UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Master's thesis
(Magistrsko delo)

On Applications of Quantum Algorithms in Cryptanalysis of Block Ciphers

(Uporaba kvantnih algoritmov v kriptoanalizi bločnih šifer)

Ime in priimek: *Draženka Višnjić*
Študijski program: *Matematične znanosti, 2. stopnja*
Mentor: *prof. dr. Enes Pasalic*
Somentor: *doc. dr. Samir Hodžić*

Koper, september 2021

Ključna dokumentacijska informacija

Ime in PRIIMEK: Draženka VIŠNJIĆ

Naslov magistrskega dela: Uporaba kvantnih algoritmov v kriptoanalizi bločnih šifer

Kraj: Koper

Leto: 2021

Število listov: 107 Število slik: 22 Število tabel: 3

Število prilog: 1 Število strani prilog: 1

Število referenc: 60

Mentor: prof. dr. Enes Pasalic

Somentor: doc. dr. Samir Hodžić

UDK: 519.712(043.2)

Ključne besede: Kvanti algoritmi, Kriptoanaliza bločnih šifer, Post-kvantna analiza, Simonov algoritem, Groverjov algoritem, Berstein-Vaziranijev algoritem, Kuperbergov algoritem, Feistelova mreža, SPN, FX

Math. Subj. Class. (2020): 03G12, 06E30, 15B34, 81P94, 94A60

Izvleček:

V magistrskem delu je opisana uporaba kvantnih algoritmov v kriptoanalizi bločnih šifer, ki se opirajo na Feistelovo omrežje ali na zamenjalno-permutacijsko strukturo (SPN). V delu analiziramo več dobro poznanih kvantnih algoritmov, med katere spadajo Simonov, Groverjev, Berstein- Vaziranijev ter Kuperbergov algoritem. Preučena je njihova časovna zahtevnost, ki jo primerjamo z zahtevnostjo dotičnih klasičnih algoritmov. Za omenjene algoritme je podanih več aplikacij v kriptoanalizi nekaterih enkripcijskih shem, kot so npr. Even-Mansourjeva konstrukcija, FX konstrukcija ter številne Feistelove šifre. Utemeljeno je, da so kvantni napadi, ki temeljijo na kombinaciji Simonovega in Groverjevega algoritma pogosto uspešni v primeru Feistelovega omrežja, medtem ko so šifre, ki temeljijo na SPN strukturi, dovolj odporne na njih. To še posebej velja za šifre tipa AES. Orodja, ki jih v magistrskem delu uporabimo, slonijo na dobro poznanih razultatih iz linearne algebre in kvantnega računalništva. Ker je post-kvantna kriptoanaliza bločnih šifer šele v povojih, zaključimo, da je potrebno kvantno varnost bločnih šifer ponovno evalvirati.

Key document information

Name and SURNAME: Draženka VIŠNJIĆ
Title of the thesis: On Applications of Quantum Algorithms in Cryptanalysis of Block Ciphers

Place: Koper
Year: 2021

Number of pages: 107 Number of figures: 22 Number of tables: 3
Number of appendices: 1 Number of appendix pages: 1
Number of references: 60
Mentor: Prof. Enes Pasalic, PhD
Co-Mentor: Assist. Prof. Samir Hodžić, PhD

UDC: 519.712(043.2)
Keywords: Quantum algorithms, Simon's, Cryptanalysis of block ciphers, Post-quantum analysis, Simon's algorithm, Grover's algorithm, Bertein-Vazirani's algorithm, Kuperberg's algorithm, Feistel network, SPN, FX
Math. Subj. Class. (2020): 03G12, 06E30, 15B34, 81P94, 94A60
Abstract:
In this thesis, we elaborate applications of quantum algorithms in cryptanalysis of block ciphers based on Feistel network or Substitution-Permutation network (SPN). So far, many significant results have been discovered in this field. In this context, we are recalling and elaborating the widely used quantum algorithms such as: Simon's, Grover's, Berstein- Vazirani's and Kuperberg's algorithms. Also, we discuss their complexities in comparison to the classical counterparts. Then, we elaborate the applications of these algorithms in the cryptanalysis of several general encryption schemes such as Even-Mansour, FX construction, various Feistel-based ciphers, just to name a few. It appears that the quantum attacks (mainly based on the Simon-Grover combination) are efficiently applied to Feistel networks, while the SPN-based ciphers still possess sufficient resistance (especially AES-like ciphers). The methodology that we use is based on the well-known results in linear algebra and quantum computation. As the post-quantum development of the field of cryptanalysis of block ciphers is still in its early phase, we conclude that the quantum security of block ciphers has to be re-evaluated.

List of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

i.e.	that is
e.g.	for example

Acknowledgments

First, I would like to thank my mentor professor Enes Pasalic for introducing me to the "world of cryptography." Special thanks goes to my co-supervisor, doc. dr. Samir Hodžić, for his time, patience, and all reviews of my thesis; without him, this thesis would not have been possible. I am especially grateful to my love, family, and friends for their unconditional support.

Najveću zahvalnost dugujem najposebnijoj i najvažnijoj osobi u mom životu, mojoj bezuslovnoj podršci, potpori i najvećoj životnoj inspiraciji, bez koje ja ne bih danas bila to što jesam, jednoj i jedinoj mojoj majci.

