Cryptanalysis on Symmetric Ciphers in the Quantum Realm

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Rajeet Ghosh

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2024

Supervisor: Prof. Avishek Adhikari

Certificate

This is to certify that this dissertation entitled Cryptanalysis on Symmetric Ciphers in the Quantum Realm towards the partial fulfillment of the BS-MS dual degree program at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Rajeet Ghosh at Indian Institute of Science Education and Research under the supervision of Prof. Avishek Adhikari, Professor (Head of the Department), Presidency University, Kolkata, Department of Mathematics , during the academic year 2019-2024.

Prof. Avishek Adhikari

Committee:

Prof. Avishek Adhikari

Presidency University, Kolkata

Dedicated to the cherished memory of Late Mrityunjoy Adak.

Declaration

I hereby declare that the matter embodied in the report entitled Cryptanalysis on Symmetric Ciphers in the Quantum Realm are the results of the work carried out by me at the Department of Mathematics, Indian Institute of Science Education and Research, Pune, under the supervision of Prof. Avishek Adhikari and the same has not been submitted elsewhere for any other degree.

Rajeet Ghosh

Acknowledgments

I would firstly like to thank the Department of Mathematics, IISER Pune, for allowing me to research and write a thesis. I would like to thank Prof. Avishek Adhikari for giving me an opportunity to work under him, helping me understand the concepts of cryptography, and providing help and support whenever We were unwell this year. I would also like to thank the three research scholars, Himadry Sekhar Roy, Chandan Goswami, and Sandip Kumar Mondal, for guiding me through this project and helping me out anytime I needed them. Lastly, I would like to thank my family and friends for supporting me through this significant journey in whatever way they could.

Abstract

This thesis proposes a biclique attack on AES-128, with Grover's algorithm serving as a superior search method. Bogdanov et al. previously devised and presented a classical version of such an AES-128 biclique attack at ASIACRYPT 2011. They demonstrated that with a biclique of dimension 2^8 and length 3, and 2^{112} base keys, one may mount the attack in the classical domain. This is because there was only one base key group holding the master key; therefore, a partition was created over the whole keyspace to help with the search. Thus, the time complexity is $2^{126.18}$. In contrast, the quantum paradigm allows us to simultaneously superimpose all 2^{128} keys and input them to the Oracle. Using the Grover search method and simultaneous computation of independent differences, We were able to reduce the complexity of the biclique key search to 2^{64}, outperforming Grassl et al.'s attack by a factor of four. Resource estimates for the full attack were also supplied as proof of concept.

Contents

Introduction

Motivation

Cryptanalysis is a crucial field in which quantum algorithms have made significant advancements. Shor's pioneering research challenges the security of discrete logarithm-based problems in public key cryptography, such as the complexities of factoring and computing discrete logarithms in finite cyclic groups like the multiplicative group of a finite field. However, quantum algorithms tend to have a lesser impact on symmetric encryption. While quantum versions of key attacks may pose a threat to block ciphers with quantum access to the encryption mechanism, this attack framework has its limitations. Notably, it does not apply in cases where only a few plaintext-ciphertext pairings are available for determining the encryption key.

Grover's search method [1] has long been recognized for its potential to aid in key discovery. Grover's approach surpasses conventional exhaustive key search methods by a square root, making it the most significant quantum cryptanalytic advancement for block ciphers. However, to implement Grover's method practically, the oracle gate probed by the algorithm must be translated into a circuit. Surprisingly, despite its potential ramifications, there has been a lack of thorough resource estimates for implementing Grover's method, even for well-known ciphers such as the Advanced Encryption Standard (AES), which has been reported to be quantum-safe in its 256-bit version. However, this issue has been addressed through the work of Grassl et al. Once a reversible implementation is established, a quantum implementation becomes feasible, as permutations constitute a subset of all unitary operations.

However, there are certain classical attacks that have not been successfully implemented in a quantum circuit. One such example is the biclique attack [2], one of the few that can

break a full-round AES encryption. They are an improvement on a brute force attack, which Grassl had modified in his work, to fit in a quantum circuit.

Contribution

The thesis' main contribution is the construction of a quantum biclique attack on AES-128, with resource estimates better compared to Grassl's brute force. It also consists of the decomposition of the entire attack, giving a brief explanation of the resource constraints in each of these steps. We use the Clifford+T gate universal quantum gate set. Clifford gates are often less expensive than T-gates, which are commonly implemented by state distillation. When breaking down the circuit to the level of T-gates, we focus on decreasing the overall T-count. Our study found that implementing a biclique-based Grover attack on AES requires fewer logical qubits than Grassl et al., with a total of roughly 2000 logical qubits against 3000. Implementing the Grover algorithm on a physical quantum computer is problematic owing to the enormous circuit depth required to unroll the entire loop, even if the gates are not error-corrected, as indicated by Grassl. The circuit cost of each Grover iteration is mostly due to key expansion, which involves generating round keys. Grover's algorithm's serial nature also contributes to its total depth.

Organisation

The thesis is organized as follows:

- The first chapter is a brief overview of cryptography and symmetric ciphers.
- The second chapter consists of an introduction to quantum computing with its main focus on Grover's search.
- The third chapter describes cryptanalysis in general, with some common symmetric key-based cryptanalytic attacks.
- The last chapter is the conclusion of this thesis, consisting of a biclique attack on AES-128 by Bogdanov et al. and designing a quantum algorithm based on their attack

Chapter 1

Overview of Cryptography and Some Ciphers

Cryptography is the study of securing data, information, and communications through the use of codes so that only those persons for whom the data is intended can understand and process it, thus preventing unauthorized access to information. To understand this, consider the following example:

Let us assume two people, Alice and Bob, wanted to send each other a message (called **plaintext** henceforth) but wanted to keep it a secret from a third party receiving it, Eve. They can now encrypt it in some way, like shifting a letter and adding something to the message, or as we will see, some manipulation of this plaintext using what we call the **key**, some of which remain completely hidden other than Bob and Alice themselves. Without this key, this changed message, known as the **ciphertext**, cannot be easily decrypted by Eve, and thus the message remains hidden publicly. The algorithm they use to encrypt the plaintext to a ciphertext using a certain key, which, given the same key, can also the ciphertext to the same plaintext, is called a **cipher**.

A good cipher requires the following features:

- **Confidentiality**: Data sent can only be accessed by the receiver and no one else.
- **Integrity**: Data cannot be changed in storage or transferred between sender and

intended receiver without any additional information being detected.

- **Non-repudiation**: The creator and the sender of data cannot deny their intention to send information at a later iteration.
- **Authentication**: The identities of the sender and receiver, as well as the destination/origin of the information, are confirmed.

Depending on these keys used, there are two types of cryptography:

- **Public Key Cipher**:
  Also known as **Asymmetric Ciphers**, it uses a pair of keys. One key or a set of keys is made public and is used to encrypt a given plaintext to ciphertext. The other key is kept private and is used to decrypt it.
- **Private Key Ciphers**:
  Also known as **Symmetric Ciphers**, it only uses one key, which is kept completely private. This key itself is used for both encryption and decryption. It is symmetric as this key itself is given to the receiver in secret and used to decrypt a given ciphertext. It is faster than Public Key Cryptography, and all the ciphers implemented in this project belong to this class.

1.1 Types of Symmetric Ciphers

There are two classes for ciphers in symmetric encryption:

- **Stream Ciphers**:
  A stream cipher operates as a symmetric key cipher, intertwining plaintext bits with a pseudorandom cipher bit stream, known as the keystream. In this cipher, each plaintext digit undergoes encryption individually, aligning with the corresponding digit of the keystream, culminating in a corresponding digit within the ciphertext stream.

  The pseudorandom keystream is conventionally crafted sequentially, derived from an initial random key value through shift registers. This value carries the mantle of being

the cryptographic key, pivotal for decrypting the ensuing ciphertext stream. Some examples are the GRAIN family, DRACO, LESCA, etc.

- **Block Ciphers**:
  A block cipher, a deterministic algorithm, acts on fixed-length bit groups known as blocks. These ciphers are the fundamental components of numerous cryptographic protocols and find widespread application in data storage and secure data exchange scenarios. The operation of a block cipher entails a consistent transformation of these blocks. Even though a block cipher is secure, it is designed to encrypt one data block at a time, employing a fixed key.

  Various modes of operation have been developed to enable their repeated use while maintaining security and ensuring confidentiality and authenticity in data protection. Block ciphers also find utility as foundational elements in other cryptographic protocols, such as universal hash functions and pseudorandom number generators.

1.2 Examples of Some Symmetric Ciphers

1.2.1 GRAIN-128

GRAIN is a family of stream ciphers [3], and one of the ciphers- GRAIN-128 is implemented here. It takes a 128-bit key and 96-bit initialization vector as input and produces a keystream output of 128 bits. It uses 2 FSRs (Feedback Shift Register), one with Linear Feedback

$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128},$$

and the other Non-Linear

$$g(x) = 1+x^{32}+x^{37}+x^{72}+x^{102}+x^{128}+x^{44}x^{60}+x^{61}x^{125}+x^{63}x^{67}+x^{69}x^{101}++x^{80}x^{88}+x^{110}x^{111}+x^{115}x^{117}$$

The encryption works as follows:

- First, a stream of 256 bits is created by concatenating the IV to the key and then 32

1s at the end. In other words, if key

$$K = (k_0, k_1, k_2, \cdots, k_{127})$$

$$\text{and IV} = (v_0, v_1, \cdots, v_{95})$$

$$\text{thenInitial State } S_0 = K||IV||(\underbrace{1, 1, \cdots, 1}_{32 \text{ times}}), = (k_0, k_1, \cdots, k_{127}, v_0, v_1, \cdots, v_{95}, \underbrace{1, 1, \cdots, 1}_{32 \text{ times}})$$

- Then, the output is generated as follows:
  - We define the function $h(\text{state}) := b_{i+12}s_{i+18} \oplus s_{i+13}s_{i+20} \oplus b_{i+95}s_{i+42} \oplus s_{i+60}s_{i+79} \oplus b_{i+12}b_{i+95}s_{i+95}$
  - The output for each iteration is produced as $z_i = \bigoplus_{j \in \mathcal{A}} b_{i+j} \quad \oplus h(\text{state}) \oplus s_{i+93}$ where $\mathcal{A} = \{2, 15, 36, 45, 64, 73, 89\}$
- LFSR and NFSR are applied to this state to get the new state; for simplification, the keystate can be divided into two equal parts $S = IV|| \underbrace{1, \cdots, 1}_{32 \ times}$ and $B = K$ S is updated by LFSR as follows: $s_{i+128} = s_i \oplus s_{i+7} \oplus s_{i+38} \oplus s_{i+70} \oplus s_{i+81} \oplus s_{i+96}$ B is updated by the NSFR as follows: $b_{i+128} = s_i \oplus b_i \oplus b_{i+26} \oplus b_{i+56} \oplus b_{i+91} \oplus b_{i+96} \oplus +b_{i+3} \cdot bi + 67 \oplus b_{i+11} \cdot b_{i+13} \oplus b_{i+17} \cdot b_{i+18} \oplus b_{i+27} \cdot b_{i+59} \oplus b_{i+40} \cdot b_{i+48} \oplus b_{i+61} \cdot b_{i+65} \oplus b_{i+68} \cdot b_{i+84}$
- For the first 256 iterations, no output is produced. Instead z_i is XORed with both s_{i+128} and b_{i+128}.
- For the next 128 iterations, the respective output is produced.

1.2.2 DES

The DES (Data Encryption Standard) [4] cipher is now considered obsolete in data encryption but has paved the way for many future block ciphers that have come in the present, including AES. It takes a 64-bit plaintext with a 64-bit key. However, only 56 bits of the 64 bits are used for encryption; thus, brute force attacks are feasible. The algorithm works as follows:

- First, the plaintext of 64 bits is converted to some base text (call it IP) using an initial permutation function.

- The IP is then divided into two parts independently referred to as Left Permutation Block (LPT) and Right Permutation Block (RPT).
- Both LPT and RPT undergo 16 rounds of encryption.
- Finally, both LPT and RPT are concatenated, and the whole block is passed through a Final Permutation (FP) function, giving us a ciphertext of 64 bits.

The encryption follows the following steps:

Key Transformation

First, every 8th bit of the 64-bit key is removed. Then, the 56-bit key is divided into two parts, each 28 bits. Each of these parts undergoes a left circular permutation by 1 in rounds 1, 2, 9, and 16, and 2 in other rounds.

After this, they are again concatenated, and this new 56-bit key is reduced to 48-bit text using the Compression Permutation as depicted in table **??**. For example, bit 14 moves to the first position in the new key, bit 17 moves to the second position, and so on. In this Table, however, 8 bits (like bit 18)are left, reducing the key size to 48.

Expansion Permutation

The 32-bit RPT is expanded to 48 bits as follows: The RPT is divided into eight blocks of size 4. The size of each of these blocks is then extended to 6 bits. Then, this 48-bit expanded RPT is XORed with the 48-bit compressed key.

S-box and P-box Permutation

The 48-bit RPT is converted to 32-bit as follows: Again, the RPT is divided into eight blocks of size 6 bits. Each of these blocks is passed into an S-box of given ordering to yield a 4-bit text, all of which concatenated gives us 32 bits back. The generated text is again

permuted in a P-Box. All the S-boxes, the P-box, and the IP function can be referred to in this:

XOR and Swap

Now, LPT is XORed with the generated RPT to give the new RPT. Also, the new LPT is the old RPT. The new LPT and RPT concatenated will give the ciphertext for that stage.

