technique used for ChaCha as proposed by Aumasson *et al.* (2007). It uses differential cryptanalysis to precompute biases, find probabilistic neutral bits (PNBs) and perform a probabilistic backward computation. Then

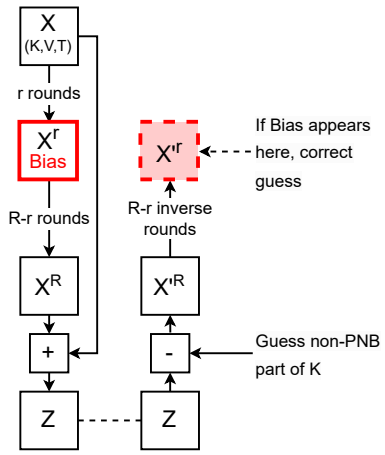

Figure 3.1: Principle of attack of ChaCha

in the online phase the actual key recovery attack is performed with the help of these PNBs.

Let the matrix state be X. Matrix state after r rounds is X^r. R is the total number of rounds. The keystream of 512 bits is obtained as

$$Z = X^0 + X^R \tag{3.1}$$

x_i is the i-th word in X, and $x_{i,j}$ represent the j-th bit of x_i. The 0-th bit is the least significant bit for any word.

3.2.1 Precomputation Phase

Single-Bit Differential Biases

Given two states $X^{(r)}, X'^{(r)}$, we denote $\Delta_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$. And $\Delta_{i,j}^{(r)} = x_{i,j}^{(r)} \oplus x_{i,j}'^{(r)}$ denotes the difference between two states at the j-th bit of the i-th word after r many rounds.

Based on the difference $\Delta_{i,j}^{(0)} = 1$ to the initial state matrix $X^{(0)}$, which is called the **input difference or** $\mathcal{ID}$, we obtain the corresponding initial state matrix $X'^{(0)}$. Then, we execute the round function of ChaCha using these two initial state matrices as inputs and obtain $\Delta_{p,q}^{(r)} = x_{p,q}^{(r)} \oplus x_{p,q}'^{(r)}$ from the r-round output internal state matrices $X^{(r)}$ and $X'^{(r)}$, which is called the **output difference or** $\mathcal{OD}$. From this the single-bit differential probability over a fixed key, and all possible nonces(V) and counter(T), can

be calculated as

$$\Pr_{V,T}(\Delta_{p,q}^{(r)} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \epsilon_d) \tag{3.2}$$

Here ϵ_d denotes the $\mathcal{OD}$ bias.

Let D_0 be the uniform distribution and D_1 be a distribution of the truncated $\mathcal{OD}$ bit strings obtained from the internal state of ChaCha. In this case, the target event (3.2) occurs in D_0 and D_1 with probabilities of $\frac{1}{2}$ and $\frac{1}{2} + \epsilon_d$, respectively. Based on this, the number of samples of the best distinguisher between $D = D_0$ and $D = D_1$ can be estimated as $\frac{4}{\epsilon_d^2}$ with an overall probability of error of $P_e \approx \Phi(-\sqrt{4}/2) = \Phi(-1) \approx 0.158655$, where $\Phi(.)$ is the distribution function of the standard normal distribution (Miyashita *et al.*, 2021).

Probabilistic Neutral Bit (PNB)

For Probabilistic neutral bit (PNB) we first need the concept of neutrality measure, which is the degree of influence of the key bit k with respect to the $\mathcal{OD}$. It is formally defined as:

Definition 3.2.1 (Neutrality Measure)**.** The neutrality measure of the key bit position k with respect to the OD is defined as γ_k, where $\frac{1}{2}(1 + \gamma_k)$ is the probability (over all K and V,T) that complementing the key bit k does not change the $\mathcal{OD}$.

For example, we have the following singular cases of neutral measure:

- $\gamma_i = 1$: $\mathcal{OD}$ does not depend on the i-th key bit (i.e., it is **non-significant**). These are the **PNBs**.
- $\gamma_i = 0$: $\mathcal{OD}$ is statistically independent of the i-th key bit (i.e., it is **significant**). These are the **Non-PNBs**
- $\gamma_i = -1$: $\mathcal{OD}$ linearly depends on the i-th key bit.

The formula for neutrality measure can be given as

$$\Pr(\Delta_{p,q}^{(r)} = \Gamma_{p,q}^{(r)} | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \gamma_k) \tag{3.3}$$

$\Gamma_{p,q}^{(r)}$ has been defined in Algorithm 2. It gives the process to compute the neutrality measure and retrieve the set of PNBs.

Algorithm 2 Computation of PNBs

Input: Number of rounds R and $r(r < R)$, key bit position k, neutrality measure threshold γ_τ

Output: Set of significant and non-significant key bits

1: Start with pair of input states X and X', where X' is X with $\Delta^0_{i,j} = 1$
▷ The (i, j) pair is the precomputed suitable $\mathcal{ID}$ bit.
2: Derive keystream blocks $Z = X^{(0)} + X^{(R)}$ and $Z' = X'^{(0)} + X'^{(R)}$
3: Prepare new input pair $\overline{X}^{(0)}$ and $\overline{X'}^{(0)}$, such that X and $\overline{X}^{(0)}$ have only key bit position k is flipped. Similarly, X' and $\overline{X'}^{(0)}$ have only key bit position k flipped.
4: Compute r-round internal state matrix pair $Y^{(r)}$ and $Y'^{(r)}$ with $Z - X^{(0)}$ and $Z' - X'^{(0)}$ as inputs to the inverse round function of ChaCha.
5: Compute $\Gamma^{(r)}_{p,q} = y^{(r)}_{p,q} \oplus y'^{(r)}_{p,q}$ for the fixed $\mathcal{OD}$ bit, where $y^{(r)}_{p,q}$ denotes the q-th bit of the p-th word of $Y^{(r)}$ and similarly for $Y'^{(r)}$).
▷ The (p, q) pair is the precomputed suitable $\mathcal{OD}$ bit.
6: Repeat steps $1 - 4$ for T times, using different initial state matrices with the same $\mathcal{ID}$ bit, to get the value of $\Pr(\Delta^{(r)}_{p,q} = \Gamma^{(r)}_{p,q} | \Delta^{(0)}_{i,j} = 1) = s$(say).
7: Compute $\gamma_k = 2s - 1$
8: Set a threshold γ and place all key bits with $\gamma_\tau < \gamma$ into the set of m-bit significant key bits and those with $\gamma_\tau \geq \gamma$ into the set of n-bit non-significant key bits.
9: Return set of significant and non-significant (PNB) bits.

Probabilistic Backward Computation (PBC)

The r-round bias ϵ_a is given as:

$$\Pr(\Delta^{(r)}_{p,q} = \hat{\Gamma}^{(r)}_{p,q} | \Delta^{(0)}_{i,j} = 1) = \frac{1}{2}(1 + \epsilon_a) \tag{3.4}$$

We can obtain the r-round single-bit differential biases for ChaCha20/R from the obtained keystream by performing the following probabilistic backward computation (Algorithm 3). The bias of $\hat{\Gamma}^{(r)}_{p,q}$ is denoted as ϵ and given as:

$$\Pr(\hat{\Gamma}^{(r)}_{p,q} = 1 | \Delta^{(0)}_{i,j} = 1) = \frac{1}{2}(1 + \epsilon) \tag{3.5}$$

According to Aumasson *et al.* (2007), the following approximation can be used

$$\epsilon \approx \epsilon_d . \epsilon_a \tag{3.6}$$

This is used to compute the overall complexity of the attack on the R-round target cipher.

Algorithm 3 Probabilistic Backward Computation

Input: Number of rounds R and $r(r < R)$, key bit position k

Output: r-round bias ϵ_a

1: Compute the R-round internal state matrix pair $(X^{(R)}, X'^{(R)})$ corresponding to the input pair $(X^{(0)}, X'^{(0)})$ with $\Delta_{i,j}^{(0)} = 1$, and derive the keystream blocks $Z = X^{(0)} + X^{(R)}$ and $Z' = X'^{(0)} + X'^{(R)}$, respectively.
▷ The (i, j) pair is the precomputed suitable $\mathcal{ID}$ bit.

2: Prepare a new input pair $(\hat{X}^{(0)}, \hat{X}'^{(0)})$ with only non-significant key bits reset to a fixed value (e.g., all zeros) from the original input pair $(X^{(0)}, X'^{(0)})$

3: Compute the r-round internal state matrix pair $(\hat{Y}^{(r)}, \hat{Y}'^{(r)})$ with $Z - \hat{X}^{(0)}$ and $Z' - \hat{X}'^{(0)}$ as inputs to the inverse round function of ChaCha.

4: Compute $\hat{\Gamma}_{p,q}^{(r)} = \hat{y}_{p,q}^{(r)} \oplus \hat{y}'^{(r)}_{p,q}$ for the fixed $\mathcal{OD}$ bit, where $\hat{y}_{p,q}^{(r)}$ denotes the q-th bit of the p-th word of $\hat{Y}^{(r)}$ and similarly for $\hat{Y}'^{(r)}$).
▷ The (p, q) pair is the precomputed suitable $\mathcal{OD}$ bit.

5: Repeat steps $1 - 4$ for T times, using different initial state matrices with the same $\mathcal{ID}$ bit, to get the value of $\Pr(\Delta_{p,q}^{(r)} = \hat{\Gamma}_{p,q}^{(r)} | \Delta_{i,j}^{(0)} = 1) = s$(say).

6: Compute $\epsilon_a = 2s - 1$ and return

3.2.2 Online Phase

After the precomputation phase, the following steps are performed to recover an unknown key:

Algorithm 4 Online Phase

Input: N keystream block pairs, each pari differeing at ID bit, set of significant bits, set of non significant bits

Output: Correct key for the keystream blocks

1: For an unknown key, collect N keystream block pairs where each pair is generated by a random input pair satisfying the relevant $\mathcal{ID}$.