Draženka Višnjić

1 INTRODUCTION

Throughout history, people always had a need to group, thus creating tribes, empires, kingdoms, states, and hence the need to protect information from external enemies or even from members of their group. So, it was natural to create a science, which keeps the messages secure, which is today known as cryptography (kryptós-hidden, graphein-to write). The beginning of its development has been initiated by the discovering of letters. The first known evidences of this science were found in a wall of the tomb from the Old Kingdom of Egypt circa 1900 BC. At first, before the modern era, which is a turning point in science and technology, the main goal of cryptography was to secure confidential documents, to ensure the security of communication during wars, political competition and religious revolutions. However, the modern cryptography is used to protect our bank accounts, databases, computers, e-mails, etc.

On the other hand, in order to improve cryptographic schemes or test their security, the field of *cryptanalysis* has been developing over time as well, which is the science of analyzing and breaking cryptographic systems. These two sciences, cryptography and cryptanalysis, are combined into one common field called *cryptology* that studies the methods of providing/breaking the secrecy in a communication between two parties. This is one of the most important and significant sciences today that has changed its shape and form throughout history, but its main goal to provide secrecy has remained the same. In order to understand better the modern cryptography that we know today (in context of the use of secret keys), we now briefly describe its two main types, namely *public-key* and *symmetric key cryptography* (see [54] for more details).

Suppose that Alice and Bob are communicating over an insecure channel (e.g. telephone line, computer network, or radio). The main goal of cryptography is to disable an adversary, usually referred to as Eve/Oscar, to see what is being sent. The message that Alice and Bob want to exchange is called *plaintext* (e.g. text in English, numbers). To disable the adversary, Alice chooses a secret key K (i.e. an encryption key) and encrypts the plaintext P to obtain an encrypted message called *ciphertext* C. Then, Alice sends ciphertext to Bob, which uses a decryption key K' to obtain the original message (K and K' are not necessarily the same in general). Here, the transformation of the plaintext into ciphertext is called the *encryption* process, and the reverse process is called the *decryption*. This is illustrated in Figure 1.

Depending on the type of encryption and decryption keys, there are two types of cryptography. If the sender and receiver use the same keys, the secret key to encrypt and decrypt the message (i.e. $K = K'$), then it regards the *symmetric-key cryptography*. On the other hand, if $K \neq K'$, in which case two separate keys are required: the public key for encryption and the private one for decryption of the message, then it regards *asymmetric-key cryptography*. Apart from the traditional modern cryptogra-

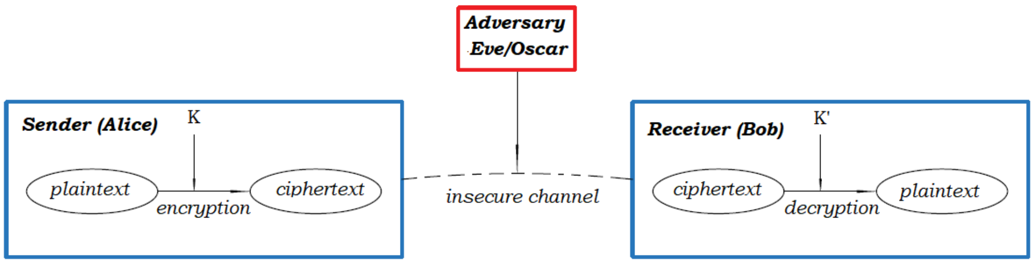

Figure 1: The communication over an insecure channel.

phy, there has been a lot of discussion recently concerning post-quantum cryptography, which is a result of the development of quantum computers. An initial idea to develop quantum computers belongs to the physicist Richard Feyman (1982) [23], whose work influenced the derivation of various quantum algorithms. The first construction of quantum computers appeared in the last two decades with the tendency to achieve the quantum supremacy, i.e. to have a programmable quantum device capable of solving certain problems faster than the most powerful classical computer in any feasible amount of time.

In general, the term algorithm refers to a series of steps that we must perform to solve a given problem. However, a quantum algorithm is an algorithm such that at least one of the steps is based on quantum states such as superposition and/or entanglement. The algorithms are usually described by a quantum circuit, which represents a model of quantum computation where the steps of solving a given problem are quantum gates applied to one or more qubits. A detailed description of these terms will be given later on.

In certain cases, quantum algorithms are much faster and efficient than classical algorithms. However, the following question arises: Will the privacy of our data be compromised in the (near) future by quantum computers, which are predicted to be a part of our daily lives? The application of quantum algorithms in the last two decades shows that the classical public-key encryption schemes can be completely broken. While the symmetric-key algorithms do not provide the same level of security as in the classical environment. For instance, in 1994, Peter Shor [52] described a polynomial-time algorithm for solving the integer factorization and discrete logarithm problems, which means that Shor's algorithm can completely break the RSA-like cryptosystems. On the other hand, the application of Grover's algorithm [25] to symmetric-key encryption schemes imposes the requirement that a secret master key has to be doubled in size. Later on, the application of other quantum algorithms, such as Simon's [53], Bernstein-Vazirani's [6], Kuperberg's [36] and their combinations, shows that our understanding of the security of symmetric-key schemes is not on a proper level yet.