1.2.3 AES

Also known as the **Rijndael** cipher [5], this is a better cipher than DES and is still used today. It takes a 128-bit plaintext and 128/192/256-bit key and encrypts it to a 128-bit ciphertext. In this project, we focus on the 128-bit key. The key and the plaintext are viewed as a 4x4 block, each carrying a byte (8 bits) of information, which will be viewed as two hexadecimal (base-16) digits. Firstly, the key is XORed with the plaintext. Encryption uses the following functions: 1) **SubBytes**, 2) **ShiftRows**, 3) **MixColumns**, 4) **Key-Generation** and 5) **AddRoundKey**, which are described later. For a 128-bit key, the encryption is run on the modified plaintext ten times and in the abovementioned order. In the tenth round, however, MixColumns is skipped.

SubBytes

Data in each cell are substituted with respect to the table in figure 1.1. For example, "19" is replaced with "d4", "a0" to "e0", and so on.

ShiftRows

Except for the first one, rows of the 4x4 block undergo left circular permutation. The second row is permutated once, the third one twice, and the fourth thrice.

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

Figure 1.1: The SubBytes table

MixColumns

For this, the cell is viewed as an element of the Galois Field GF(2^8) with the primitive $p(x) = x^8 + x^4 + x^3 + x + 1$

For example, "ab" and "f1" can be seen as: $(ab)_{16} = (1010\ 1011)_2 \equiv x^7 + x^5 + x^3 + x + 1$ $(f3)_{16} = (1111\ 0011)_2 \equiv x^7 + x^6 + x^5 + x^4 + x + 1$

Addition and Multiplication are defined as follows: $If\ x, y \in GF(2^8), \text{then } x \oplus y = (x + y),\ \text{and } x \odot y = (x \cdot y) \bmod p(x)$

For example, $ab \oplus f3 \equiv ((x^7+x^5+x^3+x+1)+(x^7+x^6+x^5+x^4+x+1)) = x^6+x^4+x^3 \equiv 58$ And, $ab \odot f3 \equiv (x^7 + x^5 + x^3 + x + 1) \cdot (x^7 + x^6 + x^5 + x^4 + x + 1)(x \cdot y) \bmod p(x) = x^7 + x^5 + 1 \equiv a1$

The columns of each block are multiplied by the matrix $\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$

For example, taking the first column as $\begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix}$, we see $\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ e5 \end{bmatrix}$

Key Generation

An Rcon table is made, with the starting vector $\begin{pmatrix} 01 \\ 00 \\ 00 \\ 00 \end{pmatrix}$ and a new element of each step is generated by multiplying (wrt to $GF(2^8)$ and with the primitive $= x^8 + x^4 + x^3 + x + 1$) 02 with the previous element. Since we are dealing with a 128-bit key, we can have a predefined Rcon table, which is shown in the table 1.1

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Table 1.1: The Rcon Table for AES-128 for 10 rounds

The starting round key block is the original key provided During the n-th iteration of the encryption process, the first column of the new round key is generated as follows:

- First, the last column of the previous key block is taken, and it undergoes a circular permutation, with the first element becoming the previous, the second element the first, and so on.
- Then, this column is passed through the SubBytes function.
- This column is XORed with the first column of the old key and the n-th column of the Rcon table. The rest of the columns are generated by XORing the first column of the new key with the respective column of the old key.

AddRoundKey

The modified plaintext is now XORed with the Round Key generated at the n-th iteration.

Before the first round, the plaintext is XORed with the original key, and each round except the last round consists of SubBytes, ShiftRows, MixColumns, and AddRoundKey, in this order. In the last round, the MixColumns step is skipped. There are ten rounds for AES-128, 12 rounds for 192, and 14 rounds for 256, respectively.

1.2.4 Kalyna

The cipher [6] uses a very similar algorithm to AES in terms of functions but differs in the fact that the last step will still use MixColumns and that the Key Generation uses a simple circular shift function ($K_i = K_{i-1} <<< (\frac{b}{4} + 24)$), where b is the size of plaintext in bits). It has five variants, depending on the plaintext/key size: 128/128, 128/256, 256/256, 256/512, and 512/512. The blocks for 128, 256, and 512-bit plaintext are of sizes 8×2, 8×4, and 8×8, respectively, with each cell storing 1 byte (8 bits) of information.

Chapter 2

An Introduction to Quantum Computing

Our main focus of this paper is to implement a quantum attack, for which we require an understanding of quantum computing in general. We'll mainly borrow concepts from [7] and [8] to use in our attack.

2.1 Postulates of Quantum Mechanics

Since quantum computing is an offshoot of quantum mechanics, therefore before describing anything related to quantum computing, we must follow some axioms to understand this realm, known as the **5 postulates of quantum mechanics**:

1. The state of a particle is completely determined by a vector function $|\psi(\mathbf{r}, t)\rangle$ where $\mathbf{r}$ denotes the position of the particle and t denotes time. In particular, ψ must abide by the following:
$$\int_{-\infty}^{\infty} \langle \psi \,|\, \psi \rangle \, d\tau \;\; = 1$$
where $\langle \psi | := |\psi\rangle^{\dagger} = \overline{{}^{t}A}$ is the dual of ψ, and $\tau(\mathbf{r}, t)$ is a measure of space and time.

2. There is a linear Hermitian operator that correlates to a particular observable in clas-

sical mechanics.

3. Only the values of eigenvalue a that meet the following eigenvalue equation will ever be detected in any measurement of the observable associated with operator A:

$$A|\psi\rangle = a \cdot |\psi\rangle$$

4. For any system represented by ψ, the expectation value of an operator A, defined as $\langle A \rangle$ is calculated as follows:

$$\langle A \rangle = \int_{-\infty}^{\infty} \langle \psi | A | \psi \rangle \, d\tau$$

5. For a state ψ, the Schrödinger equation describes how ψ of a system changes over time:

$$\hat{H}|\psi\rangle = \mathrm{i}\frac{\partial}{\partial t}|\psi\rangle$$

where $\hat{H}$ is the Hamiltonian, also called the total energy operator, $\hat{H} = -\frac{\partial^2}{\partial x^2} + V(x)$ such that V is an operator over position only.

2.2 Introduction to Qubits

As one can follow, the function ψ can be assumed to be a member of a Hilbert Space $\mathcal{H}$ since its dual is also the same space $\mathcal{H}$, the first postulate collapses to simply $\langle \psi \,|\, \psi \rangle$, and we now simply need an orthonormal basis to describe any element $|\psi\rangle$ Instead of considering the classical bit system generated from $\{0, 1\}$, we now consider a qubit one, where $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, two basis of the complex projective space $\mathbb{CP}^1 = \frac{\mathbb{C}^2}{\sim}$, where $\sim$ describes an equivalence class, $v \sim w \iff v = k \cdot w$ for some $k \in \mathbb{C}$ and for all $v, w \in \mathbb{C}^2$. $\mathbb{CP}^1$ is often represented by a Bloch Sphere, with vectors originating from the origin $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and reach to the surface, with entries $\begin{bmatrix} \cos(\frac{\theta}{2}) \\ e^{i\phi} sin(\frac{\theta}{2}) \end{bmatrix}$ for all $\theta \in [0, 2\pi)$. Note that $\mathbb{CP}^1$ is a Hilbert space since this space is both normed-linear and complete.

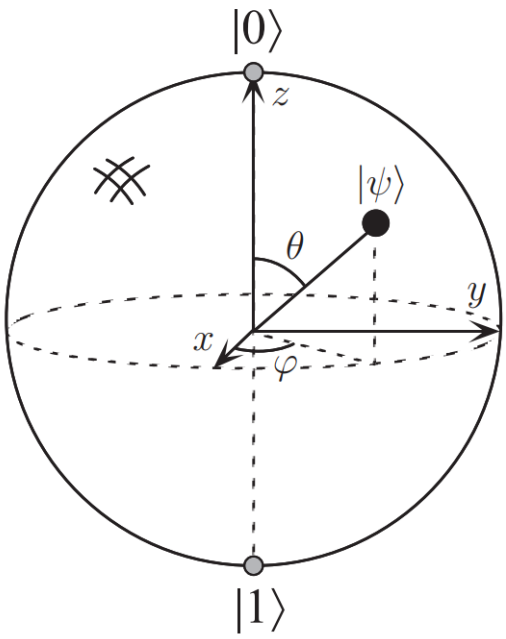

Figure 2.1: Bloch Sphere representation of a qubit

The reason $\mathbb{CP}^1$ is chosen to describe the function $|\psi\rangle$ is simple; it simplifies our choice of the coefficients. We can create any state representing a vector in the Bloch Sphere, such as $v = \alpha|0\rangle + \beta|1\rangle$ with $|\alpha|^2 + |\beta|^2 \leq 1$. Following the first postulate, we can only choose v such that $|\alpha|^2 + |\beta|^2 = 1$, which are called **pure states**; otherwise, they are called mixed states. For this thesis, we will skip the idea of mixed states; otherwise, they are a crucial concept, especially in understanding the concept of quantum entanglement. This also restricts operators A to be unitary as well, which helps a lot in the choice of A

2.3 Rotation and Phase Change of qubits

Like rotations in $\mathbb{R}^3$, $\mathbb{CP}^1$ also has rotation matrices around its axes. We define Pauli matrices as follows:

$$\sigma_x := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\sigma_y := \begin{bmatrix} 0 & \mathrm{i} \\ -\mathrm{i} & 0 \end{bmatrix}$$

$$\sigma_z := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Thus to rotate these vectors by an angle θ, around the x,y and z axis, we just multiply the vectors by $e^{-\frac{i\sigma_x\theta}{2}}, e^{-\frac{i\sigma_y\theta}{2}}$ and $e^{-\frac{i\sigma_z\theta}{2}}$ respectively.

We also need to note that since we're working in a sphere, the phase shift is also necessary. For changing its phase by ϕ, we can simply multiply it with $\begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$

2.4 Increasing Dimensions of Qubits

To mimic our classical way of extending bits, which was a simple Cartesian product, we need to define a system to increase the dimensions of our qubits for the simple reason that the classical counterpart can represent any information with a large number of bits. We do so by going for a tensor product, as the tensor of two Hilbert spaces is also a Hilbert Space. The tensor product of two elements is defined as follows: If $|v\rangle = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$ and $|w\rangle = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$, are two qubits of dimensions n and m, respectively, then

$$v \otimes w = \begin{bmatrix} v_1 \cdot w \\ v_2 \cdot w \\ \vdots \\ v_m \cdot w \end{bmatrix} = \begin{bmatrix} v_1 w_1 \\ v_1 w_2 \\ \vdots \\ v_1 w_n \\ v_2 w_1 \\ v_2 w_2 \\ \vdots \\ v_m w_n \end{bmatrix}$$

We denote $|v\rangle \otimes |w\rangle$ as $|vw\rangle$. Thus, for example,

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Similarly, for all $1 \leq j \leq n$ and $a_i \in \{0, 1\}$ we define

$$|a_1 a_2 \dots a_n\rangle := \bigotimes_{i=1}^{n} |a_i\rangle$$

Also, the product of two bras and two kets is a tensor product, whereas that of a bra followed by a ket is an inner product. So for example,

$$|v\rangle|w\rangle = |v\rangle \otimes |w\rangle$$

and

$$\langle v|\langle w| = \langle v| \otimes \langle w|$$

, but

$$\langle v \,|\, w\rangle = \langle v, w\rangle$$

. Further, note that $|a^n\rangle := \bigotimes_{i=1}^{n} |a\rangle$. So, for example, later on in the second last chapter, one would encounter $|0^8\rangle = |00000000\rangle = \bigotimes_{i=1}^{8} |0\rangle$

2.5 Quantum Gates

Now that we have defined a quantum analogue to a classical bit, we require some quantum alternative to the classical logic gates. Some properties have to be maintained since we are working in a different space, the Hilbert Space of dimension n instead of $\mathbb{F}_2^n$. One important postulate in quantum mechanics implies that the operators acting on states to retrieve information must be unitary, because of which classical operations like AND, OR, XOR, etc. cannot be used in a quantum circuit. Some gates of 1-dimensional qubits are:

- **Identity** (Denoted by I):

  $$I(|0\rangle) = |0\rangle \text{ and } I(|1\rangle) = |1\rangle$$

  Its matrix representation would be $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

- **NOT** :

$$X(|0\rangle) = |1\rangle \text{ and } X(|1\rangle) = |0\rangle$$

Its matrix representation is the **Pauli Matrix** σ_x

- **Hadamard Gate** (Denoted by H):

$$H(|0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Note that Hadamard gate has no classical alternatives. Its matrix representation is

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Also note that $H|0\rangle$ and $H|1\rangle$ form an orthonormal basis and hence one can also use $|+\rangle := H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle := H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ in calculations instead. This is also one of a few gates which doesn't have a classical counterpart,

- The **Pauli Matrices** Y and Z as above. Similar to

- **S gate**. Often called the **phase gate**, its matrix representation is $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$.It is also called the $\sqrt{Z}$ gate, as $S^2 = Z$. It takes $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $i|1\rangle$

- **T gate**. This gate has a matrix representation of $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$. It has two names: it's called $Z^{\frac{1}{4}}$ gate as $T^2 = S$ and thus $T^4 = S^2 = Z$, and its also called $e^{\frac{i\pi}{8}}$ gate as it can also be factorised as $T = e^{\frac{i\pi}{8}} \begin{bmatrix} e^{-\frac{i\pi}{8}} & 0 \\ 0 & e^{\frac{i\pi}{8}} \end{bmatrix}$. It takes $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $e^{\frac{i\pi}{4}}|1\rangle$

Also, note that tensor products can extend these gates to a higher dimension. For example, Hadamard gates can be extended as an n-dimensional gate as follows:

$$H^{\otimes n} = \bigotimes_{i=1}^{n} H$$

and hence for $x \in \{0,1\}^n$

$$H^{\otimes n}|x\rangle = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$$