2: For each choice of the subkey (i.e., m-bit significant key bits), the following steps should be performed

2.1: Derive the r-round single-bit differential biases from the obtained N keystream block pairs by performing backward computation.

2.2: If the optimal distinguisher legitimates the subkeys candidate as (possibly) correct, perform an additional exhaustive search over the n-bit non-significant key bits to confirm the correctness of the filtered subkey and identify the n-bit non-significant key bits.

2.3: Stop if the correct key is reported and output the recovered key.

3.2.3 Complexity Analysis

Given N keystream block pairs and a false alarm probability of $P_{fa} = 2^{-\alpha}$, the time complexity of the attack is (Miyashita *et al.*, 2021)

$$2^m(N + 2^n P_{fa}) = 2^m N + 2^{256-\alpha}, \text{where} N \approx (\frac{\sqrt{\alpha}\log 4 + 3\sqrt{1-\epsilon^2}}{\epsilon})^2, \tag{3.7}$$

for a probability of non-detection $P_n d = 1.310^{-3}$. In practice, α and thus N are selected to minimize the time complexity of the attack. Based on an existing study (Aumasson *et al.*, 2007), generally the median bias ϵ is used in attacks; therefore, the attacks have a success probability of approximately 0.5.

3.3 Finding ID-OD pair

As we discussed in single bit differential (3.2.1) part, the first step for this complete process is to identify the suitable pair of input difference ($\mathcal{ID}$) and output difference ($\mathcal{OD}$) bits. The classical procedure to determine the same is as follows:

Algorithm 5 Find best $\mathcal{ID} - \mathcal{OD}$ pair

1: Let COUNT be a 2D matrix of to maintain differential bias of each $\mathcal{ID} - \mathcal{OD}$ pair over multiple initial states
2: **for** each initial state X in a set of randomly generated initial states **do**
3: **for** all (i, j) (ie. j-th bit of i-th word) **do**
4: Fix (i, j) as the $\mathcal{ID}$ bit
5: Create a second copy X' of X by applying flipping bit at $\mathcal{ID}$ bit
6: Perform r rounds of ChaCha cipher on both X and X'
7: **for** all (p, q) (ie. q-th bit of p-th word) **do**
8: Calculate the bit difference ($\mathcal{OD}$) at (p, q) bit.
9: If $\mathcal{ID} = \mathcal{OD}$, then increment by 1 value of $[(i, j), (p, q)]$ in COUNT
10: If $\mathcal{ID} \neq \mathcal{OD}$, then decrement by 1 value of $[(i, j), (p, q)]$ in COUNT
11: **end for**
12: **end for**
13: **end for**
14: Normalize COUNT
15: Return the $\mathcal{ID} - \mathcal{OD}$ pair with highest differential bias

In 5 in step 2, when used practically for cryptanalytic attacks, for the set of randomly generated initial states, number of sample of states taken are in range 2^{20} to 2^{30} since exhaustively trying all inputs is not possible. If we wanted to exhaustively run the

experiments over all states (step 2), i.e., key, nonce and counter the number of iterations needed would be 2^{384}. Classically, this is infeasible. To overcome this limitation, we have developed a quantum alternative which has been elaborately explained in the next section.

3.4 Quantum Algorithm to find best ID-OD pair

To simplify notations let us take index (i, j) in the matrix state as i and (p, q) as p. Since, this is only a representation of the bit position it would not affect the algorithm or analysis.

Algorithm 6 Oracle for generating OD: ODGEN

Input: Some initial state X, i bit for $\mathcal{ID}$, p bit for $\mathcal{OD}$, r rounds
Output: The $\mathcal{OD}$ at p) bit

1: Take 2 registers. First register has some key(K), nonce(V), counter(T) bit ($X = (K, V, T)$). Second register initialised as 0.
$\triangleright |X\rangle |0\rangle$

2: Apply Hadamard gate on second register
$\triangleright \frac{1}{\sqrt{2}} (|X\rangle |0\rangle + |X\rangle |1\rangle)$

3: Apply $CNOT$ gate to i in first register, conditioned on whether second register $|1\rangle$.
$\triangleright \frac{1}{\sqrt{2}} (|X\rangle |0\rangle + |X'\rangle |1\rangle)$

4: Perform r rounds of ChaCha. Let W be the unitary corresponding to r-round ChaCha.
$\triangleright \frac{1}{\sqrt{2}} \{(|W(X)\rangle) |0\rangle + (|W(X')\rangle) |1\rangle\}$

5: Apply a Z-gate on (p, q) bit of first register
$\triangleright \frac{1}{\sqrt{2}} \left\{(-1)^{(W(X))_p}(|W(X)\rangle) |0\rangle + (-1)^{(W(X'))_p}(|W(X')\rangle) |1\rangle\right\}$
where $(W(X))_p$ denotes p bit of W(X)

6: Perform r reverse rounds of ChaCha
$\triangleright \frac{1}{\sqrt{2}} \left\{(-1)^{(W(X))_p} |X\rangle |0\rangle + (-1)^{(W(X'))_p} |X'\rangle |1\rangle\right\}$

7: Apply $CNOT$ gate to i in first register, conditioned on whether second register $|1\rangle$
$\triangleright \frac{1}{\sqrt{2}} \left\{(-1)^{(W(X))_p} |X\rangle |0\rangle + (-1)^{(W(X'))_p} |X\rangle |1\rangle\right\} = \frac{1}{\sqrt{2}} |X\rangle [(-1)^{(W(X))_p} |0\rangle + (-1)^{(W(X'))_p} |1\rangle]$

8: Hadamard gate on the second register.
$\triangleright (-1)^{(W(X))_p} |X\rangle |(W(X))_p \oplus (W(X'))_p\rangle$

Explanation of step $7, 8$: $(W(X))_p$ denotes a single bit, and therefore will be either

0 or 1. In step 7

$$\text{If } (W(X))_p = (W(X'))_p \implies \text{State is } \frac{(-1)^{(W(X))_p}}{2} |X\rangle (|0\rangle + |1\rangle) \tag{3.8}$$

$$\text{If } (W(X))_p \neq (W(X'))_p \implies \text{State is } \frac{(-1)^{(W(X))_p}}{2} |X\rangle (|0\rangle - |1\rangle) \tag{3.9}$$

When H is applied on this state, we get

$$\text{If } (W(X))_p = (W(X'))_p \implies \text{State is } (-1)^{(W(X))_p} |X\rangle (|0\rangle) \tag{3.10}$$

$$\text{If } (W(X))_p \neq (W(X'))_p \implies \text{State is } (-1)^{(W(X))_p} |X\rangle (|1\rangle) \tag{3.11}$$

Collectively we can say,

$$\text{State is } (-1)^{(W(X))_p} |X\rangle |(W(X))_p \oplus (W(X'))_p\rangle \tag{3.12}$$

where the value $|(W(X))_p \oplus (W(X'))_p\rangle$ represents the $\mathcal{OD}$ at p for $\mathcal{ID}$ at i.

Algorithm 6 gives the $\mathcal{OD}$ at a particular bit position. It can be modified to take input i and p as registers, and give the same result, i.e.,

$$|i\rangle |p\rangle |X\rangle |0\rangle \rightarrow |i\rangle |p\rangle |X\rangle (-1)^{(W(X))_p} |(W(X))_p \oplus (W(X'))_p\rangle \tag{3.13}$$

Since all the operations in the algorithm are unitary, we can apply the algorithm linearly over a superposition of states. Taking the initial state, i and p registers respectively as,

$$|X\rangle = \frac{1}{\sqrt{2^{n_X}}} \sum_{X \in \mathbb{F}_2^{n_X}} |X\rangle \tag{3.14}$$

$$|i\rangle = \frac{1}{\sqrt{2^{n_i}}} \sum_{i \in \mathbb{F}_2^{n_i}} |i\rangle \tag{3.15}$$

$$|p\rangle = \frac{1}{\sqrt{2^{n_p}}} \sum_{p \in \mathbb{F}_2^{n_p}} |p\rangle \tag{3.16}$$

where, n_X, n_i, n_p are the number of qubits of the X, i, p registers, and n is the word size for the matrix state. For ChaCha256, for naive exhaustive search the value of n_X should be 512 and n_i, n_p should be $\log(512)$. Generally if word size is n, $n_X = 12n$, $n_i = n_p = \log(12n)$. However, if analyzed carefully some superpostions can be removed. For instance, as per the scheme of the cipher, practically we should only be allowed

to make change in the key bits and the nonce values, thus we can take $n_i = \log 12n$ and $n_p = \log 12n$ and $n_X = 12n$. Due to linearity of unitary operations, applying Algorithm 6 on these superpostion will give the state

$$\frac{1}{\sqrt{2^{n_X+n_i+n_p}}} \sum_{i\in\mathbb{F}_2^{n_X}} \sum_{p\in\mathbb{F}_2^{n_i}} \sum_{X\in\mathbb{F}_2^{n_p}} |i\rangle |p\rangle |X\rangle |0\rangle$$
$$\rightarrow \frac{1}{\sqrt{2^{n_X+n_i+n_p}}} \sum_{i\in\mathbb{F}_2^{n_X}} \sum_{p\in\mathbb{F}_2^{n_i}} \sum_{X\in\mathbb{F}_2^{n_p}} |i\rangle |p\rangle |X\rangle (-1)^{(W(X))_p} |(W(X))_p \oplus (W(X'))_p\rangle \tag{3.17}$$

Applying appropriate filtering on the above state, can help retrieve the a good $\mathcal{ID}-\mathcal{OD}$ pair. We use the algorithm given by Bera and Sapv (2023) to filter. We rewrote their PROFIL algorithm for our specific problem, as shown in Algorithm 7.