Before we describe the main goals of this thesis, we briefly recall the notation related to symmetric-key encryption schemes. As they are traditionally divided in two main representatives, namely stream ciphers and block ciphers, in this thesis we will be considering only block ciphers. An n-bit block cipher is a function $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ such that for a fixed key $K \in \mathcal{K}$ ($\mathcal{K}$ denotes the *keyspace*) the *encryption function* $E(P,K) = E_K(P)$ is a permutation, which takes as inputs a key K of a length k and a message block P (*plaintext*) of a length n, and produces an n-bit output C called *chipertext.* The inverse mapping is the decryption function $D : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ defined as $D(C,K) = E_K^{-1}(C) = D_K(C)$, such that for each $K \in \mathcal{K}$, $D_K(E_K(P)) = P$.

The most common design approachs of block ciphers today are iterated product block ciphers with the structure based on either Feistel Network (FN) or Substitution Permutation Network (SPN). The encryption of iterated product block ciphers is performed by applying the same round function (certain amount of times) which uses the so-called round keys, which are derived from a secret master key by a key-scheduling algorithm.

In this thesis, we elaborate applications of quantum algorithms (mentioned earlier) in cryptoanalysis of block ciphers based on FN or SPN structure. So far, many significant results have been discovered in this field. For instance, although Simon's algorithm was described in 1994, its first application in cryptanalysis of block ciphers was demonstrated in 2010 [37] when it was shown that security of the 3-round Feistel cipher can be completely broken. Later, it was shown that the so-called Even-Mansour construction is not secure [38] as well. Furthermore, Leander and May [39] firstly introduced the combination of Simon's [53] and Grover's algorithms [25], which were applied to the FX-construction [34]. They showed that the key whitening procedure of increasing the key space is not efficient as it is in the classical environment. Also, the applications of Berstein-Vazirani's [6] and Kuperberg's [36] algorithms are given in the context of constructing quantum distinguishers based on period finding (for binary and non-binary block ciphers respectively). Overall, the post-quantum development of the field of cryptanalysis of block ciphers is still in its early phase.

2 PRELIMINARIES

In this section, we give an overview of definitions important for understanding the results discussed in the subsequent sections. We start with the definition of Boolean functions, which are utilized in the design of almost all symmetric-key encryption schemes.

Definition 2.1. A *Boolean function* f in n variables is a map $\{0,1\}^n \to \{0,1\}$.

Some further properties of Boolean functions are given in Chapter 5. As linear algebra plays an important role in the description of quantum algorithms, in what follows we recall the main definitions and properties.

2.1 GROUPS AND VECTOR SPACES

We start with the well-known results related to groups and vector spaces. If not stated otherwise, the definitions have been taken from [27].

Definition 2.2. The *group* is a pair $(G, \cdot)$, where $\cdot : G \times G \to G$ is a binary operation with the following properties:

1. *Associativity:* $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, $\forall x, y, z \in G$.

2. *Identity element:* $\exists! e \in G : e \cdot x = x \cdot e = x, \forall x \in G$.

3. *Inverse element:* $\forall x \in G$, $\exists! x^{-1} : x \cdot x^{-1} = x^{-1} \cdot x = e$.

A group in which the group operation is commutative is called *a commutative* or *Abelian group.* The definition of the vector space is given as follows.

Definition 2.3. Let $V(+, \cdot)$ be an Abelian group and $(F, +, \cdot)$ be a field. We say that V is *a vector space* over F if there exist function $F \times V \to V$, such that for every $\alpha, \beta \in F$ and $\mathbf{u}, \mathbf{v} \in V$ the following holds:

1. $\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \beta\mathbf{v}$,

2. $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$,

3. $(\alpha\beta)\mathbf{u} = \alpha(\beta\mathbf{u})$,

4. $e\mathbf{u} = \mathbf{u}$,

where $(\alpha, \mathbf{u}) = \alpha\mathbf{u}$.

The elements of the set V are called *vectors* and are usually denoted as $\mathbf{v}$ or $\overrightarrow{v}$. However, in quantum computing it is common to use the so-called *Dirac notation (bra-ket)*, which will be introduced in the next subsection. In most applications of quantum computation, finite vector spaces over the field of complex numbers $\mathbb{C}$ is used, and it is denoted by $\mathbb{C}^n$. Several definitions related to vectors spaces are given as follows.

Definition 2.4. In a vector space V, a sequence of vectors $\mathbf{v}_1, \mathbf{v}_2, \dots \mathbf{v}_n$ is *linearly dependent*, if there exist scalars $\alpha_1, \alpha_2, \dots, \alpha_n \in \mathbb{C}$ such that at least one scalar is non-zero and

$$\alpha_1\mathbf{v_1} + \dots + \alpha_n\mathbf{v_n} = \mathbf{0}.$$

Otherwise, we say that a sequence of vectors is *linearly independent.*

Definition 2.5. A *basis* for a vector space V is a sequence of vectors which is linearly independent and spans V.

Definition 2.6. We say that a vector space V is *finite-dimensional* if its basis contains finitely many vectors.

Definition 2.7. The *dimension* of a vector space V, denoted by $dim(V)$, is the number of vectors in its bases.

Theorem 2.8. *In a vector space V, a sequence of vectors $\boldsymbol{v_1}, \dots, \boldsymbol{v_n}$ is a basis if and only if any vector $x \in \mathbb{C}^n$ can be written in the form*

$$x = \sum_{i=1}^{n} \alpha_i \boldsymbol{v_i}, \quad \alpha_1, \dots \alpha_n \in \mathbb{C}^n.$$

We note that the definitions, theorems and other results given in the subsequent subsections of this chapter are taken from [46, 57].