Unless explicitly mentioned, we will write $H^{\otimes n}$ as simply H_n. Another freedom is that the tensor of two gates is possible, i.e., any m-dimensional gate A can be tensored by an n-dimensional gate B to result in a gate of dimension $m+n$. The resultant gate $A \otimes B$ is a unitary operator such that for any $|\psi_1\rangle$ and $|\psi_2\rangle$ in the domains of A and B respectively,

$$A \otimes B(|\psi_1\rangle \otimes |\psi_2\rangle) = A|\psi_1\rangle \otimes B|\psi_2\rangle$$

One qubit can affect others by using gates. An example is the **CNOT** (Controlled NOT) gate, a 2-dimensional qubit gate, which acts as follows: for $a, b \in \{0,1\}$

$$C(|a\rangle|b\rangle) = |a\rangle|b \oplus a\rangle$$

where $\oplus$ is the XOR addition in the classical gates. Upon further observation, its matrix form is simply $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = I \otimes \sigma_x$. Similarly there is a **CCNOT** Gate, a 3-dimensional gate, which works as follows:

$$C|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|c \oplus (a \cdot b)\rangle$$

which again is $\left(\begin{array}{c|c} I_6 & 0 \\ \hline 0 & \sigma_x \end{array}\right)$ Hence we can extend this to n-dimension, called the n-fold Toffoli gate, as

$$C(|a_1, a_2, \ldots, a_{n+1}\rangle) = |a_1, a_2, \ldots, a_n, a_{n+1} \oplus (a_1 \cdot a_2 \cdot \ldots \cdot a_n)\rangle$$

which again can be represented as $\left(\begin{array}{c|c} I_{2^n-2} & 0 \\ \hline 0 & \sigma_x \end{array}\right)$ Another commonly used operator is called the SWAP Gate as follows:

$$|a\rangle|b\rangle \xmapsto{SWAP} |b\rangle|a\rangle$$

. But as we will see, it is just a composition of 3 CNOT gates in this order:

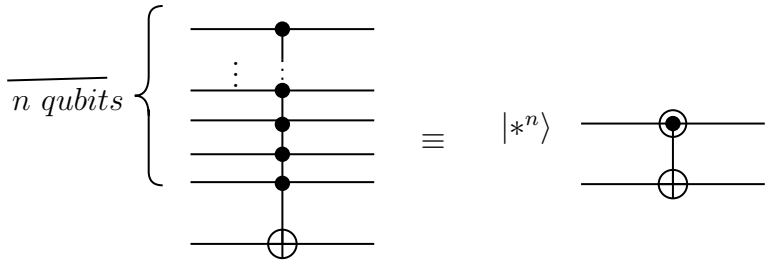

Figure 2.2: n-fold Toffoli Gate

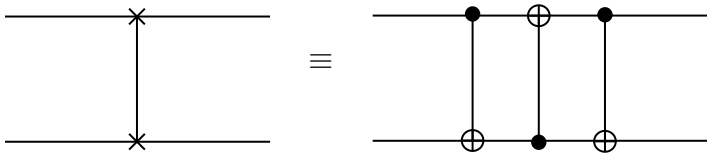

Figure 2.3: SWAP gate as a composition of 3 CNOT Gates

This is because

$$|a\rangle|b\rangle \xmapsto{CNOT} |a\rangle|b \oplus a\rangle \xmapsto{NOTC} |a \oplus b \oplus a\rangle|b \oplus a\rangle = |b\rangle|b \oplus a\rangle \xmapsto{CNOT} |b\rangle|b \oplus a \oplus b\rangle = |b\rangle|a\rangle$$

Here, $NOTC$ simply implies that the control qubit is the second instead of the first.

2.6 Measuring States in Multi-qubit Systems

Now we work with $|\psi\rangle = \sum_{i_1,i_2,\ldots,i_n \in \{0,1\}} \alpha_{i_1,i_2,\ldots,i_n} |i_1, i_2, \ldots, i_n\rangle$ with $\sum_{i_1,i_2,\ldots,i_n \in \{0,1\}} |\alpha_{i_1,i_2,\ldots,i_n}|^2 = 1$ To find the probability of the multi-qubit $|i_1, i_2, \ldots, i_n\rangle$ for some fixed $i_1, i_2, \ldots, i_n \in \{0, 1\}$ we use an operator operator called the measurement operator, defined as $M_{i_1,i_2,\ldots,i_n} = m^\dagger_{i_1,i_2,\ldots,i_n} \cdot m_{i_1,i_2,\ldots,i_n}$, where $m_{i_1,i_2,\ldots,i_n} := |i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n|$. The probability of state

$|i_1, i_2, \ldots, i_n\rangle$ in ψ is defined as:

$$\begin{aligned}
&P_\psi(|i_1, i_2, \ldots, i_n\rangle) \\
&\quad = \langle\psi|(|i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n|)^\dagger(|i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n|)|\psi\rangle \\
&\quad = [(|i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n|)|\psi\rangle]^\dagger[(|i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n|)|\psi\rangle] \\
&\quad = [|i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n \,|\, \psi\rangle]^\dagger[|i_1, i_2, \ldots, i_n\rangle\langle i_1, i_2, \ldots, i_n|\psi \,|\,]\rangle \\
&\quad = [|i_1, i_2, \ldots, i_n\rangle\alpha_{i_1,i_2,\ldots,i_n}]^\dagger[|i_1, i_2, \ldots, i_n\rangle\alpha_{i_1,i_2,\ldots,i_n}] \\
&\quad = \alpha^\dagger_{i_1,i_2,\ldots,i_n}\langle i_1, i_2, \ldots, i_n| \cdot \alpha_{i_1,i_2,\ldots,i_n}|i_1, i_2, \ldots, i_n\rangle \\
&\quad = |\alpha_{i_1,i_2,\ldots,i_n}|^2\langle i_1, i_2, \ldots, i_n \,|\, i_1, i_2, \ldots, i_n\rangle = |\alpha_{i_1,i_2,\ldots,i_n}|^2
\end{aligned}$$

So, the probability of each state is still the square of the absolute value of the coefficients of the state in ψ. As one can observe, measurement is not a unitary operator. After measurement, a quantum state now collapses to its classical bitstring, and the entire state $|\psi\rangle$ will cease to exist.

2.7 No-Cloning Theorem

The idea that we can easily copy a classical bitstring onto another is immediately refuted by this general fundamental theorem of quantum mechanics, particularly quantum computing. It is one of the major reasons for the existence of quantum key distribution systems, which are substitutes for public key cryptosystems.

Theorem 2.7.1. *Take an arbitrary qubit $|\psi\rangle$ belonging in a Hilbert space $\mathcal{H}$. Then there does not exist any unitary operator U other than identity, such that $\forall\ |j\rangle \in \mathcal{H}$ and some $0 \leq \alpha < 1$ depending on ψ and j,*

$$U(|\psi\rangle|j\rangle) = e^{2\pi i\alpha(\psi,j)}|\psi\rangle|\psi\rangle$$

Proof. We prove it using contradiction. Suppose there exists a unitary operator $\tilde{U}$ which does as follows: for some $|\tilde{j}\rangle \in \mathcal{H}$, so that for any $|\psi\rangle \in \mathcal{H}$,

$$U(|\psi\rangle|\tilde{j}\rangle) = |\psi\rangle|\psi\rangle$$

Now for any $|\phi\rangle, |\psi\rangle \in \mathcal{H}$,

$$\begin{aligned}
\langle\phi\,|\,\psi\rangle &= \langle\phi\,|\,\psi\rangle\langle\tilde{j}\,|\,\tilde{j}\rangle \\
&= \langle\tilde{j}|\langle\phi||\psi\rangle|\tilde{j}\rangle \\
&= \langle\tilde{j}|\langle\phi|\tilde{U}^\dagger\tilde{U}|\psi\rangle|\tilde{j}\rangle \\
&= (\tilde{U}|\phi\rangle|\tilde{j}\rangle)^\dagger(\tilde{U}|\psi\rangle|\tilde{j}\rangle) \\
&= (e^{2\pi i\alpha(\phi,\tilde{j})}|\phi\rangle|\phi\rangle)^\dagger(e^{2\pi i\alpha(\psi,\tilde{j})}|\psi\rangle|\psi\rangle) \\
&= e^{2\pi i(\alpha(\psi,\tilde{j})-\alpha(\phi,\tilde{j}))}\langle\phi|\langle\phi||\psi\rangle|\psi\rangle \\
&= e^{2\pi i(\alpha(\psi,\tilde{j})-\alpha(\phi,\tilde{j}))}\langle\phi\,|\,\psi\rangle^2 \\
\implies |\langle\phi\,|\,\psi\rangle| &= |\langle\phi\,|\,\psi\rangle|^2 \\
\implies |\langle\phi\,|\,\psi\rangle| &= 0 \text{ or } 1
\end{aligned}$$

This implies that either $|\phi\rangle = e^{2\pi i\alpha}|\psi\rangle$ or $|\phi\rangle$ and $|\psi\rangle$ were orthogonal. But we had stated that the choice of $|\phi\rangle$ and $|\psi\rangle$ was arbitrary, thus implying our assumption was wrong. □

2.8 Quantum Gate Approximation

Universal gate sets in classical logic gates are those gates using which you can generate other gates, for example, using AND, OR, and NOT operations, which can generate logic gates like NAND and NOR The question now arises if there is a quantum analog to a universal gate set. This question is solved using the **Solovay Kitaev Theorem**.

Theorem 2.8.1. *Let $G \subset SU(2)$, such that it holds the following three conditions.*

- *G is finite.*
- *If $g \in G, g^{-1} \in G$*
- *$\langle G\rangle$ is dense in SU(2).*

Then for any $\epsilon > 0$ and $U \in SU(2)$, there exists $G' \subseteq G$, such that $D(U, \prod_{g\in G'} g) < \epsilon$[1] with $|G'| = O(\log^c(1/\epsilon))$ and c a constant independent of ϵ, G and U

[1]Here $D(U,V) = tr(X)$ where X is a unique matrix such that $X^2 = (U-V)^\dagger(U-V)$

The proof will be skipped as it is a massive one, a thesis of its own. However, the theorem implies there exists a universal quantum gate set whose composition approximates any gate in $SU(2)$, and hence for $U(2)$. One example could be to use rotations, phase shifts and the CNOT gate. However, we will work with a specific universal gate set, one which helps us compute complexity in our quantum computers.

2.9 Clifford Gates

$U(2)$ defines all 1-dimensional quantum gates. One can also thus define a subgroup $\mathcal{P}_1 := \{c \cdot p \mid c \in \{\pm 1, \pm\iota\}, p \in \{I, X, Y, Z\}\}$ containing all Pauli gates.

Definition 2.9.1. ***Clifford Group of dimension 1*** *C_1 is the normaliser of $\mathcal{P}$ over $U(2)$. In other words,*

$$C_1 := Norm_{U(2)}(\mathcal{P}_1) = \{g \in U(2) \mid gpg^\dagger \in \mathcal{P}_1 \text{ for all } p \in \mathcal{P}_1\}$$

Any member of the Clifford Group is called a Clifford Gate. C_1 is generated using H and S, thus $C_1 = \langle H, S \rangle$. X, Y, and Z are obvious members of $\mathcal{P}_1$, but $Z = S^2$, X=HS^2H and Y=$(SH)S^2(SH)^\dagger = SHS^2HS^\dagger$, so they are easily generated by H and S gates In general $\mathcal{P}_n := \{c \bigotimes_{j=1}^{n} p_i | c \in \{\pm 1, \pm\iota\}, p_i \in \{I, X, Y, Z\}\}$ However, as one can notice in the 2-dimensional case, we have skipped the CNOT gate, which also fulfills this criterion, because of which we add it in the Clifford group C_2, and CNOT is also added as a generator of C_n in general. The question now arises: is Clifford Gates sufficient to cover all $U(2^n)$? The answer is no! But using Clifford gates alongside the T gate is enough to generate all possible gates in $SU(2^n)$. Take, for example, the CCNOT gate (2-fold Toffoli gate), whose decomposition in terms of Clifford gates and the T gate looks as follows:

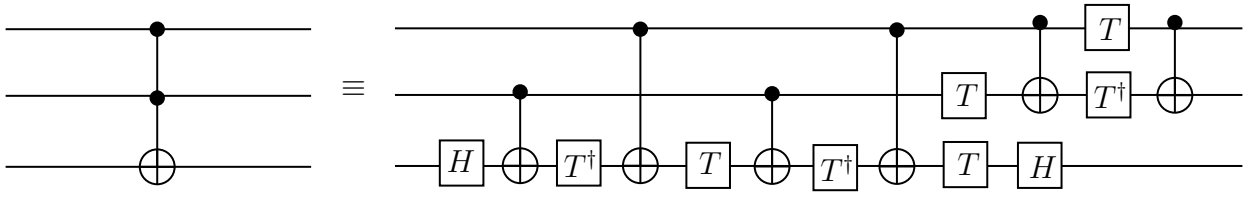

Figure 2.4: CCNOT gate as a composition of H, S, and T gates

Now, we come to the reasoning behind choosing the Clifford Gates and the T gate as a universal quantum gate set. The following theorem, called the **Gottesman–Knill theorem**, brings a simplistic answer to the complexity problem:

Theorem 2.9.1. *A classical computer can simulate, in polynomial time, a quantum circuit consisting of only the following parts:*

- *Preparing qubits in a computational basis states.*
- *Measurement in the same computational basis.*
- *Clifford Gates*

This theorem's proof will also be skipped. However, the statement of the theorem implies that a classical computer can easily simulate a quantum circuit consisting only of Clifford gates, and as such, in this specific case, a quantum computer does not perform significantly faster than its classical counterpart. Therefore, the majority of quantum circuits are made so that the use of the T gate is minimized, and in essence, since the classical computer cannot simulate T gates efficiently, the quantum computer has an advantage. The number of times T gates are used in a circuit is called the T-complexity (not to be confused with time complexity), and the maximum number of times it is used on a single qubit in the entire circuit is called T-depth.