Algorithm 7 Good_ID_OD

Input: Oracle ODGEN, threshold p_t, threshold of single differential bias required in single differential calculation of ChaCha ϵ_t, accuracy of estimation $\epsilon_{est}(\epsilon_{est} < \epsilon_t)$

Output: One ID_OD pair, such that it has $\Pr(\Delta_r^r = 1|\Delta_i^0 = 1) \geq p_t$

1: **Generate superposition** over all possible i, p, X, where i is possible positions of ID, p is possible positions of OD, X is possible initial states of ChaCha cipher.
2: With oracle as $ODGen$, apply **amplitude estimation** (infact, probability estimation), and good state as $\Delta = 0$. Size of the estimate register is m-qubit. $m = \log \frac{1}{\epsilon_t} + 3$
3: Threshold register set as $a_t = \frac{2^m}{\pi} \sin^{-1} \sqrt{p_t}$. **Transforming relation** of the estimate register and threshold register using operator $N : N|a\rangle = |2^m - a\rangle$
4: **Applying threshold** of estimate register to mark flag qubit as 1.
5: Applying amplitude amplification with good state as flag$= 1$.
6: Measure and return (i, p) as Good_ID_OD pair

Theorem 3.4.1. *Given r rounds of ChaCha, n bit words Good_ID_OD algorithm 7 gives the following gaurantees:*

- *If there exists an $\mathcal{ID}-\mathcal{OD}$ pair i, p (indices) such that $\Pr(\Delta_p^{(r)} = 1|\Delta_i^{(0)} = 1) \geq \frac{1}{2} + \epsilon_t = p_t$ then the output register (flag) will be in state $|1\rangle$ with probability $\geq 0.8p_t$, and simultaneous measurement of i, p register gives a good i, p ($ID-OD$ bit pair).*
- *If there exists no i, p such that $\Pr(\Delta_p^{(r)} = 1|\Delta_i^{(0)} = 1) \geq \frac{1}{2} + \epsilon_t - \frac{1}{2^{m-3}} = p_t - \frac{1}{2^{m-3}}$ then the output register (flag) will be in state $|0\rangle$ with probability ≥ 0.8.*

where, ϵ_t=threshold error for ChaCha $\mathcal{ID}-\mathcal{OD}$ bias, m is the number of qubits required for estimate register.

Proof. We rewrite equation 3.17 for ChaCha word size of n

$$\frac{1}{\sqrt{2^{12n+\log(12n)+\log(12n)}}} \sum_{\substack{X\in\mathbb{F}_2^{12n},\\ i\in\mathbb{F}_2^{\log(12n)},\\ p\in\mathbb{F}_2^{\log(12n)}}} |i\rangle\,|p\rangle\,|X\rangle\,(-1)^{(W(X))_p}\,|(W(X))_p \oplus (W(X'))_p\rangle \tag{3.18}$$

$$= \frac{1}{\sqrt{2^{12n+\log(12n)+\log(12n)}}} \sum_{\substack{X\in\mathbb{F}_2^{12n},\\ i\in\mathbb{F}_2^{\log(12n)},\\ p\in\mathbb{F}_2^{\log(12n)}}} |i\rangle\,|p\rangle\,|X\rangle\,(-1)^{(W(X))_p}\,|\Delta_p^r\rangle \tag{3.19}$$

$$\text{where } \Delta_p^r = (W(X))_p \oplus (W(X'))_p$$

Let $\alpha_{i,p,\Delta}$ denote measure of number of X for a particular i, p, Δ. Then state can be written as

$$\sum_{i=0}^{\log(12n)-1} \sum_{p=0}^{\log(12n)-1} \alpha_{i,p,\Delta}\,|i\rangle\,|p\rangle\,|\Delta\rangle\,|0\rangle^m \tag{3.20}$$

$$= \sum_{i,p,\Delta=0} \alpha_{i,p,0}\,|i\rangle\,|p\rangle\,|0\rangle\,|0\rangle^m + \sum_{i,p,\Delta=1} \alpha_{i,p,1}\,|i\rangle\,|p\rangle\,|1\rangle\,|0\rangle^m \tag{3.21}$$

with an addition m-qubit register which will be used for amplitude estimation. We apply amplitude estimation (Brassard *et al.*, 2002) on this with good state as $\Delta = 0$. Resultant state is:

$$\sum_{i,p,\Delta=0} \alpha_{i,p,0}\,|i\rangle\,|p\rangle\,|0\rangle\,(\beta_{i,p,0,s}\,|\widetilde{a_{i,p,0}}\rangle + \beta_{i,p,0,\bar{s}}\,|\widetilde{E_{i,p,0}}\rangle) + \sum_{i,p,\Delta=1} \alpha_{i,p,1}\,|i\rangle\,|p\rangle\,|1\rangle\,|0\rangle^m \tag{3.22}$$

Actual probability $= \left|\alpha_{i,p,0}^2\right| = p_{i,p,0}$ (say). Probability observed $= \sin^2 \frac{\widetilde{a_{i,p,0}}}{2^m} = \widetilde{p_{i,p,0}}$ (say). Here

$$\beta_{i,p,0,s} \geq \frac{8}{\pi^2} \tag{3.23}$$

$$\beta_{i,p,0,\bar{s}} \leq 1 - \frac{8}{\pi^2} \tag{3.24}$$

$$|\widetilde{p_{i,p,0}} - p_{i,p,0}| \leq \frac{1}{2^{m-3}} \tag{3.25}$$

Estimate of number of qubits m For all i, p such that $p_{i,p,0} \geq p_t = \frac{1}{2} + \epsilon_t$ flag

register should be $|1\rangle$ and accuracy $\epsilon_{est} = |\widetilde{p_{i,p,0}} - p_{i,p,0}| \leq \frac{1}{2^{m-3}}$.
We want, $\epsilon_{est} < \epsilon_t$, so that, if $p_{i,p,0} > \frac{1}{2} + \epsilon_t$, then even in th case of worst accuracy (e_{est} close to e_t),

$$\widetilde{p_{i,p,0}} = p_{i,p,0} - \epsilon_{est} = \frac{1}{2} + \epsilon_t - \epsilon_{est} > \frac{1}{2} \tag{3.26}$$

Therefore,

$$\epsilon_{est} = \frac{1}{2^{m-3}} < \epsilon_t \tag{3.27}$$

$$\implies m - 3 > \log \frac{1}{\epsilon_t} \tag{3.28}$$

$$\implies m > \log \frac{1}{\epsilon_t} + 3 \tag{3.29}$$

Transforming relation: In order to apply threshold to the algorithm we first need to transform the amplitude appropriately. We have $\widetilde{a_{i,p,0}}$ in register and a_t should be input. They are related to $\widetilde{p_{i,p,0}}$ and p_t as follows:

$$\widetilde{p_{i,p,0}} = \sin^2\left(\pi \frac{\widetilde{a_{i,p,0}}}{2^m}\right) \tag{3.30}$$

$$p_t = \sin^2\left(\pi \frac{a_t}{2^m}\right) \implies a_t = \frac{2^m}{\pi} \sin^{-1}\sqrt{p_t} \tag{3.31}$$

For $\widetilde{p_{i,p,0}} \geq p_t$ we want to mark flag $= |1\rangle$.

$$\widetilde{p_{i,p,0}} \geq p_t \tag{3.32}$$

$$\implies \sin^2\left(\pi \frac{\widetilde{a_{i,p,0}}}{2^m}\right) \geq \sin^2\left(\pi \frac{a_t}{2^m}\right) \tag{3.33}$$

$$\implies \left|\frac{\pi}{2} - \frac{\pi \widetilde{a_{i,p,0}}}{2^m}\right| \leq \left|\frac{\pi}{2} - \frac{\pi a_t}{2^m}\right| \tag{3.34}$$

$$\implies \left|2^{m-1} - \widetilde{a_{i,p,0}}\right| \leq \left|2^{m-1} - a_t\right| \tag{3.35}$$

From the state in 3.22 to obtain state in the form $||2^{m-1} - \widetilde{a_{i,p,0}}|\rangle$ we use a negation operator $N : N|a\rangle \rightarrow |2^m - a\rangle$, where m is the size of the register. Applying N-operator on amplitude register of 3.22, and adding two registers for threshold and flag,

the state becomes:

$$\sum_{i,p,0} \alpha_{i,p,0} \left|i,p,0\right\rangle \left(\beta_{i,p,0,s} \left|2^{m-1} - \widetilde{a_{i,p,0}}\right\rangle + \beta_{i,p,0,\bar{s}} \left|2^{m-1} - \widetilde{E_{i,p,0}}\right\rangle\right) \left|2^m - a_t\right\rangle \left|0\right\rangle$$
$$+ \sum_{i,p,1} \alpha_{i,p,1} \left|i,p,1\right\rangle \left|0^m\right\rangle \left|2^m - a_t\right\rangle \left|0\right\rangle \tag{3.36}$$

In further steps we will substitute $|2^{m-1} - x\rangle$ as $\hat{x}$ for the convenience of notation. For marking the flag qubit, let us consider the unitary F

$$F(\left|x\right\rangle \left|y\right\rangle \left|b\right\rangle) = \left|x\right\rangle \left|y\right\rangle \left|b \oplus f(x,y)\right\rangle \tag{3.37}$$

For $f(x,y)$ we consider the following function

$$f(x,y) = \begin{cases} 1 & , \text{ if } x \leq y \\ 0 & , \text{ otherwise} \end{cases} \tag{3.38}$$

For any unitary we can implement the controlled version of it. We apply controlled unitary F on 4-th, 5-th and 6-th register, with 3-rd register as the control. The state becomes

$$\sum_{i,p,0} \alpha_{i,p,0} \left|i,p,0\right\rangle \Bigg\{ \beta_{i,p,0,s} \left|\widehat{a_{i,p,0}}\right\rangle \left|\widehat{a}_t\right\rangle \left|f(\widehat{a_{i,p,0}}, \widehat{a}_t)\right\rangle$$
$$+\beta_{i,p,0,\bar{s}} \Bigg(\sum_{\hat{E} > \hat{a}_t} \gamma_{i,p,\hat{E}} \left|\widehat{E_{i,p,0}}\right\rangle \left|\widehat{\alpha}_t\right\rangle \left|0\right\rangle + \sum_{\hat{E} \leq \hat{a}_t} \gamma_{i,p,\hat{E}} \left|\widehat{E_{i,p,0}}\right\rangle \left|\widehat{\alpha}_t\right\rangle \left|1\right\rangle \Bigg)$$
$$+ \sum_{i,p,1} \alpha_{i,p,1} \left|i,p,1\right\rangle \left|0^m\right\rangle \left|\widehat{a}_t\right\rangle \left|0\right\rangle \Bigg\} \tag{3.39}$$