2.2 THE DIRAC NOTATION

In the previous subsection, vectors were denoted as $\boldsymbol{v}$. However, as mentioned before, in quantum computing it is common to use Dirac notation (bra-ket) introduced by the English physicist Paul Dirac. The notation uses the angle brackets,"$\langle$" and "$\rangle$", and a vertical bar "$|$".

Let $\boldsymbol{v}$ be an arbitrary vector in $\mathbb{C}^n$, then

$$\boldsymbol{v} = |v\rangle = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix},$$

such that all entries v_i for $i = 1, \dots, n$ are complex numbers, i.e. $v_i = x_i + y_i i$, where real numbers x_i and y_i are real and imaginary part of v_i, respectively. Its dual vector is denoted as follows

$$\langle v| = (\bar{v_1}, \dots, \bar{v_n}),$$

where $\bar{v_i}$ is a conjugate of the complex number v_i, for $i \in \{1, \dots, n\}$, respectively. Then "$|\rangle$" is called a *ket* and "$\langle|$" is called a *bra*. Using this notation we can represent vectors of the computational basis of $\mathbb{C}^n$ in the following way:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \ \dots, |n-1\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}. \tag{2.1}$$

Example 2.9. Let us consider 2-dimensional vector space $\mathbb{C}^2$. From (2.1) we know that its computational basis is $B = \{|0\rangle, |1\rangle\}$, where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

For a better understanding of quantum algorithms, in their mathematical description, an inner product is used, which is defined as follows.

Definition 2.10. Let V be vector space over field F. Then a function $\langle \cdot | \cdot \rangle : V \times V \to F$ is called an *inner product* that satisfies the following conditions for all vectors $|x\rangle, |y\rangle, |z\rangle \in V$ and all scalars $\alpha \in F$

1. $\langle x|y\rangle = \overline{\langle y|x\rangle}$,
2. $\langle x + y|z\rangle = \langle x|z\rangle + \langle y|z\rangle$,
3. $\langle \alpha x|y\rangle = \overline{\alpha}\langle x|y\rangle$,
4. $\langle x|x\rangle \geq 0$,
5. $\langle x|x\rangle = 0 \iff |x\rangle = 0$.

Example 2.11. In the vector space $\mathbb{C}^n$, a function defined by

$$\langle x|y\rangle = \sum_{i=1}^{n} \overline{x_i} y_i,$$

where $|x\rangle = (x_1, \dots, x_n)^\top$, $|y\rangle = (y_1, \dots, y_n)^\top \in \mathbb{C}^n$ is called an inner product.

Definition 2.12. A vector space over the field of complex numbers with an inner product is called *a unitary vector space.*

Definition 2.13. In a unitary vector space V, the function $\|\cdot\| : V \to \mathbb{R}$ defined as

$$\||x\rangle\| = \sqrt{\langle x|x\rangle}$$

is called a *norm* on V. A non-negative real number $\||x\rangle\|$ is called a *norm* of vector $|x\rangle$.

Definition 2.14. In a unitary vector space V we say that vectors $|x\rangle, |y\rangle \in V$ are orthogonal if $\langle x|y\rangle = 0$.

Definition 2.15. A basis $\{|a\rangle, |b\rangle\}$ of $\mathbb{C}^2$ is *orthonormal* if

1. $\langle a|b\rangle = 0$,
2. $\langle a|a\rangle = \langle b|b\rangle = 1$.

Example 2.16. Let us consider a basis $\mathcal{H} = \{|+\rangle, |-\rangle\}$ of $\mathbb{C}^2$, where

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \text{ and } |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.2}$$

The basis introduced above is called a *Hadamard basis* and it is an example of an orthonormal basis.

Example 2.17. An example of another orthonormal basis of $\mathbb{C}^2$ is the Nega-Hadamard basis: $\mathcal{N} = \{|i\rangle, |-i\rangle\}$, where

$$|i\rangle = \frac{|0\rangle + i|1\rangle}{\sqrt{2}} \text{ and } |-i\rangle = \frac{|0\rangle - i|1\rangle}{\sqrt{2}} \tag{2.3}$$

We note that the two bases introduced in Examples 2.15 and 2.16 are important in the construction of quantum algorithms.

2.3 QUBITS AND QUANTUM GATES

A *bit* is a fundamental unit of classical information processing. Similarly, a *qubit* is a fundamental unit of quantum information processing and it is defined as follows.

Definition 2.18. The *qubit* is a *unit vector* in $\mathbb{C}^2$. Each qubit can be represented as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \tag{2.4}$$

where $\alpha, \beta \in \mathbb{C}$ are called *amplitudes* of state $|\psi\rangle$ and $|\alpha|^2 + |\beta|^2 = 1$. If $\alpha \neq 0$ and $\beta \neq 0$ than qubit is in the *superposition* of states $|0\rangle$ and $|1\rangle$.

The *measurement* plays a central role in quantum computations since, we have to be able to extract information out of the computational system and it is associated to an orthonormal basis, say $\{|\psi_1\rangle, |\psi_2\rangle\}$. Thus, if we want to measure (2.4) the qubit is forced into one of the following two states, either $|\psi_1\rangle$ or $|\psi_2\rangle$. Based on the Born rule described in [60], the probability to get $|\psi_1\rangle$ equals

$$|\langle\psi_1|\psi\rangle|^2. \tag{2.5}$$

Similarly, the probability to get $|\psi_2\rangle$ equals

$$|\langle\psi_2|\psi\rangle|^2. \tag{2.6}$$

This measurement is called a *single-qubit* measurement.