2.10 Boolean functions in the Quantum Realm

A boolean function f is a mapping from a set of boolean variables $\{x_1, x_2, \ldots, x_n\}$ to the boolean domain $\{\text{True}, \text{False}\}$, denoted as:

$$f : \{0,1\}^n \to \{0,1\}$$

where n is the number of boolean variables. Each boolean variable x_i can take either the value 0 or 1, and the function f returns a boolean value True(= 1) or False(= 0) based on the combination of truth values of the input variables.

A common way of viewing a discrete function (including boolean) is to look at the **truth table**, which is

$$[f] := \{f(0), f(1), \ldots, f(2^n - 1)\}$$

We first define the **Hamming weight** in terms of binary codes.

Definition 2.10.1. *The Hamming weight of a binary code S is the number of nonzero coordinates in S, that is, coordinates with value 1. In mathematical sense,*

$$wt(S) = \#\{j|S_j = 1\}$$

In the case of boolean functions, the Hamming weight of a boolean function f is defined instead on the truth table of the function, that is:

$$wt(f) := wt([f])$$

Definition 2.10.2. *The Hamming distance between 2 binary codes S_1 and S_2 is the number of coordinates where S_1 and S_2 differ.*

$$d(S_1, S_2) := \{i|S_{1_i} \neq S_{2_i}\}$$

Again, similar to Hamming weight, the Hamming distance of boolean functions f and g is defined on the truth tables

$$d(f, g) := d([f], [g])$$

Note that $d(S_1, S_2) = \{i|S_{1_i} \neq S_{2_i}\} = \{i|S_{1_i} \oplus S_{2_i} \neq 0\} = wt(S_1 - S_2)$ We will deal with non-linearity and Walsh Transform now:

2.10.1 Walsh Transform and Non-linearity

Definition 2.10.3. *Suppose we have a boolean function $f : \{0,1\}^n \to \{0,1\}$. The Walsh Transform is defined as follows:*

$$W_f(a) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus a \cdot x}$$

So for example,

$$\begin{aligned} W_f(\underbrace{0,0,\ldots,0}_{n \text{ times}}) &= \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \\ &= \#\{x|f(x)=0\} - \#\{x|f(x)=1\} \\ &= 2^n - wt(f) - wt(f) \\ &= 2^n - 2wt(f) \end{aligned}$$

We represent $h(a,x) = f(x) \oplus a \cdot x$. Then

$$\begin{aligned} W_f(a) &= \#\{x|f(x) = a\cdot x\} - \#\{x|f(x) \neq a\cdot x\} \\ &= \#\{x|h(a,x)=0\} - \#\{x|h(a,x)=1\} \\ &= 2^n - 2wt(h(a)) \end{aligned}$$

Definition 2.10.4. *Non-linearity of n-bit Boolean function f is defined as*

$$\eta(f) = \min_{g\in\mathcal{A}_n} d(f,g)$$

where $\mathcal{A}_n$ denotes the collection of all possible affine boolean functions with taking in n variables as the input.

It can easily be shown that, in terms of Walsh Transform of f, the non-linearity can be re-framed as

$$\eta(f) = 2^{n-1} - \frac{1}{2}\cdot \max_{a\in\{0,1\}^n} |W_f(a)|$$

We will later verify that $\eta(f)$ attains its maximum if and only if $|W_f(a)| = 2^{\frac{n}{2}}$, and thus

$$\max \eta(f) = 2^{n-1} - 2^{\frac{n}{2}-1}$$

If f follows the above maximum non-linearity, it is called a **bent-boolean**.

2.10.2 Oracle Function

The major problem that we face again is the fact Boolean functions are not unitary if the input has at least two variables. We can overcome this, however, by defining a new function

as follows:

Definition 2.10.5. *Let F be a boolean function of m-variables as stated above. Then the oracle function U_F is defined as follows:*

$$U_F : \{0,1\}^{m+n} \to \{0,1\}^{m+n}$$

with

$$(x, b) \overset{U_F}{\longmapsto} (x, b \oplus F(x))$$

where x and b are bit-strings of size m and n

Properties of U_F

- U_F is unitary
- $U_F \circ U_F = I$ implying $U_F^\dagger = U_F$
- Number of gates to implement U_F in a quantum system is polynomially related to the number of gates to implement F in a classical domain.

2.11 Faster Computations in the Quantum System: An Introduction to Deutsch Algorithm

The Deutsch algorithm is an example of one of the reasons quantum computers are better than their classical counterparts. Let us assume that we are provided with a Boolean function F taking in a bit-string of size n, which is either a constant or balanced, where "balanced" means that the size of the kernel of F is equal to the size of its support, i.e.,

$$\#\{x|f(x) = 0\} = \#\{x|f(x) = 1\}$$

To check if it is balanced or constant in the classical sense, we firstly store $F(0)$ and then check if any $F(x) \neq F(0)$ for $x \in \{1, 2, 3, \ldots, 2^{n-1}\}$. If any does, the algorithm returns that it is balanced; otherwise, it is constant. As we can see, the algorithm, in the worst case scenario, will take 2^{n-1} steps to verify if it is balanced and would otherwise return constant.

Firstly, we assume that f is a one-bit boolean function. The Deutsch Algorithm goes as follows:

1. Pass $|\psi_0\rangle = |01\rangle$ through $H^{\otimes 2} = H \otimes H$ to get $|\psi_1\rangle$:

$$\begin{aligned} H^{\otimes 2}(|01\rangle) &= \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) \\ &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \end{aligned}$$

2. Pass this new qubit $|\psi_1\rangle$ through an Oracle function U_f of input size 2 using the boolean function f, to get $|\psi_2\rangle$:

$$\begin{aligned} U_f(|\psi_1\rangle) &= \frac{1}{2}\Big((|0\rangle \otimes (|f(0) \oplus 0\rangle - |f(0) \oplus 1\rangle) + |1\rangle \otimes (|f(1) \oplus 0\rangle - |f(1) \oplus 1\rangle)\Big) \\ &= \frac{1}{2}\Big((|0\rangle \otimes (|f(0)\rangle - |\neg f(0)\rangle) + |1\rangle \otimes (|f(1)\rangle - |\neg f(1)\rangle)\Big) \\ &= \frac{1}{2}\Big((-1)^{f(0)}|0\rangle \otimes (|0\rangle - |1\rangle) + (-1)^{f(1)}|1\rangle \otimes (|0\rangle - |1\rangle)\Big) \\ &= \frac{1}{2}\Big((-1)^{f(0)}\big(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\big)\Big)(|0\rangle - |1\rangle) \\ &= \frac{1}{\sqrt{2}}\Big((-1)^{f(0)}\big(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\big)\Big) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= |\psi_{21}\rangle \otimes |-\rangle \qquad\qquad [\text{Assume}] \end{aligned}$$

3. Now Pass $|\psi_2\rangle$ through $H \otimes I$ to get $|\psi_3\rangle$. Since $I(|\psi_{22}\rangle) = |\psi_{22}\rangle$, we only need to calculate $H(|\psi_{21}\rangle)$ and thus $|\psi_3\rangle = H(|\psi_{21}\rangle) \otimes |\psi_{22}\rangle$. So

$$H(|\psi_{21}\rangle) = \frac{1}{2}[((-1)^{f(0)} + (-1)^{f(1)})|0\rangle + (((-1)^{f(0)} - (-1)^{f(1)})|1\rangle]$$

4. Thus the probabilities of finding each state is as follows:

$$p_0 := Pr_{H(\psi_{21})}(|0\rangle) = \frac{1}{4}((-1)^{f(0)} + (-1)^{f(1)})^2$$

$$p_1 := Pr_{H(\psi_{21})}(|0\rangle) = \frac{1}{4}((-1)^{f(0)} - (-1)^{f(1)})^2$$

5. Now using the idea that f is either constant or balanced, we see that:

   - $p_0 = 0$ and $p_1 = 1$ if f is balanced

- $p_0 = 1$ and $p_1 = 0$ if f is constant.

This has reduced the time to check by four steps! A modification of Deutsch called the Deutsch -Jozsa Algorithm, is used for a multi-bit system

Deutsch -Jozsa Algorithm

Assuming f is an n-bit boolean function, the Algorithm goes as follows:

1. Begin by passing $|\psi_0\rangle = |0^{\otimes n}\rangle|1\rangle$ through $H^{\otimes n+1} = H^{\otimes n} \otimes H$ to get $|\psi_1\rangle$

$$H^{\otimes n} \otimes H(|\psi_0\rangle) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle(|0\rangle - |1\rangle)$$

2. Pass $|\psi_1\rangle$ through the oracle function U_f of dimension n+1 to get $|\psi_2\rangle$

$$\begin{aligned} U_f(|\psi_1\rangle) &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle(|f(x)\oplus 0\rangle - |f(x)\oplus 1\rangle) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle(|f(x)\rangle - |\neg f(x)\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=1}^{2^n-1} (-1)^{f(x)}|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= |\psi_{21}\rangle \otimes |-\rangle \qquad \text{[Assume]} \end{aligned}$$

3. Pass $|\psi_{21}\rangle$ through $H^{\otimes n}$:

$$\begin{aligned} &\frac{1}{\sqrt{2^n}} \sum_{x=1}^{2^n-1} (-1)^{f(x)} \Big[\frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle\Big] \\ &= \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} \Big[(-1)^{f(x)\oplus x\cdot y}|y\rangle\Big] \\ &= \frac{1}{2^n} \sum_{y\in\{0,1\}^n} W_f(y)|y\rangle \end{aligned}$$

4. Thus probability of finding $|0^{\otimes n}\rangle$ is:

$$p := P_{H^{\otimes n}(|\psi_{21}\rangle)}(|0^{\otimes n}\rangle) = \frac{1}{2^{2n}} \sum_{x\in\{0,1\}^n} \left[(-1)^{f(x)}\right]^2 = \frac{1}{2^{2n}}|W_f(0)|^2$$

5. Again if f is constant $p = 1$, otherwise $p = 0$ if f is balanced. It has also taken four steps.

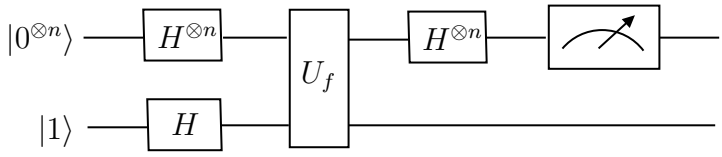

Figure 2.5: Quantum Circuit Diagram for Deutsch-Jozsa Algorithm

2.12 A New Way to Search: Grover's algorithm

Two of the major ways that quantum computing has threatened the field of classical cryptography are **Shor's Factorisation Algorithm** and **Grover's Search Algorithm**. The former factorizes any number into its coprime factors in polynomial time, thus weakening a majority of Discrete Logarithm Problems used in Public Key Cryptosystems, whereas Grover's Search is a significant improvement on classical algorithms, which can be improved further by searching for more than one single element (Which at classical search would rather worsen it). We will mainly focus on Grover's algorithm since we will cover attacks on private key ciphers.

We will assume that we will search for M elements in a set of size $N = 2^n$ elements, where n is a positive integer.

2.12.1 Grover's Oracle

The Oracle function for Grover's algorithm will be slightly different than the standard definition. We first define a set of solutions **S** (the elements we are searching for) of size M,

as a quantum state as

$$|\beta\rangle := \frac{1}{\sqrt{M}} \sum_{x \in \mathbf{S}} |x\rangle$$

We also define a quantum state for the set of non-solutions, $\mathbf{S}^{\perp}$ of size $N - M$, as

$$|\alpha\rangle := \frac{1}{\sqrt{N-M}} \sum_{x \in \mathbf{S}^{\perp}} |x\rangle$$

Added to this we define a Boolean function $f : \{0,1\}^N \to \{0,1\}$ as follows:

$$f(x) := \begin{cases} 1, & x \in \mathbf{S} \\ 0, & x \in \mathbf{S}^{\perp} \end{cases}$$

We now define the Grover's Oracle.

Definition 2.12.1. *The Grover's Oracle function $O_{\boldsymbol{S}}$ is a unitary operator on $\mathbb{CP}^N$, such that for all $x \in \{0,1\}^N$,*

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle$$

One question immediately arises after this definition: We have already defined our oracle gate in Definition 2.10.5, so why are we defining a new Grover's Oracle operator? The answer is quite simple: Grover's Oracle is, in fact, a specific case of an oracle gate. Notice that

$$\begin{aligned} |x\rangle(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \xmapsto{U_f} |x\rangle(\frac{|f(x)\rangle - |\neg f(x)\rangle}{\sqrt{2}}) \\ = (-1)^{f(x)}|x\rangle(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \end{aligned}$$

Thus $U_f = O_{\mathbf{S}} \otimes I$ if the last qubit is $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$! The practical implications are that if one doesn't know the solution state but rather its conditions, one can simply define its indicator function f, and that can still be used to define Grover's Oracle on f, using simply the oracle on f. It is also fast in the sense that instead of creating a quantum gate of size n out of fundamental gates, we can simply define U_f of size $n+1$ whose construction is polynomially bounded, and use that in place of $O_{\mathbf{S}}$. We will now define something called a conditional phase shift operator.