Rewriting the above state omitting the $\hat{a}_t$ register, since we would not be needing it anymore.

$$\sum_{i,p,0} \alpha_{i,p,0} \left|i,p,0\right\rangle \Bigg\{ \beta_{i,p,0,s} \left|\widehat{a_{i,p,0}}\right\rangle \left|f(\widehat{a_{i,p,0}}, \widehat{a}_t)\right\rangle$$
$$+\beta_{i,p,0,\bar{s}} \Bigg(\sum_{\hat{E} > \hat{a}_t} \gamma_{i,p,\hat{E}} \left|\widehat{E_{i,p,0}}\right\rangle \left|0\right\rangle + \sum_{\hat{E} \leq \hat{a}_t} \gamma_{i,p,\hat{E}} \left|\widehat{E_{i,p,0}}\right\rangle \left|1\right\rangle \Bigg)$$
$$+ \sum_{i,p,1} \alpha_{i,p,1} \left|i,p,1\right\rangle \left|0^m\right\rangle \left|0\right\rangle \Bigg\} \tag{3.40}$$

The last register is the flag register. Considering the following two cases:

- **Case 1:** $\forall(i,p), p_{i,p,0} < p_t - \epsilon_{est}$. The state will be:

$$\sum_{i,p,0} \alpha_{i,p,0} |i,p,0\rangle \Bigg\{ \beta_{i,p,0,s} |\widehat{a_{i,p,0}}\rangle |0\rangle + \beta_{i,p,0,\bar{s}} \Bigg(\sum_{\hat{E}>\hat{a_t}} \gamma_{i,p,\hat{E}} |\widehat{E_{i,p,0}}\rangle |0\rangle + \sum_{\hat{E}\leq\hat{a_t}} \gamma_{i,p,\hat{E}} |\widehat{E_{i,p,0}}\rangle |1\rangle \Bigg) + \sum_{i,p,1} \alpha_{i,p,1} |i,p,1\rangle |0^m\rangle |0\rangle \Bigg\} \tag{3.41}$$

  In this case there exists no (i,p) such that, $p_{i,p,0} < p_t - e_est$. However, from equation 3.41, we see some of the estimates $\widehat{E_{i,p,0}}$ might be marked as 1, this will cause an error, which we can bound as follows:
  Since, from amplitude estimation, $|\beta_{i,p,0,\bar{s}}|^2 < 1 - \frac{8}{\pi}^2$
  $\Pr[\text{ flag } = 1] = \left|\alpha_{i,p,0}.\beta_{i,p,0,\bar{s}}.\gamma_{i,p,\hat{E}}\right|^2 < 1 - \frac{8}{\pi^2}$
  Therefore, probability that all the flag registers are 0 in Case 1, is given by: $\Pr[\text{ flag } = 0] \geq \frac{8}{\pi^2}$

- **Case 2:** $\exists(i,p), p_{i,p,0} \geq p_t$. The state will be:

$$\sum_{i,p\in good} \alpha_{i,p,0} |i,p,0\rangle \Bigg\{ \beta_{i,p,0,s} |\widehat{a_{i,p,0}}\rangle |1\rangle \tag{3.42}$$

$$+\beta_{i,p,0,\bar{s}} \Bigg(\sum_{\hat{E}>\hat{a_t}} \gamma_{i,p,\hat{E}} |\widehat{E_{i,p,0}}\rangle |0\rangle \tag{3.43}$$

$$+ \sum_{\hat{E}\leq\hat{a_t}} \gamma_{i,p,\hat{E}} |\widehat{E_{i,p,0}}\rangle |1\rangle \Bigg) \Bigg\} \tag{3.44}$$

$$+ \sum_{i,p\notin good} \alpha_{i,p,0} |i,p,0\rangle \Bigg\{ \beta_{i,p,0,s} |\widehat{a_{i,p,0}}\rangle |0\rangle \tag{3.45}$$

$$+\beta_{i,p,0,\bar{s}} \Bigg(\sum_{\hat{E}>\hat{a_t}} \gamma_{i,p,\hat{E}} |\widehat{E_{i,p,0}}\rangle |0\rangle \sum_{\hat{E}\leq\hat{a_t}} \gamma_{i,p,\hat{E}} |\widehat{E_{i,p,0}}\rangle |1\rangle \Bigg) \Bigg\} \tag{3.46}$$

$$+ \sum_{i,p,1} \alpha_{i,p} |i,p,1\rangle |0^m\rangle |0\rangle \Bigg\} \tag{3.47}$$

$$\tag{3.48}$$

  Since, $|\beta_{i,p,0,s}|^2 \geq \frac{8}{\pi}^2$ and $\alpha_{i,p,0} = p_{i,p,0} \geq p_t$
  $\therefore \Pr[\text{ flag } = 1 | (i,p) \in good] = |\alpha_{i,p,0}.\beta_{i,p,0,s}|^2 \geq \frac{8}{\pi^2}.p_t$

□

3.4.1 Complexity analysis

Table 3.1 gives the comparison of the time and space complexity of finding Good ID-OD pair using classical exhaustive algorithm and quantum algorithm 7.

Ideally, in classical we would need to exhaustively compute for all possible ID, OD and initial state X, the probabilities of ID-OD associated. For this we would require $O\left(n^2\right)$ space and $O\left(n^2 \cdot 2^{12n}\right)$ time, wheren is the word size of ChaCha cipher matrix state.

In our quantum algorithm, the estimate of probability should have accuracy $O\left(\frac{1}{2^{12n}}\right)$ in the worst case, so we put ϵ_{est} as this. Since, we are using the PROFIL algorithm [Bera and Sapv (2023)], the number of oracle calls to estimate probability with ϵ accuracy is $O(\frac{1}{\epsilon})$. Therefore, here for probability estimate accuracy of $O\left(\frac{1}{2^{12n}}\right)$ we will need $O(2^{12n})$ oracle calls and qubits needed is $O\left(n + \log \frac{1}{\epsilon_{est}}\right) = O(n)$. Also, here ϵ_{est} is upper bounded by ϵ_t, which is threshold for single differential bias of ChaCha

	Classical	**Quantum**
Time	$O\left(n^2 \cdot 2^{12n}\right)$	$O\left(\frac{1}{\epsilon_{est}}\right) = O\left(2^{12n}\right)$
Space	$O\left(n^2\right)$	$O\left(n + \log \frac{1}{\epsilon_{est}}\right) = O\left(n\right)$

Table 3.1: Comparison of complexity of Classical and quantum algorithm for find good ID-OD pairs for ChaCha cipher, where ϵ_{est} is the probability estimate accuracy, and is upper bounded by ϵ_t which is threshold for single differential bias of ChaCha

3.5 Finding Probabilistic Neutral Bits

Algorithm 8 gives a register value that can be use to get the neutrality measure 3.3. It is quantum parallel of the Algorithm 2. Steps $2-7$ obtain the Δ value in the 4-th register, and steps $8-22$ obtain the Δ' value in the 3-rd register. The last step is to obtain $\Delta \oplus \Delta'$ in a register.

Algorithm 8 consists of a lot of steps, including subtraction circuits which can be quite large, thus using this as an oracle is not desirable. Unlike the Algorithm 6 which was smaller and we could as an oracle in with amplitude estimation, we cannot apply the

Algorithm 8 Quantum Getting PNBs

Input: Bit for input difference i, bit for output difference p, bit for reverse round output difference l. In existing cryptanalysis of ChaCha cipher $l = p$.

Output: The value of $\Delta_p^r \oplus \Gamma_p^r$ in the 4-th register. (For notation refer step 6 of algorithm 2) In the algorithm, $\Delta = \Delta_p^r$ and $\Delta' = \Gamma_p^r$.

1: Take 7 registers. First two registers initialised as intital state X, 3-rd register for Δ', 4-th register for Δ, 5-th register for input difference bit i, 6-th register for output different bit p, 7-th register for l bit
   ▷ $|X\rangle\,|X\rangle\,|0\rangle\,|0\rangle$, 5th,6th,7th registers not shown since they remain constant
2: Hadamard gate on 4-th register and $CNOT$ on i-th bit in 2-nd register, controlled on 4-th register
3: ChaCha r-rounds on 2nd register
4: Z-gate on 2-nd register at p bit
5: ChaCha reverse r-rounds on 2-nd register
6: $CNOT$ on i-th bit in 2-nd register, controlled on 4-th register
7: Hadamard gate on 4-th register
   ▷ $(-1)^{W_p}\,|X\rangle\,|X\rangle\,|0\rangle\,|\Delta\rangle$, where W is the state after r-rounds of ChaCha.
8: Hadamard gate on 3-rd register
9: $CNOT$ on i-th bit in 1-st register and 2-nd register, controlled on 3-rd register
10: X-gate on l-th bit in 2-nd register
11: 2-nd register = 1-st register - 2-nd register (We can use method of Cuccaro *et al.* (2004*b*) for this).
   ▷ $(-1)^{W_p}\Big[|X\rangle\,|X-\bar{X}\rangle\,|0\rangle + |X'\rangle\,|X'-\bar{X}'\rangle\,|1\rangle\Big]\,|\Delta\rangle$
12: ChaCha R-rounds on 1st register
13: 1-st register = 1-st register + 2-nd register
14: ChaCha reverse $(R-r)$-rounds on 1-st register
15: Z-gate on 1-st register at p bit
16: ChaCha $(R-r)$-rounds on 1-st register
17: 1-st register = 1-st register - 2-nd register
18: ChaCha reverse R-rounds on 1st register
19: 2-nd register = 1-st register - 2-nd register
20: X-gate on l-th bit in 2-nd register
21: $CNOT$ on i-th bit in 1-st and 2-nd register, controlled on 3-rd register
22: Hadamard gate on 3-rd register
23: $CNOT$ on 4-th register controlled on 3-rd register.
   ▷ $(-1)^{W_p}(-1)^{(Y_p)}\,|X\rangle\,|X\rangle\,|\Delta'\rangle\,|\Delta'\oplus\Delta\rangle$, where Y is the state as described in Algorithm 2
24: The resulting state is the output

same to this algorithm. This lead us to exploring more generic and simpler cryptanalysis techniques, as discussed in the next chapter.