Example 2.19. Let us consider the single-qubit state $|\psi\rangle = \frac{i}{\sqrt{3}}|0\rangle + \frac{\sqrt{2}}{\sqrt{3}}|1\rangle$ and the measurement basis $\{|0\rangle, |1\rangle\}$. By (2.5) and (2.6) we are able to determine their probabilities. Since we have

$$\langle 0|(\frac{i}{\sqrt{3}}|0\rangle + \frac{\sqrt{2}}{\sqrt{3}}|1\rangle)\rangle = \frac{i}{\sqrt{3}}\langle 0|0\rangle + \frac{\sqrt{2}}{\sqrt{3}}\langle 0|1\rangle = \frac{i}{\sqrt{3}}$$

the probability to obtain $|0\rangle$ after the measurement equals

$$|\langle 0|\psi\rangle|^2 = \left|\frac{i}{\sqrt{3}}\right|^2 = \frac{1}{3}.$$

Equivalently, the probability to obtain $|1\rangle$ equals

$$|\langle 1|\psi\rangle|^2 = \left|\frac{\sqrt{2}}{\sqrt{3}}\right|^2 = \frac{2}{3}.$$

If we want to transform one qubit state into another we use unitary transformations.

Definition 2.20. A linear transformation U in a vector space V is said to be *unitary* if

$$U^\dagger U = UU^\dagger = I$$

where I is a identity linear transformation and $U^\dagger$ denotes the conjugate transpose of U.

A set of linear operations that are of fundamental importance in quantum computations is known as *Pauli transformations*, which are defined as follows.

Definition 2.21. With respect to the computational basis the matrix of the Pauli transformations I, X, Y and Z are given by

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.7}$$

All these transformations can be written using the Dirac notation as follows:

- $I = |0\rangle\langle 0| + |1\rangle\langle 1|$,
- $X = |1\rangle\langle 0| + |0\rangle\langle 1|$,
- $Y = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$,
- $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$.

These transformations act to the computational basis states as follows:

$$\begin{aligned} I|0\rangle &= |0\rangle, & I|1\rangle &= |1\rangle, \\ X|0\rangle &= |1\rangle, & X|1\rangle &= |0\rangle. \end{aligned}$$

Since X is similar to classical NOT operation it is sometimes known as the NOT operator or *bit flip*.

$$\begin{aligned} Y|0\rangle &= -i|1\rangle, & Y|1\rangle &= i|0\rangle, \\ Z|0\rangle &= |0\rangle, & Z|1\rangle &= -|1\rangle. \end{aligned}$$

Let us introduce the fundamental quantum operation, called the Hadamard transformation, which is represented by a Hadamard matrix.

Definition 2.22. A matrix $H_{n\times n}$ which has all entries ± 1 and satisfies $HH^\top = I$ is called a *Hadamard matrix.*

Example 2.23. For a single qubit we use 2×2 matrixes, therefore the Hadamard matrix $H_{2\times 2}$. It is given by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.8}$$

Remark 2.24. Through the thesis the Hadamard matrix 2×2 is used more frequently.

The Hadamard transform acts on the computational basis states in the following way:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \text{ and } H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle. \tag{2.9}$$

In Dirac notation (2.8) can be represented by

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

The Hadamard and Pauli transformations represented in (2.8) and (2.7), respectively are the basic operations for one single-qubit. Some of the rest quantum transformations are presented in Table 1. These operators are simply called *quantum gates.*

Now we describe a multi-qubit state using tensor product. In this context, a *quantum register* is a set of qubits treated as a composite system. We consider two single qubit states $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\psi_2\rangle = \alpha'|0\rangle + \beta'|1\rangle$, respectively. The combination of two state is given by

$$\begin{aligned}|\psi_1\rangle \otimes |\psi_2\rangle &= (\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha'|0\rangle + \beta'|1\rangle)\\ &= \alpha\alpha'|0\rangle|0\rangle + \alpha\beta'|0\rangle|1\rangle + \beta\alpha'|1\rangle|0\rangle + \beta\beta'|1\rangle|1\rangle\\ &= \alpha\alpha'|00\rangle + \alpha\beta'|01\rangle + \beta\alpha'|10\rangle + \beta\beta'|11\rangle,\end{aligned}$$

where $|\alpha\alpha'|^2 + |\alpha\beta'|^2 + |\beta\alpha'|^2 + |\beta\beta'|^2 = 1$. For simplicity we often write $|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = |\psi_1\psi_2\rangle$. In general state of n-qubit register is

$$|\psi\rangle = \sum_{i\in\{0,1\}^2} a_i|i\rangle, \tag{2.10}$$

where $\sum_{i\in\{0,1\}^2} |a_i|^2 = 1$. Consequently, all transformations to the n-qubit states are $2^n \times 2^n$ matrices, that are obtained using Kronecker product.

Definition 2.25. Let A be a $m \times n$ matrix and B a $p \times q$ matrix. Then,

$$C = A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}. \tag{2.11}$$

The resulting matrix C is $mp \times nq$ matrix.