Definition 2.12.2. *The Conditional Phase Shift Operator, P, is defined as a unitary opera-*

tor that flips the phases of all elementary qubits except $|0\rangle$*. In other words, for all* $x \in \{0,1\}^n$

$$P|x\rangle = \begin{cases} |x\rangle & , x = 0 \\ -|x\rangle & , otherwise \end{cases}$$

In terms of matrices, P over dimension n can be represented as

$$\begin{aligned} P &= 2|0\rangle\langle 0| - I_n \\ &= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & \dots & 0 \\ 0 & 0 & -1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -1 \end{bmatrix} \end{aligned}$$

Also note that $\tilde{P} = -P$ will also work in the search algorithm, with the only difference being that the phase of a solution state qubit is flipped by π, i.e., instead of getting $|\beta\rangle$ we instead get $-|\beta\rangle$. Now $\tilde{P}$ can be simply implemented using a simple $n+1$ fold CCNOT gate , using the same $|-\rangle$ as the last qubit

2.12.2 The Algorithm

1. Apply H_{n+1} on $|0^{\otimes n}\rangle|1\rangle$
2. Apply U_f
3. On the first n-bits apply H_n, P and H_n in this order.
4. Repeat steps 2 and 3, for about $O(\sqrt{\frac{N}{M}})$ times
5. Measure the first n-bits

.

Step 2, followed by 3, form what is called the **Grover's operator**. In other words, Grover's operator is defined as

$$G = H_n \circ P \circ H_n \circ O_{\mathbf{S}}$$

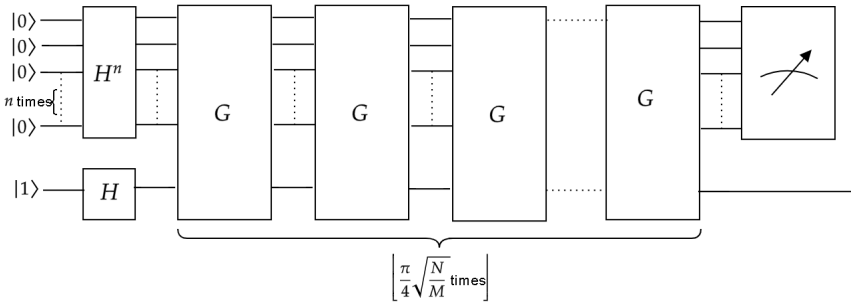

Figure 2.6: The Grover's Algorithm in a Quantum Circuit

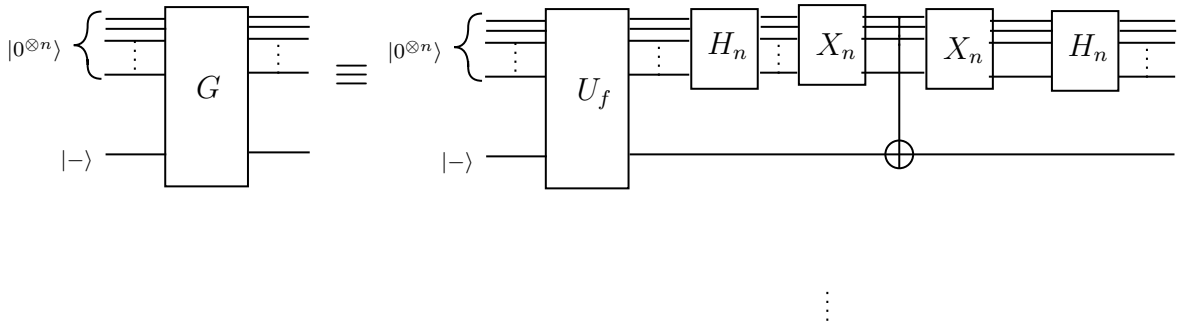

Figure 2.7: The Grover's Operator in a quantum circuit

2.12.3 Mathematical Interpretation

To understand how this algorithm works, we need to see what Grover's Operator G does.

Firstly, we view $|\psi\rangle = H_n(|0^{\otimes n}\rangle) = \sum_{x \in \{0,1\}^n} |x\rangle$ as a linear combination of $|\alpha\rangle$ and $|\beta\rangle$. Using simple algebra,

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle$$

Since both the coefficients lie in $[0, 1]$, we assign an angle $\theta \in [0, \pi]$ such that

$$\cos\frac{\theta}{2} = \sqrt{\frac{N-M}{N}} \text{ and } \sin\frac{\theta}{2} = \sqrt{\frac{M}{N}}$$

and thus

$$|\psi\rangle = \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle,$$

and since $|\alpha\rangle$ and $|\beta\rangle$ are orthonormal to each other, one can represent that $|\psi\rangle$ aligned between these two. Now we will notice the fact that

$$\begin{aligned} H_nPH_n &= H(2|0\rangle\langle 0| - I_n)H \\ &= 2H_n|0\rangle\langle 0|H_n - H_nI_nH_n \\ &= 2|\psi\rangle\langle\psi| - I_n \end{aligned}$$

Now we will view the transformation of $|\psi\rangle$ by G as follows:

$$|\psi\rangle \overset{O_S}{\longmapsto} |\psi_1\rangle \overset{H_nPH_n}{\longmapsto} |\psi_2\rangle$$

The first transformation tags the solutions as follows:

$$\begin{aligned} |\psi_1\rangle &= O_S(|\psi\rangle) \\ &= O_S(\cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle) \\ &= \cos\frac{\theta}{2}|\alpha\rangle - \sin\frac{\theta}{2}|\beta\rangle \end{aligned}$$

The second transformation will now rotate it towards the solution state $|\beta\rangle$:

$$\begin{aligned} |\psi_2\rangle &= H_nPH_n(|\psi_1\rangle) \\ &= (2|\psi\rangle\langle\psi| - I_n)(\cos\frac{\theta}{2}|\alpha\rangle - \sin\frac{\theta}{2}|\beta\rangle) \\ &= 2|\psi\rangle(\cos\frac{\theta}{2}\langle\psi\,|\,\alpha\rangle - \sin\frac{\theta}{2}\langle\psi\,|\,\beta\rangle) - \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle \\ &= 2(\cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle)(\cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2}) - \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle \\ &= 2\cos\theta(\cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle) - \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle \\ &= (2\cos\theta - 1)\cos\frac{\theta}{2}|\alpha\rangle + (2\cos\theta + 1)\sin\frac{\theta}{2}|\beta\rangle \\ &= (2(2\cos^2\frac{\theta}{2} - 1) - 1)\cos\frac{\theta}{2}|\alpha\rangle + (2(1 - 2\sin^2\frac{\theta}{2}) + 1)\sin\frac{\theta}{2}|\beta\rangle \\ &= (4\cos^3\frac{\theta}{2} - 3\cos\frac{\theta}{2})|\alpha\rangle + (3\sin\frac{\theta}{2} - 4\sin^3\frac{\theta}{2})|\beta\rangle \\ &= \cos\frac{3\theta}{2}|\alpha\rangle + \sin\frac{3\theta}{2}|\beta\rangle \end{aligned}$$

In short, if solutions exist, the state $|\psi\rangle$ will rotate by twice the angle it is projected on $|\alpha\rangle$. Also, though not shown further, using G n-times, rotate $|\psi\rangle$ by $n\theta$. To prevent it from over-rotation, $n\theta$ should be less than $\pi/4$. Thus, the optimal number of rotations should be $\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \rfloor$. Thus the operator is used $O(\sqrt{\frac{N}{M}})$.

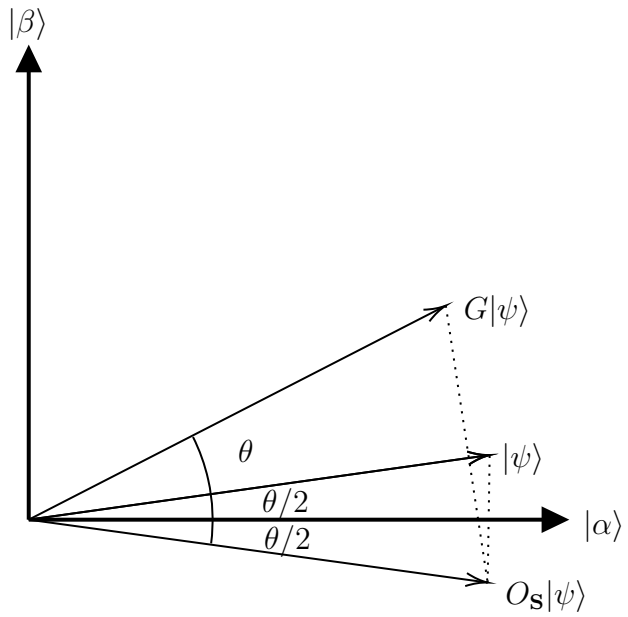

Figure 2.8: Geometrical interpretation of the action of a simple Grover iteration G unto the solution state $|\beta\rangle$ vs the non-solution state $|\alpha\rangle$

Chapter 3

Classical Cryptanalytic Techniques

Definition 3.0.1. *Cryptanalysis, the branch of cryptology attacking ciphers, seeks to gain concealed information about these systems through well-designed attacks. Its primary purpose is to breach ciphers and gain access to the contents of original messages, even when the cryptographic key remains a secret.*

Beyond the mathematical dissection of cryptographic algorithms, cryptanalysis encompasses the exploration of side-channel attacks, which exploit not the cryptographic algorithms' weaknesses but rather vulnerabilities in their execution.

Sometimes, the attacks can be used to extract keys, whereas sometimes, they will try to defunct some properties that a cipher must have, like "indistinguishability."

3.1 Necessity of Cryptanalysis

Cryptanalysis serves a dual purpose—it's not just about breaking encryption. It plays a crucial role in assessing the security of encryption protocols and ciphers, pinpointing vulnerabilities that need fortification rather than exploitation. This essential practice ensures the robustness of encrypted messages and data, fostering a comprehensive understanding of their contents, which, in turn, facilitates access to data, whether it's in transit or at rest. Cryptanalysis is the cornerstone of safeguarding sensitive information and maintaining the

integrity of secure communication.

There are three major properties that are used in general that will cause hindrance for a cryptanalyser.

Definition 3.1.1. *(**Shannon's Perfect Secrecy**): Given a key, the probability of choosing a ciphertext should be independent of the choice of the plaintext, and the reverse should also be true.*

Definition 3.1.2. *(**Shannon's Secrecy**:) For a system with $\mathcal{K}, \mathcal{M}, \mathcal{C}$, denoting the keyspace, the plaintext space, and the ciphertext space, respectively, with $|\mathcal{M}| = |\mathcal{C}|$, the size of the keyspace the size of $|\mathcal{M}|$.*

Theorem 3.1.1. *(**Kerchoff's Principle**) : Security depends not on concealing the algorithm but solely on safeguarding the key, which must remain a solitary secret.*

3.2 Types of Cryptanalysis

In the realm of cryptographic attacks, we encounter these distinct strategies:

- Ciphertext-Only Analysis (COA): Only using the ciphertext, the attacker predicts the plaintext and the key with a high probability.
- Known Plaintext/Ciphertext Analysis (KPA/KCA): The attacker knows a plaintext and ciphertext pair, and using that, they find out the key
- Chosen Plaintext/Ciphertext Analysis (CPA/CCA): The attacker chooses a particular plaintext for which he gets back a ciphertext, and using that, they figure out a key
- Adaptively Chosen Plaintext/Ciphertext Analysis **(ACPA/ACCA)**: Instead of choosing one single plaintext, the attacker uses the ciphertext to generate a plaintext, which generates another ciphertext, which, on comparison, can help figure out the key.

3.3 Some Classical Attacks

3.3.1 Differential Cryptanalysis

Differential cryptanalysis [9] is a general chosen plaintext attack that attempts to predict if a change in a sequence of bits in the plaintext results in a desired change in the ciphertext and, if possible, what sort of round keys are ideal for guessing. The attack primarily exploits the concept of indistinguishability in that the attacker may determine if the provided plaintext is encrypted using a real random function and the "indistinguishable" cipher. It is assumed that the attacker is already aware of the round-key-generation algorithm. Assume that Δ modifies one bit of plaintext. The change in ciphertext caused by the change in plaintext is denoted by $\Delta_P \rightarrow \Delta_C$. Then, the probability of the following events is always one:

- Change in ciphertext = Change in plaintext after plaintext enters an XOR addition

$$Pr(\Delta \xrightarrow{\oplus} \Delta) = 1$$

- Change in ciphertext = L(Change in plaintext) after plaintext enters a linear function L

$$Pr(\Delta \xrightarrow{L} L(\Delta)) = 1$$

If the encryption just employs XOR and linear operations, predicting a key becomes trivial. As a result, many ciphers utilize a nonlinear function, such as an S-box or a SubBytes function. In addition, these ciphers use a quasi-uniform difference propagation, which means that just one difference has a high likelihood of appearing, but only a modest probability of appearing over every other combination of differences. Consider AES-128, where the SubBytes Differential Distribution Table reveals the following distribution (note that Δ_{IN} represents the difference in input and Δ_{OUT} represents the difference in output):

- 129 cases of Δ_{IN} with no change in output ($\Delta_{OUT} = 0$)
- 126 cases of Δ_{IN} with $\Delta_{OUT} = 2$
- 1 specific case of Δ_{IN} with $\Delta_{OUT} = 4$

As a result, we don't describe exactly what the ciphertext change would be, but rather the structure of this change, which aids us in determining the cipher's distinguishability. As an example, suppose we try to break the AES encryption for four rounds. We will assume, in this case, a specific Δ_P:

$$\Delta_P = \begin{bmatrix} 0 & 0 & 0 & f \\ g & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & i & 0 \end{bmatrix}$$

such that it transforms itself with a high likelihood (approximately $\frac{1}{2^{24}}$) to

$$\Delta S_1^{AK} = \Delta P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

To put it another way, we're looking for a Δ_P that propagates to a singleton cell difference after one round. We then attack normally, spreading any conceivable differences across three further rounds.