CHAPTER 4

Generic linear quantum cryptanalysis

4.1 Linear Cryptanalysis

Linear cryptanalysis was first introduced by Matsui as a theoretical attack and later successfully used in the practical cryptanalysis of DES (Matsui, 1994). Linear cryptanalysis tries to take advantage of high probability occurrences of linear expressions involving plaintext bits, ciphertext bits and subkey bits. It is a known plaintext attack, that is, it is premised on the attacker having information on a set of plaintexts and the corresponding ciphertexts Heys (2002).

A linear expression is of the following form

$$X_{i_1} \oplus X_{i_2} \ldots X_{i_u} \oplus Y_{i_1} \oplus Y_{i_2} \ldots Y_{i_v} = 0 \tag{4.1}$$

where X_i represents the i-th bit of the input $X = [X_1, X_2, \ldots]$ and Y_j represents the j-th bit of the output $Y = [Y_1, Y_2, \ldots]$, and $\oplus$ denotes the XOR operation.

Let us assume an arbitrary function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ such that $f(X) = Y$, i.e. X is n bit and Y is m bit. For $f(X)$ if we can find of the form 4.1 that holds with high or low probability, it means the function $f(X)$ can be approximated as linear expression. Presence of such strong linear expressions in ciphers is a weakness, since these can be used to launch attacks.

Equation 4.1 can be re-written using the concept of masks

$$\mu_{1_1} X_1 \oplus \mu_{1_2} X_2 \oplus \ldots \mu_{1_n} X_n \oplus \mu_{2_1} Y_1 \oplus \mu_{2_2} Y_2 \oplus \ldots \mu_{2_m} Y_m = 0 \tag{4.2}$$

$$\mu_1.X \oplus \mu_2.Y = 0 \tag{4.3}$$

here μ_1, μ_2 are the masks of X and Y respectively. $\mu_1 \in \mathbb{F}_2^n$ and $\mu_2 \in \mathbb{F}_2^n$, and $\mu_1.X$ represents the dot product of μ_1 and X. If μ_i is 1, it implies X_i is part of the linear expression.

The amount by which the probability of a linear expression holding deviates from $\frac{1}{2}$ is known as the **linear probability bias**. That is, if $\Pr(\text{some event}) = 1/2 + \epsilon$ then the linear probability bias is ϵ.

4.1.1 Process of Linear Cryptanalysis

The aim is to apply linear cryptanalysis on a cipher. Most components consist of smaller components, like S-boxes, P-boxes or simply repititive rounds of a particular operation. The idea is to find linear approximations for these smaller components, and then to find a trail to launch an attack on the complete cipher.

Just like any arbitrary function in the previous section, we can consider these cipher components to be functions $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ such that $f(X) = Y$. Classically, to find linear approximation a **Linear Approximation Table (LAT)** is used. The features of the LAT are:

- The rows are columns represent the masks μ_1, μ_2 respectively.
- Each cell gives a measure proportional to the linear probability bias, of that particular input and output mask. To record this, for every input for which it holds we add 1 and if it doesn't hold we subtract 1. Let us call this value COUNT. Finally, (2^{n+m-1}) is subtracted from each COUNT. This is the linear approximation table. If these resultant values are divided by 2^{n+m}, we get the exact value of linear probability bias.
  Alternatively, we could divide COUNT by 2^{n+m} and subtract $\frac{1}{2}$, it would still give us the linear probability bias.

$$\frac{\text{COUNT} - 2^{n+m-1}}{2^{n+m}} = \frac{\text{COUNT}}{2^{n+m}} - \frac{1}{2} \tag{4.4}$$

- The sum of any row or column will be 2^{n+m-1} Heys (2002).

Piling up Principle

Let two random variables X_1, X_2 have the following probability distribution:

$$\Pr(X_1 = i) = \begin{cases} p_1 & , i = 0 \\ 1 - p_1 & , i = 1 \end{cases} \quad \Pr(X_2 = i) = \begin{cases} p_2 & , i = 0 \\ 1 - p_2 & , i = 1 \end{cases} \tag{4.5}$$

Let $p_1 = 1/2 + \epsilon_1$ and $p_2 = 1/2 + \epsilon_2$, Then,

$$\begin{aligned} p_1 p_2 =& \frac{1}{4} + \frac{1}{2}\epsilon_1 + \frac{1}{2}\epsilon_2 + \epsilon_1\epsilon_2 \\ \Pr(X_1 \oplus X_2 = 0) =& \Pr(X_1 = 1, X_2 = 1) + \Pr(X_1 = 1, X_2 = 1) \\ =& p_1(p_2) + (1 - p_1)(1 - p_2) \\ =& p_1 p_2 + 1 - p_1 - p_2 + p_1 p_2 \\ =& 1 - p_1 - p_2 + 2p_1 p_2 \\ =& 1 - \frac{1}{2} - \epsilon_1 - \frac{1}{2} - \epsilon_2 + \frac{1}{2} + \epsilon_1 + \epsilon_2 + 2\epsilon_1\epsilon_2 \\ =& 1/2 + 2\epsilon_1\epsilon_2 \end{aligned} \tag{4.6}$$

Matsui (1994) gave the piling up lemma as follows: For n independent, random binary variables, $X_1, X_2, \ldots X_n$,

$$\Pr(X_1 \oplus \ldots \oplus X + n = 0) = 1/2 + 2^{n-1}\Pi_{i=1}^{n}\epsilon \tag{4.7}$$

or, equivalently,

$$\epsilon_{1,2,\ldots n} = 2^{n-1}\Pi_{i=1}^{n}\epsilon \tag{4.8}$$

where $\epsilon_{1,2,\ldots n}$ represents the bias of $X_1 \oplus \ldots \oplus X + n = 0$. If $p_i = 0$ or 1 for all i, then $\Pr(X_1 \oplus \ldots \oplus X + n = 0) = 0$ or 1. If only one $p_i = 1/2$, then $\Pr(X_1 \oplus \ldots \oplus X + n = 0) = 1/2$.

Constructing Linear Approximations for the Complete Cipher

Once the linear approximation information has been compiled for the cipher components, we have the data to proceed with determining linear approximations of the overall cipher of the form of equation 4.1. This can be achieved by **concatenating appropriate linear approximations** of the components.

4.2 Finding Quantum Linear Approximations

In this section we discuss a quantum algorithm we have designed to obtain linear approximations from ciphers. Let us Boolean map $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let τ be some threshold $\in [0, 1]$. We want to identify the best mask $\mu = (\mu_1, \mu_2)$.
Let $z : \mathbb{F}_2^n \to \mathbb{F}_2^{n+m}$ such that $z(x) = (x, f(x))$

Definition 4.2.1 (Corr)**.** Measure of correlation:
$Corr(\mu) = Corr((\mu_1, \mu_2)) = \Pr_x[x.\mu_1 = f(x).\mu_2] = \Pr_x[z(x).\mu = 0]$

Definition 4.2.2 (Corr')**.** $Corr'(\mu) = Corr(\mu) - \frac{1}{2}$

Alternatively we can express $Corr(\mu)$ as follows:

$$Corr(\mu) = \Pr_x[z(x).\mu = 0] = \frac{n_1}{2^n} \text{ where } n_1 = |x : z(x).\mu = 0| \tag{4.9}$$

$$\implies Corr(\mu) - \frac{1}{2} = \frac{n_1}{2^n} - \frac{1}{2} = \frac{(2n_1 - 2^n)}{2^{n+1}} \tag{4.10}$$

$$= \frac{n_1 - n_2}{2^{n+1}} \text{ where } n_2 = |x : z(x).\mu \neq 0| \tag{4.11}$$

$$= \frac{\widetilde{z(\mu)}}{2} \tag{4.12}$$

$$\text{where, } \widetilde{z(\mu)} = \frac{1}{2^n} \sum_x (-1)^{z(x).\mu} = \frac{1}{2^n}[n_1 - n_2] \tag{4.13}$$

Few observations on $\widetilde{z(\mu)}$ and good μ

$$\left|\widetilde{z(\mu)}\right|^2 = \frac{1}{2^n} + \frac{1}{4^n} \sum_{(} x \neq y)(-1)^{\mu.(z(x)+z(y))} \tag{4.14}$$

$$\sum_{\mu \in \mathbb{F}_2^{2n}} \widetilde{z(\mu)}^2 = \frac{4^n}{2^n} + \frac{1}{4^n} \sum_{x \neq y} \sum_{\mu \in \mathbb{F}_2^{2n}} (-1)^{\mu.(z(x)+z(y))}$$

$$= 0 \text{ [since } z(x) + z(y) \neq 0 \therefore \text{ if } x \neq y] \tag{4.15}$$

$$\frac{1}{2^n} \sum_\mu \widetilde{z(\mu)}^2 = 1 \tag{4.16}$$

From equation 4.16 we can say that $\frac{1}{2^n} \sum_\mu \widetilde{z(\mu)} \left|\mu\right\rangle = \frac{1}{2^n} \sum_\mu 2.Corr'(\mu) \left|\mu\right\rangle$ is a valid quantum state.