Let us consider gate $H^{\otimes n} = \underbrace{H \otimes \cdots \otimes H}_{n \text{ times}}$ that preforms a Hadamard transform on the register of n-bits.

Proposition 2.26. *Let $|0\rangle^{\otimes n} = |0\rangle \otimes \cdots \otimes |0\rangle$ denote initial state on n qubits. Then*

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{\in\{0,1\}^n} |x\rangle. \tag{2.12}$$

Proof. We prove the above proposition by induction on n.
Base case: For $n = 1$, it follows from 2.9 that the property holds. For $n = 2$, we have that:

$$H \otimes H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 \end{pmatrix},$$

and

$$|0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Therefore,

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

Hence, $H^{\otimes 2}|00\rangle = \frac{1}{\sqrt{2^2}}\sum_{x\in 0,1^2}|x\rangle$ and (2.12) holds for $n = 2$.

Induction step: Suppose now (2.12) is true for $n-1$, i.e.

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

Let us consider $|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = H^{\otimes(n-1)}|0\rangle^{\otimes(n-1)}H|0\rangle$. By induction hypothesis and (2.9) we have

$$|\psi\rangle = \frac{1}{\sqrt{2^{n-1}}}\sum_{x\in\{0,1\}^{n-1}}|x\rangle \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2^n}}\sum_{\in\{0,1\}^n}|x\rangle.$$

□

Proposition 2.27. *[59] Let $|x\rangle = |x_1 \ldots x_n\rangle$ denote a computational basis on n qubit, i.e $x \in \{0,1\}^n$, where $x_i \in \{0,1\}$. Then*

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle. \tag{2.13}$$

Here, the dot product $x \cdot y$ denotes the dot product modulo two.

Notice that both Hadamard and Pauli quantum gates are gates for a single qubit register. The quantum gate that is often used in the construction of quantum algorithms is the Controlled Not gate (CNOT). Its matrix form, as well as other important quantum gates, are given in Table 1. Quantum gates are the building blocks of the quantum circuits, which we describe in the following subsection.

Table 1: The matrix form of some of the more important quantum gates.

Quantum Gate	Matrix Form	Quantum Gate	Matrix Form
Hadamard	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	CNOT	$\begin{pmatrix}1 & 0 & 00 & \\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$
Pauli-X	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	Swap	$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$
Pauli-Y	$\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$	CZ	$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{pmatrix}$
Pauli-Z	$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$	CPHASE	$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & i\end{pmatrix}$
Phase (S)	$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$	Toffoli	$\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{pmatrix}$
$\frac{\pi}{8}$ (T)	$\begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix}$	CSWAP	$\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$

2.4 THE QUANTUM CIRCUIT

Programs that run on quantum computers can be described through quantum circuits, where quantum gates act on qubits. The following conventions are used for drawing circuits:

- Each line in the circuit represents a *wire* in the quantum circuits.
- Horizontal single lines represent the qubits and may be labeled.
- Double lines are classical bits.
- The gates on these qubits are drawn on the wire.
- The circuit is to be read from the left to the right.
- Control qubits are represented by solid circles.

- Every new state maybe separated with vertical lines.

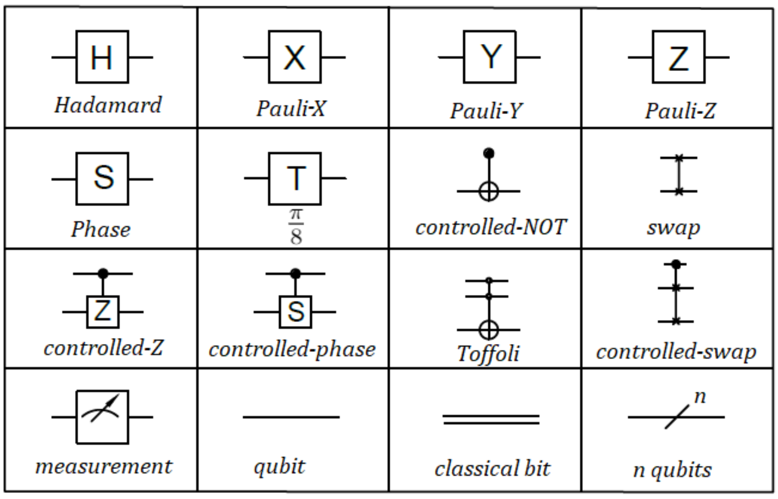

Figure 2: Symbols in quantum circuits.

Let us consider the given quantum circuit. In Figure 3 we see that

1. The initial state $|\psi_0\rangle = |x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle$.
2. The next state $|\psi_1\rangle$ is obtained by acting $H \otimes I \otimes X \otimes I$ on initial state $|\psi_0\rangle$.
3. In this step we call a black-box/oracle function which is a unitary operator i.e. $U_f|\psi_1\rangle = |\psi_2\rangle$.
4. Finally, to obtain the state $|\psi_3\rangle$ we perform the following transformation $(CNOT \otimes I \otimes T)|\psi_2\rangle$.