This is shown in the figure 3.1:

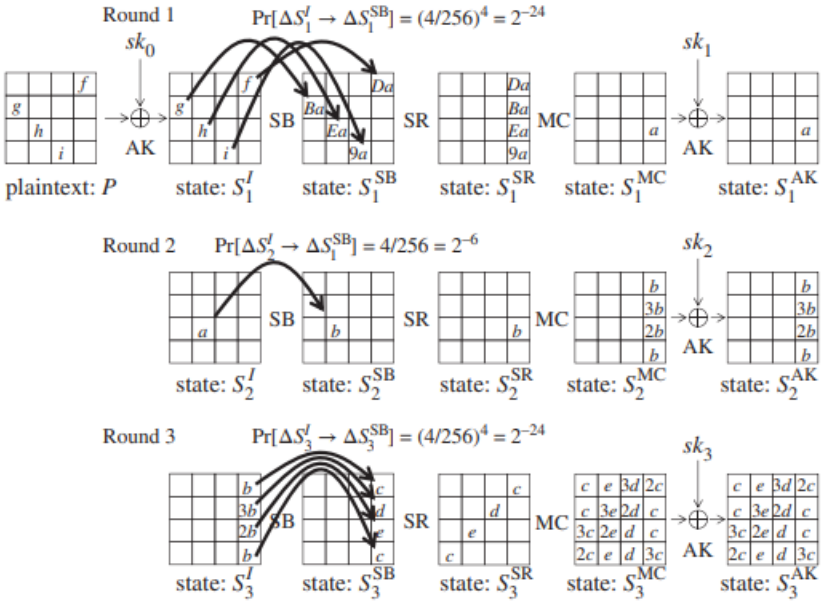

Figure 3.1: Difference Propagation in Reduced 4-round AES

3.3.2 Differential Fault Attack

This method is mainly used in the case of stream ciphers, as it requires the solution of many non-linear equations. Since the key is used to generate the keystream, the key is considered an unknown vector $(x_0, x_1, \cdots, x_n)$, leading to a series of non-linear equations. Then a fault is forcefully introduced in one of the variables, assume it was in the i-th position $(x_0, x_1, \cdots, x_{i-1}, x_i \oplus 1, x_{i+1}, \cdots, x_n)$. This will result in generating a new keystream, leading to a series of new equations. This fault is introduced in other key bits until all of the key bits are found.

3.3.3 Linear Attack

Linear cryptanalysis [10] is a general known-plaintext attack that could potentially implemented against both block and stream ciphers. The attack aims to discover an affine approximation to the given cipher in terms of the bits in key, plaintext, and ciphertext. We strive to locate a propagation with a high probability, similar to the differential counterpart. Still, instead of differences, we check on a new property called masks,' and utilizing that, a key recovery attack can be utilized. Alternatively, an attempt to crack a cipher's "confidentiality" property can be made using this strategy.

An easy example for understanding this is as follows: Take a simple cipher z taking two-bit inputs x and y, i.e., $x, y \in \{0, 1\}$ as

$$z : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$$
$$z(x, y) := x \cdot y$$

Clearly, z is not a linear map, let alone an affine one. All possible linear functions with domain $\{0, 1\}^2$ are of the form

$$f(x, y) = a_0 x \oplus a_1 y$$

where $a_0, a_1 \in \{0, 1\}$. Now we represent all possible combinations of f and check if they are equal to z, in table 3.1

This shows that every linear function in $\{0, 1\}^2$ is the same as z, with a probability of $\frac{3}{4}$, except for $x \oplus y$, with a probability of $\frac{1}{4}$. Now, we can use the fact that since the functions

Input (x, y)	**Output** $z = xy$	0	x	y	$x \oplus y$
$(0, 0)$	0	0	0	0	0
$(0, 1)$	0	0	0	1	1
$(1, 0)$	0	0	1	0	1
$(1, 1)$	1	1	0	1	0

Table 3.1: Comparison of all linear functions to the desired output

are affine, we simply take $x \oplus y \oplus 1$, which now matches with a probability of $\frac{3}{4}$. So, there are four candidates to approximate our cipher.

Now, our main focus is to find linear equations that simplify our cipher to a simple linear transformation. The masks are just coefficients to our input to get the desired linear equation. Mathematically, if E is our function taking inputs in $\{0,1\}^n$ as x, and gives an output of the same dimension, the masks are $\alpha, \beta \in \{0,1\}^n$ such that with a high probability

$$\langle \alpha, (x)_2 \rangle = \langle \beta, (E_K(x))_2 \rangle$$

where $(y)_2 = (y_{n-1}, y_{n-2}, \ldots, y_1, y_0)$ such that $y_i \in \{0,1\}$ for all $0 \leq i < n$
and , $\sum_{i=0}^{n-1} y_i 2^i = y$

3.3.4 TMTO

In TMTO (Time-Memory Trade-Off), we attack the cipher by generating as many keys as possible. As stated by the name, we decrease the amount of time taken by increasing the amount of memory used by the attack. This ratio of trade-off is proportional to $TM^2 = N^2$, and a typical setting used is $T = M = N^{\frac{2}{3}}$ where N is the size of the plaintext/ciphertext, M is the memory allocated, and T denotes time complexity of the attack. One of the procedures to generate a table of keys is as follows:

1. We define a function that linearly reduces (or increases) any ciphertext to some arbitrary key. We call it the reduction function R. To generate one key from another, we define $f(K) := R(C(T_P, K))$
   where C is the encryption function and T_P is a fixed plaintext

2. m distinct Start Points $(S_1, S_2, S_3, \cdots, S_m)$ are taken and passed through a linear

function f t times. We call the last points generated to be End Points. We define the iteration as $X_{j,i+1} := f(X_{j,i})$, where $X_{j,i}$ is the i-th iteration on S_j and $E_j := X_{j,t}$. We will have:

$$\begin{array}{lcccccccc} S_1 & = & X_{1,1} & \xrightarrow{f} & X_{1,2} & \xrightarrow{f} & X_{1,3} & \xrightarrow{f} & \cdots & X_{1,t} = E_1 \\ S_2 & = & X_{2,1} & \xrightarrow{f} & X_{2,2} & \xrightarrow{f} & X_{2,3} & \xrightarrow{f} & \cdots & X_{2,t} = E_2 \\ & \vdots & & & & \ddots & & & & \vdots \\ S_m & = & X_{m,1} & \xrightarrow{f} & X_{m,2} & \xrightarrow{f} & X_{m,3} & \xrightarrow{f} & \cdots & X_{m,t} = E_m \end{array}$$

3. Only the Start and End Points are stored, thus making a table and reducing the memory used. Then the original ciphertext, assume T_C, is passed through the same iteration, and this is done until one of the iterated values, say Y_s matches with any of the End Points, say E_{j_0}:

$$R(T_C) = Y_1 \xrightarrow{f} Y_2 \xrightarrow{f} Y_3 \xrightarrow{f} \cdots Y_s = E_{j_0}$$

4. The chain of $S_{j_0} \xrightarrow[t \text{ times}]{f} E_{j_0}$ is made again, but this time, each of the $X_{j_0,i}$ keys is checked if they satisfy the ciphertext. If none of the keys match, a false alarm is triggered, and the endpoint is discarded. The iteration Y_s is continued until one of the keys matches or all the End Points are discarded.

This is called the Hellman Table attack, [11]. Though this method produces many keys, the chances of the Table producing merging and cyclic chains are still present. Methods like ensuring the endpoints are distinct and making the Table perfect (eliminating the merging and cyclic chains) came at the cost of fewer keys.

There is also a modified form of TMTO attacks called the Rainbow Table attack [12], which is quite similar to Hellman, except we use reduction functions that depend on each iteration R_i, each pairwise independent. The functions f_i are similarly constructed. Perfect tables are also simple to create as only elements need to be checked row-wise and not column-wise because the f_i's being distinct and independent ensures that the same elements in two different columns will not produce the same elements after their respective f_i's are applied, thereby eliminating the chance of merging and cyclic chains. The way of checking the keys is different, but it is also compact. The ciphertext is first passed through R_{t-1}, and the last column of the Table is checked. If any key matches, we search the entire chain for keys. Otherwise, we apply R_{t-2}, f_{t-1} and search if this generated key matches those in the second last column. This goes on until all the keys are exhausted or the key is found.

3.3.5 TMDTO

TMDTO (Time-Memory-Data Trade-Off) [13] is a specialization of the Hellman TMTO attack to specifically exploit stream ciphers. The attack is exactly the same, with a further addition that we look further into specific data windows. We check D data points in these generated keystreams and check if those match with the original keystream. In doing so, the attack time is further reduced to $TM^2D^2 = N^2$, with a preferable setting being $T = N^{\frac{2}{3}}$ and $M = D = N^{\frac{1}{3}}$.

3.3.6 Biclique Attack

The biclique attack is a general chosen plaintext/ciphertext attack, a modified MITM (meet-in-the-middle) attack, which leverages the cipher's functionality and hence necessitates a thorough grasp of the encryption. It is the only known classical attack that can completely crack AES-k for all $k \in \{128, 192, 256\}$. The attack is inspired by Graph Theory, especially in the construction of bicliques. In this section, we will concentrate on a d-dimensional biclique on the Plaintext to State of length r.

The cipher E of R rounds may be decomposed as $E = g \circ h \circ f$, with f, g, and h signifying the first r_1, the last r_2, and the intermediate $R - (r_1 + r_2)$ rounds of the cipher respectively.

, for which we shall construct a biclique, h denotes the last r_2 This is a plaintext-based biclique as we are creating the Assume the basis key set is $\{K[0, 0]\}$ We choose some P_0, and $S_0 = f_K[0, 0](P_0)$ instead of creating random keys, plaintexts, and states to create a biclique. Then we employ the following:

- Set some ∇_j^K and Δ_i^K
- Find ∇_j and Δ_i such that:

$$\Delta_i \xleftarrow[f^{-1}]{\Delta_i^K} 0, \text{ and } 0 \xrightarrow[f]{\nabla_j^K} \nabla_j$$

Thus, we get the following relation

$$\Delta_i \xrightarrow[f]{\Delta_i^K \oplus \nabla_j^K} \nabla_j$$

and thus

$$P_0 \oplus \Delta_i \xrightarrow[f]{K[0,0] \oplus \Delta_i^K \oplus \nabla_j^K} S_0 \oplus \nabla_j$$

- Set

$$\begin{aligned} P_i &= P_0 \oplus \Delta_i \\ S_j &= S_0 \oplus \nabla_j \\ K[i,j] &= K[0,0] \oplus \Delta_i^K \oplus \nabla_j^K \end{aligned}$$

Now if the collection $\{P_i\}_{i=1}^{i=2^d-1}$ and $\{S_j\}_{j=1}^{j=2^d-1}$ were vertices of a graph, clearly $K[i,j]$ would be its edges and each P_i was connected to S_j by the key $K[i,j]$. All that remains is to compute bicliques for each base key, and we have a better brute force. To improve this computation further, we use a Meet-in-the-Middle (MITM) attack between g and h to simplify the issue. With $g(S_j, K[i,j]) = \vec{v_j}$ and $h^{-1}(C_i, K[i,j]) = \vec{v_i}$, we check $\vec{v_i} \stackrel{?}{=} \vec{v_j}$. The total time complexity (C_{attack}) of this attack would be as follows:

$$C_{attack} = 2^{n-2d}(C_{biclique} + C_{precomputation} + C_{recomputation} + C_{false\ positive})$$

. The structure of the attack is shown in Fig. 3.2.

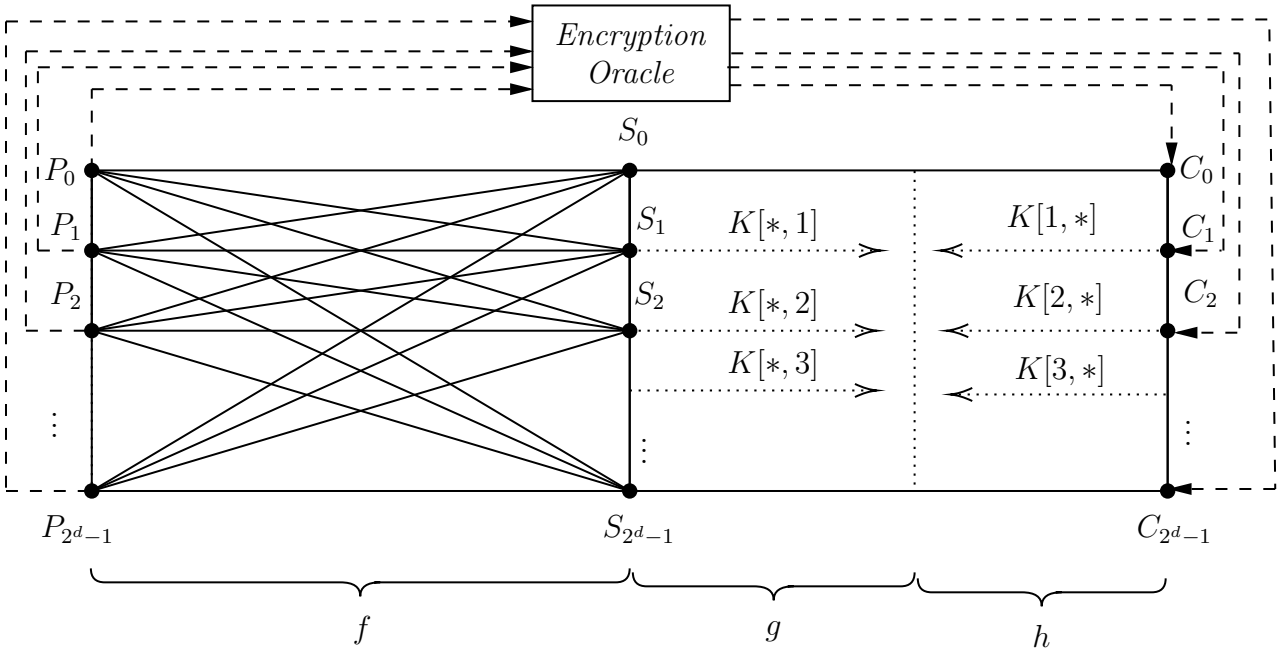

Figure 3.2: The basic demonstration of a biclique attack. Note that we also check for $S_j \xrightarrow[g]{K[*,0]} \overrightarrow{v_j} \stackrel{?}{=} \overleftarrow{v_i} \xleftarrow[h^{-1}]{K[*,0]} C_i$.