Definition 4.2.3 (Good μ)**.** μ is good if $|Corr(\mu) - \frac{1}{2}| \geq \tau \Leftrightarrow Corr'(\mu) \geq \tau$
$\Leftrightarrow \widetilde{z(\mu)} \geq 2\tau$

Claim: Let N be number of samples required for getting good μ. μ is good if

$$Corr'(\mu) \geq \tau \tag{4.17}$$

$$\Leftrightarrow Corr'(\mu)^2 \geq \tau^2 \tag{4.18}$$

$$\Leftrightarrow \frac{4}{2^n} Corr'(\mu)^2 \geq \frac{4}{2^n}\tau^2 \tag{4.19}$$

To ensure that estimated of $Corr'(\mu)$ are within $\pm\epsilon$, we choose

$$N = O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right) \text{ where } \epsilon = \frac{4\tau^2}{2^n}.\frac{1}{100} \tag{4.20}$$

$$= O\left(\frac{10^4.2^{2n}}{16\tau^4}\log\frac{1}{\delta}\right) \tag{4.21}$$

For a **good cipher**, that is close to random function, the distribution will be almost uniform, i.e., for any μ

$$\frac{4Corr'(\mu)^2}{2^n} = \frac{1}{2^{2n}} \tag{4.22}$$

$$\Leftrightarrow Corr'(\mu) = \frac{1}{4.2^n} \tag{4.23}$$

$$\Leftrightarrow Corr'(\mu) = \frac{1}{2.\sqrt{2^n}} \tag{4.24}$$

If $\tau \approx \Theta(\frac{1}{\sqrt{2^n}})$, number of samples

$$N \approx \widetilde{O}\left(\frac{10^4.2^{2n}}{16}.2^{2n}\right) \approx \widetilde{O}(2^{4n}) \tag{4.25}$$

Thus if a cipher is good, the number of samples to find linear approximation is $\widetilde{O}(2^{4n})$, which is very high. The comparison between classical and quantum approach given by us, is shown in table 4.1.

Algorithm 9 is a quantum algorithm to get a superposition of all possible masks μ weighted by the $\widetilde{z(\mu)}$ values. The cipher is taken to be Function (cipher) $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$. The circuit corresponding to the algorithm is shown in 4.1.

Algorithm 9 Quantum Linear Approx

Input: Quantum oracle to access function f is $U_f|x\rangle|0^m\rangle \longrightarrow |x\rangle|f(x)\rangle$

Output: Superposition of all μ weighted by the respective $\widetilde{z(\mu)}$ values

1: Take 2 registers of size n and m intialized as $|0\rangle$

$\triangleright\ |0\rangle^n\,|0\rangle^m$

2: Apply Hadamard gate $(H^{\otimes n})$ on first register

$\triangleright\ \sum_{x\in F_2^n} \frac{1}{\sqrt{2^n}}|x\rangle|0^m\rangle$

3: Apply U_f oracle.

$\triangleright\ \sum_{x\in F_2^n} \frac{1}{\sqrt{2^n}}|x\rangle|f(x)\rangle = \sum_{x\in F_2^n} \frac{1}{\sqrt{2^n}}|z(x)\rangle$

4: Apply Hadamard on both registers $(H^{\otimes n+m})$

$$\triangleright \sum_{x\in F_2^n} \frac{1}{\sqrt{2^n}} \sum_{\mu\in F_2^{n+m}} \frac{1}{\sqrt{2^{n+m}}}(-1)^{z(x)\cdot\mu}|\mu\rangle$$
$$= \sum_{\mu\in F_2^{n+m}} \frac{2.Corr'(\mu)}{\sqrt{2^n}}|\mu\rangle = \sum_{\mu\in F_2^{n+m}} \frac{\widetilde{z(\mu)}}{\sqrt{2^n}}\,|\mu\rangle$$

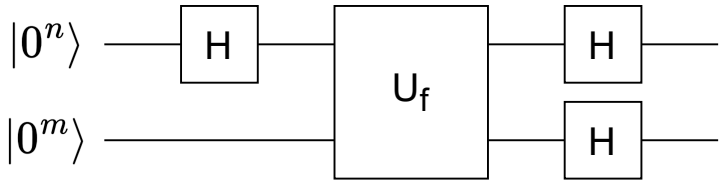

Figure 4.1: Finding Linear approximation Quantum Circuit

4.3 Classical Filtering

Once we get the superposition of all μ weighted by the respective $\widetilde{z(\mu)}$ values from Algorithm 9, we want to retrieve the μ's with highest $\widetilde{z(\mu)}$. The naive way to do this would be to take $N = \widetilde{O}(2^{4n})$ samples, store and process them to find the most frequently occurring elements. Besides the time complexity, this would result in very high space complexity of $O(2^{4n} \times \log f_{max})$, where f_max is frequency of μ with largest frequency. For any non-trivial cipher component the n is expected to be moderately large, and thus this approach would not be feasible. However we can use algorithms for Heavy Hitters, to do this in lesser space complexity.

For understanding, approximately time complexity 2^{30} and space complexity 2^{40} can be supported by computers today. For 2^{30} space and time, the naive algorithm would work. This means for $n = 8$, algorithm will work, but for non-trivial cipher components we would want $n > 8$. Like for ChaCha, for one ARX and one quarter-round we would need $n = 96$ and $n = 128$ respectively.

4.3.1 Heavy Hitters Algorithm

Heavy Hitters Problem: Given a stream of items, find those items that occur with most frequency. It has been shown that counter-based methods are the most space-efficient (Berinde and Indyk). Two types of counter-based methods are used, namely, FREQUENT and SPACESAVING algorithm. The algorithms are given in Algorithm 10 and Algorithm 11, respectively.

Lemma 4.3.1. *Both algorithms* FREQUENT *and* SPACESAVING *for space complexity* $O(k/\epsilon)$ *give an error bound of* $|f_i - \hat{f}_i| \leq \epsilon/k.F_1^{res(k)}$*, where* f_i *is the actual frequency and* $\hat{f}_i$ *is the estimated frequency and* $F_1^{res(k)}$ *is the residual tail of the stream, i.e., the sum of the frequencies of all elements other than the* k *most frequent ones (heavy hitters) (Berinde and Indyk).*

Algorithm 10 FREQUENT

$T = \emptyset$
for each i **do**
    if $i \in T$ **then**
        $c_i = c_i + 1$
    else if $|T| < m$ then **then**
        $T = T \cup i$
        $c_i = 1$
    else
        for all $j \in T$ **do** $c_j = c_j - 1$
            if $c_j = 0$ **then**
                $T = T \backslash \{j\}$
            end if
        end for
    end if
end for

4.4 Quantum Filtering

The output of Algorithm 9 is $\sum_{\mu \in F_2^{n+m}} \frac{\widetilde{z(\mu)}}{\sqrt{2^n}} |\mu\rangle$. From this state, we want to retrieve all μ such that $\widetilde{z(\mu)}$ is above a certain threshold, i.e., we want to find the masks which have a high linear probability bias. For this we will use the techniques in Bera and Sapv (2023).

Algorithm 11 SPACESAVING

```
T = ∅
for each i do
    if i ∈ T then
        c_i = c_i + 1
    else if |T| < m then
        T = T ∪ i
        c_i = 1
    else
        j = arg min_{j∈T} c_j
        c_i = c_j + 1
        T = T ∪ {i}\{j}
    end if
end for
```

4.4.1 Estimating $\widetilde{z(\mu)}$

EQAMPEST is a modification of the Quantum Amplitude Estimation Algorithm given by Brassard *et al.* (2002) that takes the good state as input register.

Lemma 4.4.1. *Given oracle $A : A|0^l\rangle = |\psi\rangle$, EQAmpEst$_A$ implements the following operation:*

$$\text{EQAMPEST}_A \sum_x \alpha_x |x\rangle |00\ldots0\rangle |0\rangle^m) \rightarrow$$
$$\sum_x \alpha_x \beta_{x,s} |x\rangle |\psi\rangle |\widehat{p_x}\rangle + \sum_x \alpha_x \beta_{x,\bar{s}} |x\rangle |\psi\rangle |\widehat{E_x}\rangle \tag{4.26}$$

where $p_x = |\langle\psi|x\rangle|^2$ = probability of measuring $|x\rangle$ when $|\psi\rangle$ is measured. On measuring the first and third registers, outcome pair $|x\rangle |\widehat{p_x}\rangle$ is obtained with probability $|\alpha_x\beta_{x,s}|^2 \geq \frac{8}{\pi^2}|\alpha_x|^2$ where $\sin^2(\pi\frac{\widehat{p_x}}{2^m}) = \widetilde{p_x}$ and $|\widetilde{p_x} - p_x| \leq \frac{1}{2^{m-3}}$.