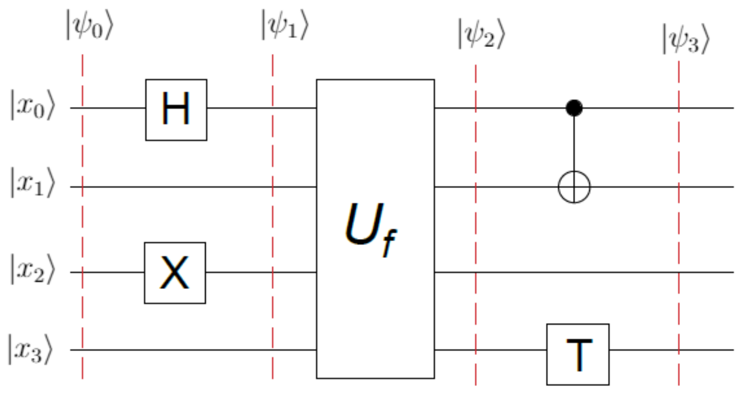

Figure 3: An example of a quantum circuit without measurement.

One of the most important quantum operations, the quantum Fourier transform, which has also found application in the construction of Kuperberg's algorithm (described later on), is introduced in the following subsection.

2.5 THE QUANTUM FOURIER TRANSFORM

Definition 2.28. The *classical Fourier transform* is a mapping that takes an input vector $(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{C}^n$ and maps it to the output vector $(y_0, y_1, \ldots, y_{n-1}) \in \mathbb{C}^n$ in the following way:

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_k e^{\frac{-2\pi ijk}{n}}, \quad k = 0, 1, \ldots, n-1. \tag{2.14}$$

Definition 2.29. The *quantum Fourier transform* (QFT) on an orthonormal basis $|0\rangle, |1\rangle, \ldots, |n-1\rangle$ is defined as a linear operator with the following action on the basis states:

$$|k\rangle \to \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} e^{\frac{2\pi ijk}{n}} \tag{2.15}$$

Similarly, applying QFT on an arbitrary state, we can write

$$\sum_{j=0}^{n-1} x_j |j\rangle \to \sum_{k=0}^{n-1} y_k |k\rangle,$$

where the amplitudes y_k are classical Fourier transformation of the amplitudes x_j. The quantum Fourier transform can be represented by the unitary $n \times n$ matrix as follows:

$$F_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \ldots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \ldots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \ldots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \ldots & \omega^{(n-1)(n-1)} \end{pmatrix}, \tag{2.16}$$

where $\omega = e^{\frac{2\pi i}{n}}$ is an *n-th* root of unity.

Example 2.30. Let us consider matrix form of QFT for $n = 2$, then $\omega = e^{\frac{2\pi i}{2}} = e^{\pi i} = -1$. From Equation 2.16 we have

$$F_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

which is the Hadamard transform introduced in (2.8).

When constructing algorithms, it is important to consider their efficiency. A special branch of science, Complex Theory studies this. For our needs, we introduce the following two definitions, which are taken from [18].

Definition 2.31. For a given function $g(n)$ let us introduce the following set:

$$\Theta(g(n)) = \{f \mid \exists\, c_1, c_2 > 0 \wedge n_0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n),\ \forall n \geq n_0\}.$$

We say that $g(n)$ is an *asymptotically* tight bound for $f(n)$.

Definition 2.32. For a given function $g(n)$ let us introduce the following set:

$$O(g(n)) = \{f \mid \exists\, c > 0 \wedge n_0 : 0 \leq f(n) \leq c_g(n),\ \forall n \geq n_0\}.$$

We use O-notation to give an *upper bound* for the functions.

3 SIMON'S ALGORITHM

The idea of quantum algorithms arose from the need to solve some more complex problems much faster than classical algorithms do. One such algorithm is Simon's algorithm described by Daniel R. Simon in 1994 [53]. A very simple quantum algorithm was derived, being exponentially faster than any known classical algorithm, which solves the so-called Simon's problem. In this section we define this problem and collect some of the most important applications of Simon's algorithm. We note that Simon's algorithm is one of the main tools used in cryptanalysis of block ciphers today.

Problem 3.1. [53] *(Simon's problem) Given a function $f : \{0,1\}^n \to \{0,1\}^m$ $(m \geq n)$ with the promise that there exists a secret string $s \in \{0,1\}^n$ such that $\forall x \neq y$*

$$f(x) = f(y) \Leftrightarrow x \oplus y \in \{0^n, s\}, \tag{3.1}$$

where $\oplus$ denotes bitwise addition module two. The goal is to find s.

If $x \oplus y = 0^n$, then $x = y$, which implies that f is injection (one to one) function.

Remark 3.2. Through the thesis, for the simplicity, we will assume that $n = m$.

Example 3.3. Let us consider a function $f : \{0,1\}^3 \to \{0,1\}^3$ defined by the following table of values:

x	$f(x)$
000	101
001	010
010	000
100	000
011	110
101	110
110	101
111	010

From the table we see that $f(000) = f(110), f(001) = f(111), f(010) = f(100), f(011) = f(101)$ Since $000 \oplus 110 = 110$ it implies that the secret string $s = 110$. Similarly for the remaining cases, by a simple calculation, we can conclude that the secret string s is equal to 110.

Simon's algorithm: The best classical algorithm can solve Simon's problem in time complexity $O(2^{\frac{n}{2}})$. However, with the following quantum algorithm it is possible to solve this problem with quantum complexity $O(n)$. The following description of Simon's algorithm is taken from [50, 59].

1. **Input (2n-qubits):** Prepare the initial state with two $n-$bit registers:

$$|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |0\rangle^{\otimes n},$$

where $\otimes$ is the tensor product.