Chapter 4

Quantum Analogue to a Biclique Attack

4.1 Biclique attack on AES-128

We will use the indices given in the diagram given below to understand our attack:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Figure 4.1: Indices for each byte of a key of size 128

The main contribution of this thesis is to implement a quantum biclique attack on AES-128, a modification of the classical attack given by Bogdanov et al. We are going to focus on their independent biclique attack on AES-128. Instead of going in the plaintext direction, they create the biclique from the ciphertext since one of the processes (MixColumns) is skipped in the last round of encryption. Again they have broken $E_{AES-128} = f \circ g \circ h$, where f consists of the last 3 rounds of AES-128, g the intermediate 2 rounds, and h the first 5

rounds. They have chosen $d = 8$ and the base key space as

$$K[0,0] = \{*0^8 \underbrace{* * * \cdots *}_{\text{10 bytes}} 0^8 * **\},$$

where $*$ is a random element in $\mathbb{F}_2^8$. The following is a diagrammatic representation of this:

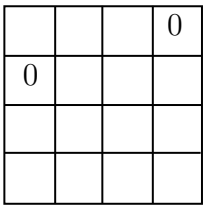

Figure 4.2: Base Key Initialisation

They chose $\Delta_i^K = (\underbrace{0^8 0^8 0^8 \ldots 0^8}_{\text{8 bytes}}\ i\ 0^8 0^8 0^8\ i\ 0^8 0^8 0^8)$ and $\nabla_j^K = (0^8\ j\ \underbrace{0^8 0^8 0^8 \ldots 0^8}_{\text{7 bytes}}\ j\ \underbrace{0^8 0^8 0^8 \ldots 0^8}_{\text{6 bytes}})$. Again, their diagrammatic representation is as follows:

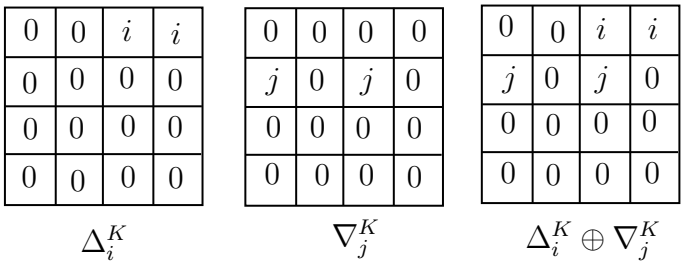

Figure 4.3: Their Addition of Key Differences

The choice of Δ_i^K and ∇_j^K is because it forms a proper 3 round Δ_i and ∇_j. The structures of them are in the image 4.4

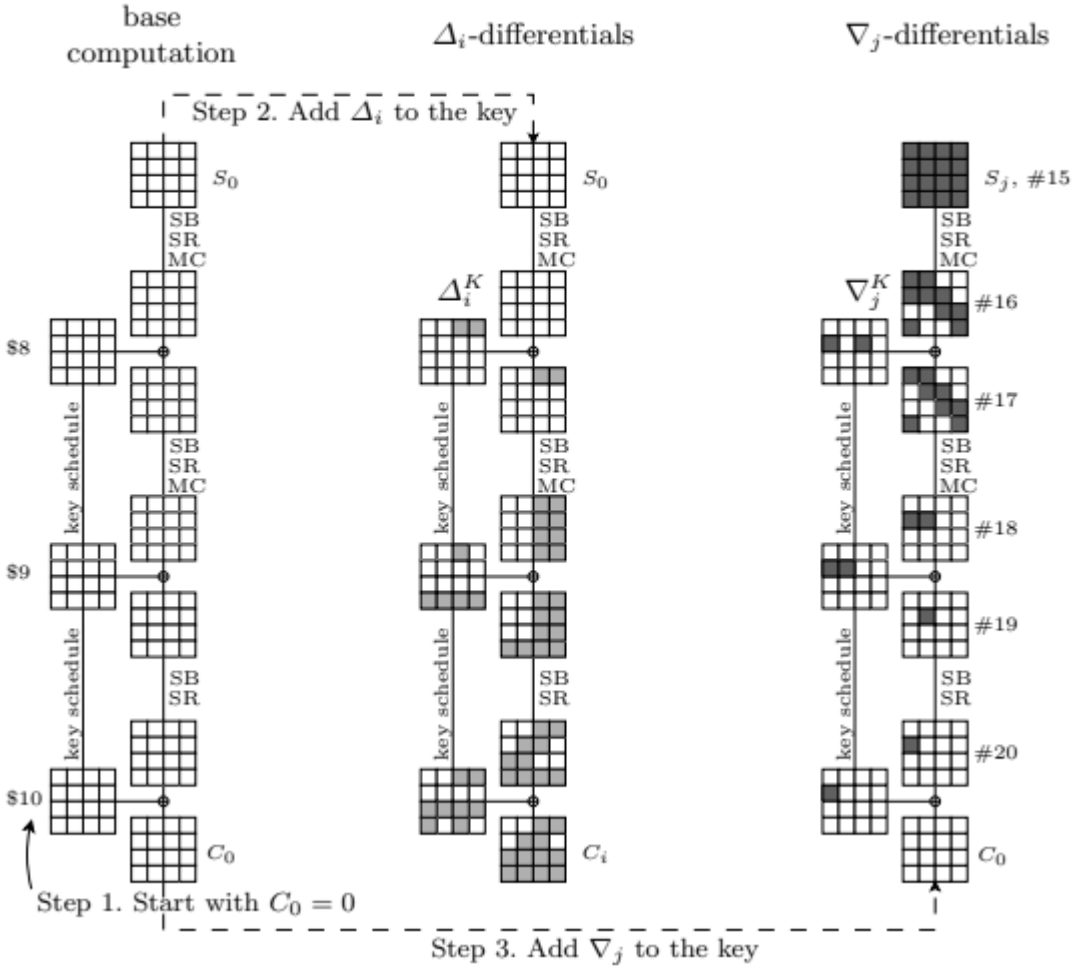

Figure 4.4: Their computation of Δ_i and ∇_j

Now, searching for the key with this biclique will again take 2^{128}, and the attack will not be better than a brute force.

Instead, we decrypt C_i to its P_i with respect to the master key we need to find using the decryption oracle provided by the challenger, which is the reverse of the encryption oracle, as stated above. We employ a search, the precomputation of $P_i \xrightarrow[h]{K[i,0]} \vec{v}$ and $S_j \xrightarrow[g^{-1}]{K[0,j]} \overleftarrow{v}$. So the recomputation $P_i \xrightarrow[h]{K[i,j]} \vec{v}$ and $S_j \xrightarrow[g^{-1}]{K[i,j]} \overleftarrow{v}$ will involve fewer non-linear parts to our encryption. This reduces computation time further. However, we can't scan the entire $\vec{v}$ as it will rather increase the computation time. They now focus on a certain byte position such that, with a high probability, $\vec{v} = \overleftarrow{v}$ only if $K[i, j]$ is the master key. All of this is described

in the figure 4.5. However, there is a chance to get a false positive.

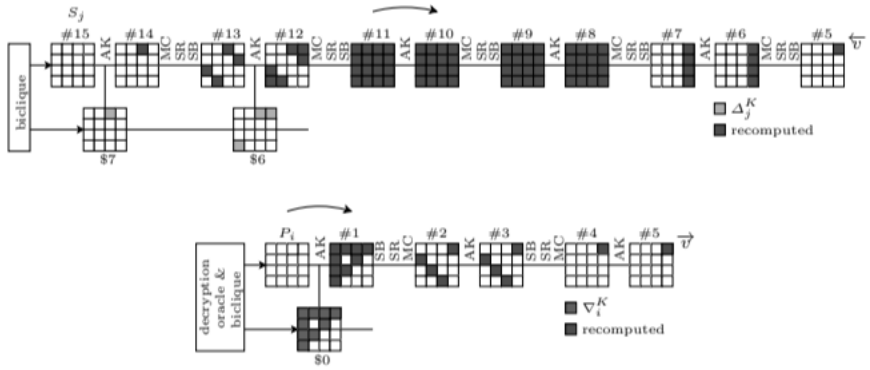

Figure 4.5: Their recomputation in the forward (top) and backward (bottom) direction

Their attack has thus a time complexity of

$$C_{attack} = 2^{112}(2^7 + 2^7 + 2^{14.14} + 2^8) = 2^{126.18}$$

4.2 Quantum Implementation of Biclique Attack

So far, we've seen some quantum algorithms and how they significantly sped up some algorithms. It is thus natural that cryptanalysis would soon adopt these quantum algorithms to these classical attacks. So far, all of them except the biclique attack have a quantum alternative. However, most of these attacks only focused on query complexity, and there was no clear answer to time-space complexity in a quantum computer, as there was no clear understanding as to what led to a change in time/memory in a quantum complexity. Some examples of these attacks are the quantum analog of a generalized TMDTO attack with an improvement as $N^3 = T^2M^3D^3$ and a quantum related-key attack on GRAIN-128a, a significant improvement on their previous classical attack.

However, after this, Grassl [14] designed AES-128 in a quantum circuit, mainly using the Clifford-T universal gate set and launching a brute-force attack. Exploiting the fact

that AES follows SKAC,[1], they check in Grover's Oracle r different plaintext-ciphertext pair corresponding to the master key, where

$$r = \begin{cases} 3 & \text{for AES-128} \\ 4 & \text{for AES-192} \\ 5 & \text{for AES-256} \end{cases}$$

Even after many checks, they perform this brute-force attack within a T-complexity of 2^86 for AES-128, implying a quantum AES-128 attack is possible and feasible.

The tables of the gate complexity on the implementations and the attacks are given in the tables below:

Key	Number of gates		depth		Number of
Size	T	Clifford	T	Overall	qubits
128	1060864	1380420	50688	110799	984
192	1204224	1567296	44352	96956	1112
256	1505280	1956099	59904	130929	1336

Table 4.1: Quantum Resource Estimates for Implementation of AES

[1]A cipher follows Key Avalanche Criterion (KAC) if keeping the plaintext fixed, a small change in the key induces a large change in the ciphertext. The Strong Key Avalanche Criterion surpasses the previous definition of strength. A cipher E follows SKAC if for two distinct keys $K \neq K'$ and a fixed plaintext P of size n, if $S = E(K, P) \oplus E(K', P)$ then $Pr(S_i = 1) > \frac{1}{2}$ for all $0 \leq i < n$, where $S = \sum_{i=0}^{2^n-1} S_i 2^i$ This is often considered a distinct advantage criterion for a classical cipher

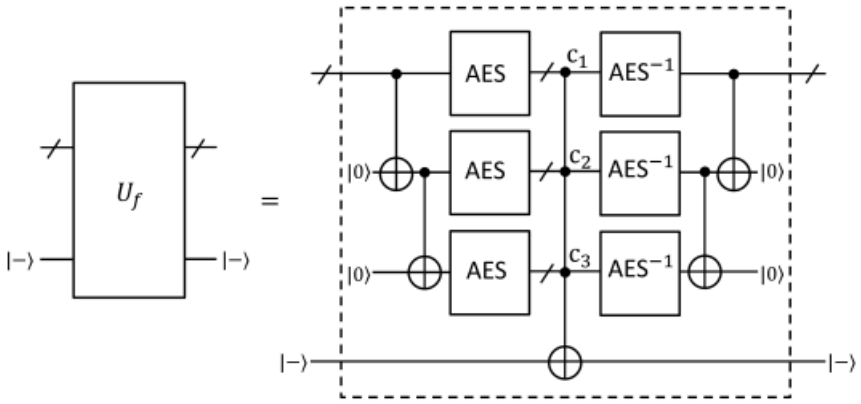

Figure 4.6: U_f for Grover's search in the Brute-force attack

Key Size	Number of gates		depth		Number of qubits
	T	Clifford	T	Overall	
128	$1.19 \cdot 2^{86}$	$1.55 \cdot 2^{86}$	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	2953
192	$1.81 \cdot 2^{118}$	$1.17 \cdot 2^{119}$	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	4449
256	$1.41 \cdot 2^{151}$	$1.83 \cdot 2^{151}$	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	6681

Table 4.2: Quantum Resource Estimates for Bruteforce of AES

Steps Involved	Number of gates		depth		Number of qubits
	T	Clifford	T	Overall	
Initial	0	0	0	0	128
Key Gen	143360	185464	5760	12626	320
10 rounds	917504	1194956	44928	98173	536
Total	1060864	1380420	50688	110799	984