In our case, $A|0\rangle^{n+m} = |\psi\rangle = \sum_{\mu\in F_2^{n+m}} \frac{\widetilde{z(\mu)}}{\sqrt{2^n}} |\mu\rangle$. Let, the probability of obtaining μ on measuring ψ be p_μ, i.e, $p_\mu = |\langle\psi|\mu\rangle|^2 = \frac{\widetilde{z(\mu)^2}}{2^n}$. On applying EQAMPEST over an equal superposition of all μ, the equation 4.26 will be as follows:

$$\text{EQAMPEST}_A \sum_\mu \frac{1}{\sqrt{2^{n+m}}} |\mu\rangle |00\ldots0\rangle |0\rangle^k) \rightarrow$$
$$\sum_\mu \frac{1}{\sqrt{2^{n+m}}} \beta_{\mu,s} |\mu\rangle |\psi\rangle |\widehat{p_\mu}\rangle + \sum_\mu \frac{1}{\sqrt{2^{n+m}}} \beta_{\mu,\bar{s}} |\mu\rangle |\psi\rangle |\widehat{E_\mu}\rangle \tag{4.27}$$

Here, $|\mu\rangle\,|\widehat{p_\mu}\rangle$ is obtained with probability $|\frac{1}{\sqrt{2^{n+m}}}\beta_{\mu,s}|^2 \geq \frac{8}{2^{n+m}\pi^2}$. The estimate of p_μ is given by the register $|\widehat{p_\mu}\rangle$, i.e., $\widetilde{p_\mu} = \sin^2(\pi\frac{\widehat{p_\mu}}{2^k})$ and $|\widetilde{p_\mu} - p_\mu| \leq \frac{1}{2^{k-3}}$. Let the corresponding estimate of $\widetilde{z(\mu)}$ be $\widetilde{z_{est}(\mu)}$. The equations are shown in 4.28 Let k be the number of bits required to estimate probability. Let the accuracy of the estimate be $epsilon$.

$$|\widetilde{p_\mu} - p_\mu| \leq \frac{1}{2^{k-3}} \text{ and } p_\mu = \frac{\widetilde{z(\mu)}^2}{2^n} \text{ and } \widetilde{p_\mu} = \frac{\widetilde{z_{est}(\mu)}^2}{2^n} \tag{4.28}$$

$$\therefore |\widetilde{z_{est}(\mu)}^2 - \widetilde{z(\mu)}^2| \leq \frac{2^n}{2^{k-3}} \text{ with probability } \geq \frac{8}{2^{n+m}\pi^2} \tag{4.29}$$

$$\text{Also, } \frac{1}{2^{k-3}} = \epsilon \tag{4.30}$$

4.4.2 Marking by Threshold

From the previous section, the state we have is:

$$\sum_\mu \frac{1}{\sqrt{2^{n+m}}}\beta_{\mu,s}\,|\mu\rangle\,|\widehat{p_\mu}\rangle + \sum_\mu \frac{1}{\sqrt{2^{n+m}}}\beta_{\mu,\bar{s}}\,|\mu\rangle\,|\widehat{E_\mu}\rangle \tag{4.31}$$

The register $|\psi\rangle$ has been removed from the notation since it will not be affecting the further steps. We want to mark all states for which $\widetilde{p_\mu}$ is above certain threshold τ. For this part, we use a modification of PROFIL algorithm in Bera and Sapv (2023).

Lemma 4.4.2. *For the* PROFIL *filtering problem:*
Input: $(\log(m)+a)$*-qubit oracle* O_D *such that* $O_D\,|0^{\log(m)+a}\rangle = \sum_{i=1}^{m}\sqrt{p_i}\,|i\rangle\,|0\rangle^a$*, and threshold* τ
For any choice of parameters $0 < \epsilon < \tau$ *for additive accuracy and* δ *for error, there exists a quantum algorithm that uses* $O((\log(m) + \log(\frac{1}{\epsilon}) + a)\log(\frac{1}{\delta_t}))$ *qubits and makes* $O(\frac{1}{\epsilon\sqrt{\tau}}\log\frac{1}{\delta_\tau})$ *queries to* O_D
such that when its final state is measured in the standard basis, we observe the following:

1. *If* $p_x < \tau - \epsilon$ *for all* x *then the output register is observed in the state* $|0\rangle$ *with probability at least* $1-\delta$
2. *If* $p_x \geq \tau$ *for any* x*, then with probability at least* $1-\delta$ *the output register is observed in the state* $|1\rangle$ *and some* x *such that* $p_x \geq \tau$ *is returned as output.*

The algorithm we use for marking is the same as in PROFIL, but we give a different

gaurantee for part 1 of 4.4.2. The theorem is given in 4.4.3. We use the oracle U_f for marking 4.28.

$$U_f |x\rangle |b\rangle = |x\rangle |b \oplus f(x)\rangle \tag{4.32}$$

$$\text{where, } f(x) = \begin{cases} 1 & , x \geq \tau \\ 0 & , x < \tau \end{cases} \tag{4.33}$$

Theorem 4.4.3. *For the choice of parameters $0 < \epsilon < \tau$, we can give the following gaurantee after the marking by threshold τ:*

1. *If there exists t number of μ's such that $p_\mu \geq \tau$, then the output register is in state $|1\rangle$ for all such μ with total probability $\geq 1 - \frac{1}{2^t}$.*
2. *If for all $\mu, p_\mu < \tau - \epsilon$ then the output register is observed in state $|0\rangle$ for all μ with probability $\geq 1 - \frac{1}{2^t}$*

Proof. We know that $|\widetilde{p_\mu} - p_\mu| \leq \frac{1}{\epsilon}$. The following cases may arise:

$$p_\mu \geq \tau \implies \widetilde{p_\mu} \geq \tau + \epsilon \tag{4.34}$$

$$p_\mu < \tau - \epsilon \implies \widetilde{p_\mu} < \tau - 2\epsilon \tag{4.35}$$

Proof of part 1

For any μ, probability of getting the state as not marked, when it actually is a good state can be given as:

$$\Pr[\widetilde{p_\mu} < \tau | p_\mu \geq \tau] \tag{4.36}$$

Let $p_\mu*$ be the least possible value of p_μ that should be marked as $|1\rangle$, i.e., $p_\mu* = \tau$. Since, the estimate of p_μ, i.e., $\widetilde{p_\mu}$, has a normal distribution with p_μ as mean,

$$\Pr[\widetilde{p_{\mu*}} < \tau | p_\mu* = \tau] \leq \frac{1}{2} \tag{4.37}$$

For any p_μ due to normal distribution we can say,

$$\Pr[\widetilde{p_{\mu*}} < \tau | p_\mu* \geq \tau] \leq \Pr[\widetilde{p_{\mu*}} < \tau | p_\mu* = \tau] \tag{4.38}$$

$$\leq \frac{1}{2} \tag{4.39}$$

Using this, probability that a good state is marked correctly can be given as

$$\Pr[\widetilde{p_\mu} \geq \tau | p_\mu \geq \tau] = 1 - \Pr[\widetilde{p_\mu} < \tau | p_\mu \geq \tau] \tag{4.40}$$

$$\geq 1 - \frac{1}{2} \tag{4.41}$$

$$\geq \frac{1}{2} \tag{4.42}$$

Now, probability of getting all good states as marked is given as product of probabilities that each good state is marked

$$\prod_{\mu \in \text{good}} \Pr\left[\widetilde{p_\mu} \geq \tau | p_\mu \geq \tau\right] \tag{4.43}$$

$$\geq \prod_{\mu \in \text{good}} \frac{1}{2} \tag{4.44}$$

$$\geq \frac{1}{2^t} \tag{4.45}$$

Proof of part 2: Part 2 of our theorem follows directly from part 2 in Lemma 4.4.2. □

4.4.3 Getting Marked States

For this part we will use the method proposed by van Apeldoorn *et al.* (2023). Once we have the marked states from previous section, the state can be concisely represented as

$$\sum_{\mu \in \text{good}} \frac{1}{\sqrt{2^{n+m}}} |\mu\rangle \left(\beta_{\mu,s} |\widehat{p_\mu}\rangle |1\rangle + \beta_{\mu,\bar{s}} |\widehat{E_\mu}\rangle |0 \text{ or } 1\rangle\right)$$
$$+ \sum_{\mu \notin \text{good}} \frac{1}{\sqrt{2^{n+m}}} |\mu\rangle \left(\beta_{\mu,s} |\widehat{p_\mu}\rangle |0\rangle + \beta_{\mu,\bar{s}} |\widehat{E_\mu}\rangle |0 \text{ or } 1\rangle\right) \tag{4.46}$$

In 4.46 the first component is already correctly marked with flag as 1, due to the p_μ part, thus the error component E_μ will not cause a change in the probability of marking. However for the second component, E_μ can be 0 or 1. The probability of error would be the probability of getting such a E_μ marked as 1. For the worst case we can take all

E_μ in the second part being marked as 1. We can bound this error probability:

$$\begin{aligned}
\Pr[\text{flag } = 1|\mu \notin \text{good}] &= \Pr\left[\text{measuring } \widehat{E_\mu}|\mu \notin \text{ good}\right] \\
&= \sum_{\mu \notin \text{good}} \left|\frac{1}{2^{n+m}}\right|^2 |\beta_{\mu,\bar{s}}|^2 \\
&< \sum_{\mu \notin \text{good}} \frac{1}{2^{n+m}} \left(1 - \frac{8}{\pi^2}\right) \\
&< \frac{2^{n+m} - t}{2^n + m} \left(1 - \frac{8}{\pi^2}\right) \\
&< 1 - \frac{8}{\pi^2}
\end{aligned} \tag{4.47}$$

Therefore with a probability of $< 1 - \frac{8}{\pi^2}$, we can get bad μ as marked, this is a low probability.

The probability that all the good states are marked is $\geq 1 - \frac{1}{2^t}$ 4.4.3. The probability that some bad state is marked alongwith the good states is $< 1 - \frac{8}{\pi^2}$ 4.47. Here, t is number of good states. For retrieving all the marked states, we will use algorithm given by van Apeldoorn *et al.* (2023).

In the remainder of this section we will cover the results given by Apeldoorn for retrieving multiple marked states given a small quantum memory. The problem can be defined as: given a bit string $x \in \{0,1\}^N, x \neq 0$, find all indices $i \in [N]$ such that $x_i = 1$. They define GROVERMULTIPLEFAST algorithm (outlined in Algorithm 12) to solve this problem. The analysis of GROVERMULTIPLEFAST is in Lemma 4.4.4.

Lemma 4.4.4. *Let $x \in \{0,1\}^N$ with $|x| = k \geq 2$, and assume one knows $k_{est} \geq 1$ such that $k/2 \leq k_{est} \leq 3k/2$. Let $0 < \rho < 1$ and $6 \leq \lambda \leq k_{est}$ be such that $t = \lceil k_{est}/\lambda \rceil \geq \log(6k_{est}/\rho)$. Then*

$$O(\sqrt{Nk}(1 + \frac{1}{\sqrt{\lambda}} \log(k/\rho\lambda))) \tag{4.48}$$

quantum queries to x suffice to, with probability $\geq 1 - \rho$, find all k indices i s.t. $x_i = 1$. The algorithm uses an additional $O(\sqrt{Nk}\lambda \log(k/\rho) \log(N))$ non-query gates.

The functionalities of subroutine algorithms are given below:

- APPROXCOUNT: Probabilistic Algorithm; Returns an approximate number of marked elements $\tilde{k}$, such that, $|\tilde{k} - k| \leq \epsilon k$, where $k = |x|$.

Algorithm 12 GROVERMULTIPLEFAST

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$, integer $k_{est} \geq 1$ such that $k/2 \leq k_{est} \leq 3k/2$, failure probability $\rho > 0$, threshold parameter $\lambda \in [6, k_{est}]$

Output: Classical list of indices $J \subseteq [N]$

1: Use GROVERCOUPON to find a uniformly random subset $J_0 \subseteq [N]$ of $\tau.k$ marked elements, where $\tau : 0 < \tau < 1$ is parameter
2: $J = J_0$ ▷ J_0 partitions $[N]$ into intervals
3: **for each** interval (i, j) **do**
4: Estimate $(k_j)_est$ by APPROXCOUNT for interval (i, j)
5: Create oracle U_y for the particular interval (i, j)
6: Use GROVERCERTAINTYMULTIPLE($U_y, 2.(k_j)_{est}$) to find all marked indices in interval (i, j) and append to set J
7: **end for**
8: Return J

- GROVERCOUPON: Probabilistic Algorithm; Given a lower bound on number of marked elements k_lb, gives a subset of marked elements J such that $x_j = 1 \, \forall \, j \in J$
- GROVERCERTAINTYMULTIPLE: Deterministic algorithm: Given a upper bound on number of marked elements k_ub, returns all marked elements J, i.e., $j \in J$ if and only if $x_j = 1$

4.4.4 Analysis

From 4.4.3, the probability with which we get all good μ as marked is $\geq (1 - \frac{1}{2^t})$. From 4.4.4 the probability with which we retrieve all marked μ is $\geq (1 - \rho)$. Therefore, the probability with which we get all good μ is $\geq (1 - \rho)(1 - \frac{1}{2^t})$. The classical (fully classical approach) and quantum (including quantum filtering) time and space complexities are given in table 4.1. For classical naive approach, we will need a 2D table of 2^n rows for μ_1 and 2^m columns for μ_2, and we need to find all elements $\geq \tau$. This can be calculate in space by $O\left(2^{\min(n,m)}/t\right)$. For calculation of each cell we need to compute all possible $x \in \mathbb{F}_2^n$. Thus, time complexity is $O\left(2^{2n+m}\right)$. However, in the quantum

	Classical	**Quantum**
Time	$O\left(2^{2n+m}\right)$	$\tilde{O}\left(\sqrt{2^{n+m} \cdot 2^{\min(n,m)}}/t\epsilon^2\right) = \tilde{O}\left(\sqrt{2^{\min(n,m)} 2^{\frac{3(n+m)}{2}}}/t\right)$
Space	$O\left(2^{\min(n,m)}/t\right)$	$\tilde{O}\left(n + m + \log\frac{1}{\epsilon}\right) = \tilde{O}\left(n + m\right)$

Table 4.1: Time and space complexity comparison of classical and quantum approach to find linear approximations of a function

algorithm for estimate with accuracy ϵ we would need $O(\log \frac{1}{\epsilon}^2)$ queries to "Quantum Linear Approx" oracle (9) and for finding marked state we need $O(\sqrt{2^{n+m}t})$ queries where 2^{n+m} are the total number of elements and t is the number of good linear approximations we want. Also, the number of qubits required will be size of the estimate register and size of μ register and few constant number of registers. In the quantum algorithm here, the minimum possible value of ϵ can be $\frac{1}{2^{n+m}}$.

CHAPTER 5

Conclusion and Future Scope

In this thesis, we implemented an Quantum Circuit for ChaCha cipher having reduced depth at the cost of 4 ancilla qubits. In classical cryptanalysis of ChaCha cipher, finding the suitable $\mathcal{ID} - \mathcal{OD}$ bit pair has required a probabilistic approach. Here, we implemented gave a quantum algorithm to retrieve the good $\mathcal{ID} - \mathcal{OD}$ pairs, which report a high differential bias. We presented an algorithm for finding the Probabilistic Neutral Bits but the resulting circuit would be of huge size.

In the last part of the thesis, we discussed linear cryptanalysis and designed a algorithm to get the linear approximations of any arbitrary function. We suggested a hybrid classical-quantum model and another purely quantum model for the same.

Some of the possible extensions to the current work are as follows:

- **Reducing circuit size of Quantum ChaCha**: We had tried to use states as a superposition of the row values in the ChaCha state matrix, instead of each word as separate states. However, this resulted in entanglement of states which was posing difficult for the ARX operations. A possible way to circumvent this, might be to encode the states as phase. However, for that the ARX operations need to be explored.
- **Multibit differential bias of ChaCha**: Our algorithm gives the good bit pairs out of all a single bit differential bias, however there might be relations between multiple bits differentials that can be explored.
- **Higher order approximation**: Most ciphers today are built in such a way that they are immune to linear cryptanalysis. It would be interesting to develop a quantum approach to identify quadratic or other non-linear approximation in a cipher
- **Quantum version of finding trails Linear Cryptanalysis**: In this work we have discussed algorithm for finding the linear approximations of cipher components. It would be highly useful if we can come up with a mechanism to find full linear trails by combining the different linear approximations obtained.
- **Differential cryptanalysis**: For finding linear approximations we need a deterministic Linear Approximation Table. Similarly for finding high probability differential relations, we can define a probabilistic table with masks for the input and output differentials. If we are able to find the equivalence of these two then the method discussed in this thesis for linear relations can also be used to find differential relations.

REFERENCES

1. **Aumasson, J.-P.**, **S. Fischer**, **S. Khazaei**, **W. Meier**, and **C. Rechberger** (2007). New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. Technical Report 472.

2. **Aumasson, J.-P.** and **M. D. Green**, *Serious Cryptography: A Practical Introduction to Modern Encryption*. No Starch Press, San Francisco, 2017. ISBN 978-1-59327-826-7.

3. **Bathe, B.**, **R. Anand**, and **S. Dutta** (2021). Evaluation of Grover's algorithm toward quantum cryptanalysis on ChaCha. *Quantum Information Processing*, **20**(12), 394. ISSN 1570-0755, 1573-1332.

4. **Beierle, C.**, **G. Leander**, and **Y. Todo** (2020). Improved Differential-Linear Attacks with Applications to ARX Ciphers. *Cryptology ePrint Archive*. URL `https://eprint.iacr.org/2020/775`.

5. **Bera, D.** and **T. Sapv** (2023). Few Quantum Algorithms on Amplitude Distribution.

6. **Berinde, R.** and **P. Indyk** (). Space-optimal Heavy Hitters with Strong Error Bounds. *ACM Transactions on Database Systems*.

7. **Bernstein, D. J.** (2008). ChaCha, a variant of Salsa20.

8. **Brassard, G.**, **P. Hoyer**, **M. Mosca**, and **A. Tapp**, Quantum Amplitude Amplification and Estimation. volume 305. 2002, 53–74.

9. **Choudhuri, A. R.** and **S. Maitra** (2016). Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. *IACR Transactions on Symmetric Cryptology*, 261–287. ISSN 2519-173X.

10. **Coutinho, M.** and **T. C. S. Neto** (2021). Improved Linear Approximations to ARX Ciphers and Attacks Against ChaCha. Technical Report 224.

11. **Crowley, P.** (2005). Truncated differential cryptanalysis of five rounds of Salsa20, 5.

12. **Cuccaro, S. A.**, **T. G. Draper**, **S. A. Kutin**, and **D. P. Moulton** (2004*a*). A new quantum ripple-carry addition circuit. *arXiv:quant-ph/0410184*.

13. **Cuccaro, S. A.**, **T. G. Draper**, **S. A. Kutin**, and **D. P. Moulton** (2004*b*). A new quantum ripple-carry addition circuit. *arXiv:quant-ph/0410184*. URL `http://arxiv.org/abs/quant-ph/0410184`. ArXiv: quant-ph/0410184.

14. **Dey, S.** and **S. Sarkar** (2017). Improved analysis for reduced round Salsa and Chacha. *Discrete Applied Mathematics*, **227**, 58–69. ISSN 0166-218X.

15. **ECRYPT** (Accessed on 2023-05-14). The eSTREAM portfolio page. https://www.ecrypt.eu.org/stream/.

16. **Fischer, S.** (2008). Analysis of Lightweight Stream Ciphers, 149.

17. **Heys, H. M.** (2002). A TUTORIAL ON LINEAR AND DIFFERENTIAL CRYPTANALYSIS. *Cryptologia*, **26**(3), 189–221. ISSN 0161-1194, 1558-1586.

18. **Maitra, S.** (2015). Chosen IV Cryptanalysis on Reduced Round ChaCha and Salsa. Technical Report 698.

19. **Matsui, M.**, The First Experimental Cryptanalysis of the Data Encryption Standard. *In* **Y. G. Desmedt** (ed.), *Advances in Cryptology — CRYPTO '94*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 1994. ISBN 978-3-540-48658-9.

20. **Miyashita, S.**, **R. Ito**, and **A. Miyaji** (2021). PNB-focused Differential Cryptanalysis of ChaCha Stream Cipher, 20.

21. **Shi, Z.**, **B. Zhang**, **D. Feng**, and **W. Wu**, Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. *In* **T. Kwon**, **M.-K. Lee**, and **D. Kwon** (eds.), *Information Security and Cryptology – ICISC 2012*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2013. ISBN 978-3-642-37682-5.

22. **Takahashi, Y.**, **S. Tani**, and **N. Kunihiro** (2009). Quantum Addition Circuits and Unbounded Fan-Out. *arXiv:0910.2530 [quant-ph]*.

23. **van Apeldoorn, J.**, **S. Gribling**, and **H. Nieuwboer** (2023). Basic quantum subroutines: finding multiple marked elements and summing numbers. URL `http://arxiv.org/abs/2302.10244`. ArXiv:2302.10244 [quant-ph].