Table 4.3: Detailed Quantum Resource Estimates on Implementation of AES-128

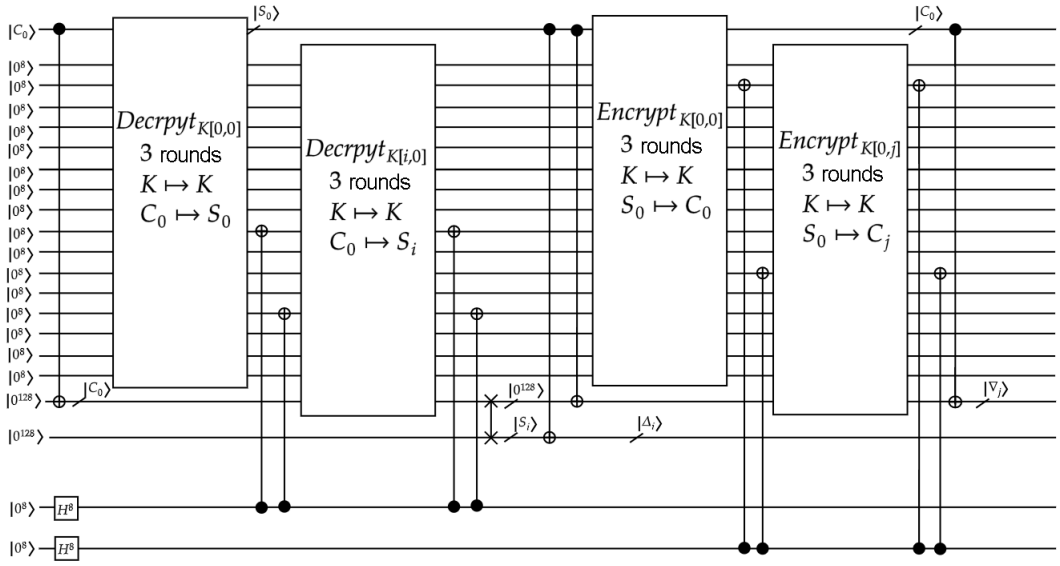

Figure 4.7: Generation of Δ_i from Δ_i^K and ∇_j from ∇_j^K

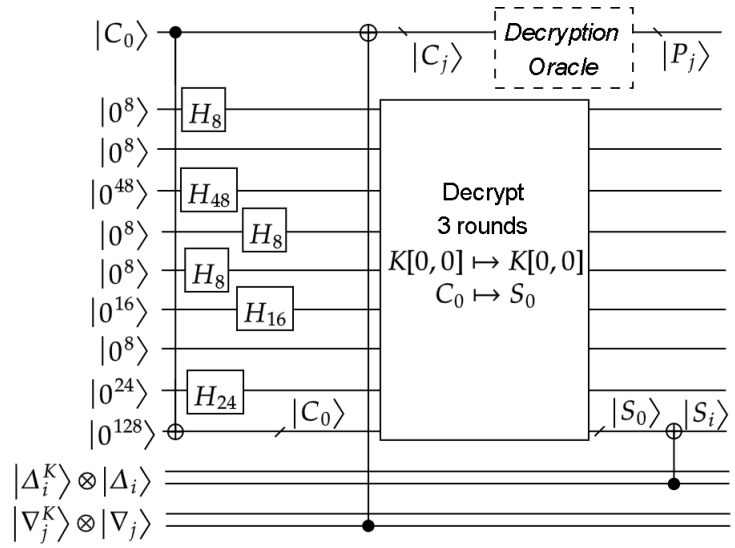

Figure 4.8: Generation of P_j and S_i for each K[0,0] grouping

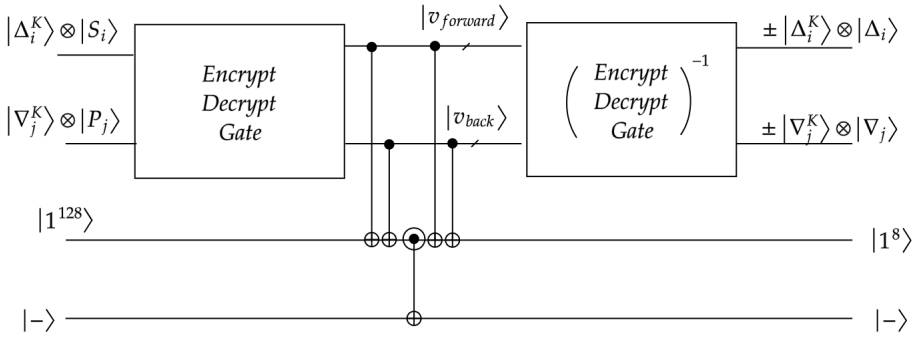

Figure 4.9: U_f for Biclique Grover's Search. Note that even though not shown, the input also has the entire K[0,0] block

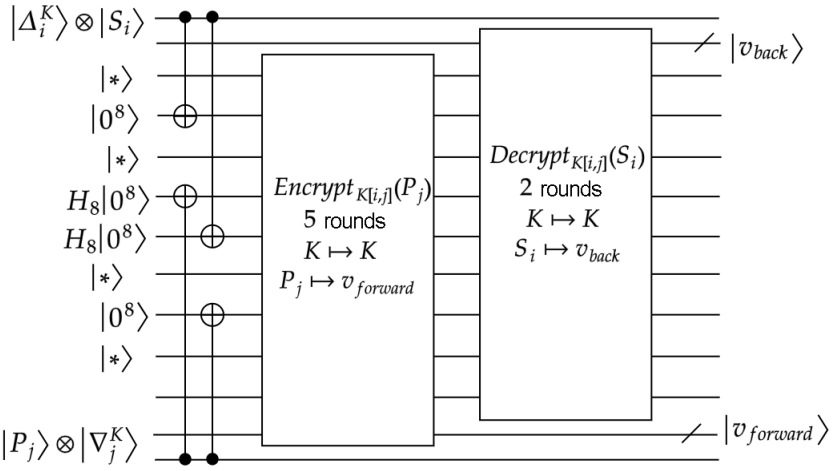

Figure 4.10: The Encrypt-Decrypt Gate in U_f in figure 4.9 Note that the inverse of this gate is the exact inverse of this circuit

So, the quantum biclique attack is applied in the following order:

- Generating Δ_i from Δ_i^K and ∇_j from ∇_j^K as in figure 4.7
- Generation of Pj and S_i for each K[0,0] grouping as in figure 4.8

- Grover's search using oracle gate

$$U_f(K[i,j]) = \begin{cases} 1 & if \text{Enc}_5(K[i,j], P_j) = \text{Dec}_2(K[i,j], S_i) \\ 0, & \text{otherwise} \end{cases}$$

  which are constructed using Figures 4.9 and 4.10

The time complexity has been significantly reduced to around 2^{64} since the first two generation blocks are computed once, implying they do not add much time complexity, compared to Grover's search component being repeated $O(\sqrt{N})$ times. but now the question arises about the resource constraints of the attack. We will use the table 4.3 in particular in my design of a biclique attack. Thus, the T-complexity will be as follows:

1. The Encryption and Decryption will take the entire Key Generation, so all of the T gates will be counted as well. Otherwise, 3 out of 10 rounds of Encryption/Decryption is done here.
2. Again, the same follows here.
3. For U_f, for the Encryption process, only 5 out of 10 rounds of Key Generation are required, and 7 out of 10 rounds for Decryption. Again, as a whole, 5 out of 10 rounds are used for encryption, and 2 out of 10 rounds are used for decryption. Also, borrowing from Grassl et al., a 128-fold Toffoli gate should take 1000 T-gate.

Therefore the total T-complexity is

$$\begin{aligned} T_{biclique} &= 5(143360 + \frac{3}{10}917504) + \lfloor\frac{\pi}{4}2^{64}\rfloor(\frac{5\times 2}{10}(1060864) + 2(\frac{7}{10}143360 + \frac{2}{10}917504) + 2\times 1000) \\ &\approx 2.36 * 10^{25} \approx 1.22 * 2^{84} \end{aligned}$$

.

This is a significant improvement over the brute force on AES-128 given by Grassl et al. The Clifford gate calculation would be similar, except a 128-fold Toffoli gate will use 3012 Clifford gates. For T-depth, it was found that the slightly lower blocks in the entire circuit have a higher number of T-gates, and T-depth is equal to the depth of these smaller circuits$(3\times 5760 + 0.9\times 44928) + \lfloor(\pi/4)\times 2^{64}\rfloor(5760+44928) \approx 1.21\cdot 2^{79}$. The same calculation

follows for the overall depth. The maximum number of qubits is simply how many separable blocks of encryption/decryption process times the total number of qubits each block uses. Again, there are two detachable blocks, including the extra 2 128 qubit blocks to store C_0 and S_0. So, the maximum number of qubits should be at max $2 \times 984 + 256 = 2224$ qubits. All these calculations are demonstrated in the following table:

Number of gates		depth		Number of
T	Clifford	T	Overall	qubits
$1.22 \cdot 2^{84}$	$1.59 \cdot 2^{84}$	$1.21 \cdot 2^{79}$	$1.33 \cdot 2^{80}$	2224

Table 4.4: Summarisation of the resource estimates of the attack

Conclusion and Future Direction

Grassl's quantum brute force method, supported by Bogdanov et al.'s biclique construction, has undergone significant refinement, resulting in a fourfold decrease in resource estimations. This breakthrough demonstrates that quantum computing is getting increasingly powerful in cryptography, and it serves as a reminder that we must continue to innovate in order to stay ahead of possible hazards. Quantum algorithms like quantum counting offer more ways to enhance cryptographic tasks, potentially boosting efficiency even more. Additionally, the consideration of quantum mechanisms for precomputation and recomputation holds promise in optimizing resource utilization and overall performance.

In light of these advancements, particularly regarding their implications for AES-128 security, it becomes increasingly apparent that a reliance solely on AES-128 for private key encryption may no longer suffice. The dynamic nature of quantum computing necessitates a more adaptive and diversified approach to encryption techniques. Despite the fact that AES-128 has long been the preferred way of strong encryption, the advent of quantum computing needs a new solution that complements the old one. Developing comprehensive methods that integrate both conventional and quantum-resistant encryption systems will be critical in dealing with security risks as the information landscape becomes more complex.

As for the future direction of this work, one can notice the following:

- As stated earlier, a quantum mechanism for generating precomputation and recomputation, as well as a way of detecting false positives in the solution state, can help in reducing the number of gates in the search algorithm.

- Bogdanov et al. have also suggested a classical independent biclique attack on AES-192 and AES-256 alongside a long biclique with a guessing stage. Implementing them

in a quantum circuit and checking a false positive guess will improve this algorithm further.

- Biclique formations are not just restricted to plaintext to state and ciphertext to state. They can happen in intermediate states as well, as is evident in Roy et al.'s ([15]) paper on FUTURE. One can design a quantum biclique attack as well.
- One can similarly start implementing classical symmetric key ciphers in a quantum circuit, which can help others in estimating the time for quantum attacks.
- ASCON [16] is a family of lightweight block ciphers developed for NIST competition, which claims to have a quantum-resistant classical cipher "ASCON-80pq". One can try to design quantum attacks for these ciphers in particular.
- Since its classical equivalent has been the subject of intense research, it is possible to build a new truncated difference quantum attack.

Bibliography

[1] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[2] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full aes. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 344–371. Springer, 2011.

[3] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *2006 IEEE International Symposium on Information Theory*, pages 1614–1618. IEEE, 2006.

[4] Christof Paar, Jan Pelzl, Christof Paar, and Jan Pelzl. The data encryption standard (des) and alternatives. *Understanding Cryptography: A Textbook for Students and Practitioners*, pages 55–86, 2010.

[5] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (aes), 2001-11-26 2001.

[6] Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov, Ruslan Mordvinov, et al. A new encryption standard of ukraine: The kalyna block cipher. *Cryptology ePrint Archive*, 2015.

[7] SAPV Tharrmashastha, Debajyoti Bera, Arpita Maitra, and Subhamoy Maitra. *Quantum Algorithms for Cryptographically Significant Boolean Functions: An IBMQ Experience*. Springer, 2021.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

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

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

[11] Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE transactions on Information Theory*, 26(4):401–406, 1980.

[12] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 617–630. Springer, 2003.

[13] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings 6*, pages 1–13. Springer, 2000.

[14] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying grover's algorithm to aes: quantum resource estimates. In *International Workshop on Post-Quantum Cryptography*, pages 29–43. Springer, 2016.

[15] Himadry Sekhar Roy, Prakash Dey, Sandip Kumar Mondal, and Avishek Adhikari. Cryptanalysis of full round future with multiple biclique structures. *Peer-to-Peer Networking and Applications*, 17(1):397–409, 2024.

[16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

[17] Mahima Ranjan Adhikari and Avishek Adhikari. *Basic modern algebra with applications*. Springer, 2014.

[18] Subhadeep Banik, Subhamoy Maitra, Santanu Sarkar, and Meltem Sonmez Turan. A chosen iv related key attack on grain-128a. Number 7959. Proceedings of 18th Australasian Conference on Information Security and Privacy, Brisbane, AU, 2013-07-24 00:07:00 2013.

[19] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.

[20] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.

[21] Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. Quantum time/memory/data tradeoff attacks. *Designs, Codes and Cryptography*, pages 1–19, 2023.

[22] K. Sakiyama, Y. Sasaki, and Y. Li. *Security of Block Ciphers: From Algorithm Design to Hardware Implementation*. IEEE Press. Wiley, 2016.

[23] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Transactions on Symmetric Cryptology*, page 71–94, December 2016.

[24] M. Van den Nest. Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond, 2009.

[25] Alexei Petrenko. *Applied Quantum Cryptanalysis*. River Publishers, 2023.

[26] Douglas R Stinson. *Cryptography: theory and practice*. Chapman and Hall/CRC, 2005.

[27] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.

[28] Thomas W Cusick and Pantelimon Stanica. *Cryptographic Boolean functions and applications*. Academic Press, 2017.

[29] N David Mermin. *Quantum computer science: an introduction*. Cambridge University Press, 2007.

[30] Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced sha-0 and sha-1. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 70–89. Springer, 2009.

[31] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on aes-256 variants with up to 10 rounds. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 299–319. Springer, 2010.

[32] Johan Borst, Bart Preneel, and Joos Vandewalle. On the time-memory tradeoff between exhaustive key search and table precomputation. In *Symposium on Information Theory in the Benelux*, pages 111–118. Technische Universiteit Delft, 1998.

[33] Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 534–541, 1991.

[34] Barton Zwiebach. Course on quantum physics i, 2016.

[35] Akinori Hosoyamada and Yu Sasaki. Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In *Cryptographers' Track at the RSA Conference*, pages 198–218. Springer, 2018.

[36] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *International journal of wireless and mobile computing*, 2(1):86–93, 2007.