2. **Apply a Hadamard transform:** In this step the input will be transformed by applying the Hadamard transform $H^{\otimes n}$ to the first register.

$$\begin{aligned}|\psi_1\rangle &= \underbrace{(H^{\otimes n}|0\rangle^{\otimes n})}_{(2.12)} \otimes |0\rangle^{\otimes n}\\ &= \Big(\sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}}|x\rangle\Big) \otimes |0\rangle^{\otimes n}\\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |0\rangle^{\otimes n}.\end{aligned}$$

3. **Apply the oracle or black-box $\mathcal{O}_f$:** Here we call the oracle $\mathcal{O}_f$, that is implemented as a unitary operation that performs the transformation $\mathcal{O}_f(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |f(x) \oplus y\rangle$. Applying it on $|\psi_2\rangle$ we obtain

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |f(x)\rangle.$$

4. **Measure the second register:** From (3.1) we have $f(x) = f(y)$ if and only if $x \oplus y \in \{0^n, s\}$. Therefore if $x \oplus y = 0^n$ it implies $x = y$, and thus f is injection, i.e. each value of x corresponds to a unique value $f(x)$. Therefore, since $\{0,1\}^n$ has 2^n elements, every $x \in \{0,1\}^n$ has equal probability $\frac{1}{2^n}$. On the other hand, if $x \oplus y = s \neq 0^n$ then $x = y \oplus s$. Hence, there are two possible values of x: $x = y$ or $x = y \oplus s$. After measuring the second register, first register is reduced to an equal superposition of those two values:

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|y\rangle + |y \oplus s\rangle).$$

Since, there will be no more operations on the second register, further calculations will be performed only on the first register.

5. **Apply a Hadamard transform to the first register:** Now we apply the Hadamard transform $H^{\otimes n}$ on the first register of state $|\psi_3\rangle$. Using Proposition

(2.27), we obtain

$$\begin{aligned}
|\psi_4\rangle = H^{\otimes n}|\psi_3\rangle &= \frac{1}{\sqrt{2}} H^{\otimes n}(|y\rangle + |y \oplus s\rangle) \\
&= \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2^n}} \big(\sum_{z\in\{0,1\}} (-1)^{y\cdot z}|z\rangle + \sum_{z\in\{0,1\}^n} (-1)^{(y\oplus s)\cdot z}|z\rangle \big) \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} ((-1)^{y\cdot z} + (-1)^{(y\oplus s)\cdot z}|z\rangle) \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} ((-1)^{y\cdot z} + (-1)^{y\cdot z \oplus s\cdot z}|z\rangle) \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} (-1)^{y\cdot z}(1 + (-1)^{s\cdot z})|z\rangle.
\end{aligned}$$

6. **Measurement:** In the case when $s = 0^n$, f is an injective function and $s \cdot z = 0$. Thus the $x \in \{0,1\}^n$ will be produced with uniform distribution.
Suppose now $s \neq 0^n$, then $s \cdot z = 1$ or $s \cdot z = 0$.
If $s \cdot z = 1$, then $(-1)^{y\cdot z}(1 + (-1)^{s\cdot z} = (-1)^{y\cdot z}(1 + (-1)^1) = 0$. Hence, $p_z = 0$, so such a z will never be measured.
If $s \cdot z = 0$, we have $(-1)^{y\cdot z}(1 + (-1)^{s\cdot z} = (-1)^{y\cdot z}(1 + (-1)^0) = 2(-1)^{y\cdot z}$. In that case $|\psi_4\rangle$ becomes

$$\begin{aligned}
|\psi_4\rangle &= \frac{2}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} (-1)^{y\cdot z}|z\rangle \\
&= \frac{1}{\sqrt{2^{n-1}}} \sum_{z\in\{0,1\}^n} (-1)^{y\cdot z}|z\rangle.
\end{aligned}$$

Since $|(-1)| = 1$, the probability of observing any of $z \in \{0,1\}^n$ such that $s \cdot z = 0$ is equal to :

$$p_z = \Big|\frac{1}{\sqrt{2^{n-1}}}\Big|^2 = \frac{1}{2^{n-1}}.$$

So, in summary if $s \neq 0^n$, we have the following probabilities:

$$p_z = \begin{cases} \frac{1}{2^{n-1}}, & \text{if } s \cdot z = 0, \\ 0, & \text{if } s \cdot z = 1. \end{cases}$$

7. **Classical post-procesing:** In the previous part we saw that measurement results in some string $z \in \{0,1\}^n$ that satisfies $s \cdot z = 0$. This is enough information to determinate s. By running this quantum subroutine several times, say $n-1$ times, we get $n-1$ strings: $z_i \in \{0,1\}^n$, $i \in \{1, n-1\}$. This gives a system of $n-1$ equations, with n unknowns to find s:

$$\begin{aligned}
z_1 \cdot s &= z_{11}s_1 + z_{12}s_2 + \dots z_{1n}s_n = 0, \\
z_2 \cdot s &= z_{21}s_1 + z_{22}s_2 + \dots z_{2n}s_n = 0, \\
&\vdots \\
z_{n-1} \cdot s &= z_{(n-1)1}s_1 + z_{(n-1)2}s_2 + \dots z_{(n-1)n}s_n = 0.
\end{aligned}$$

In order to solve given system z_i must be linearly independent. Hence, the probability of finding strings is $p_i = 1 - \frac{2^{i-1}}{2^{n-1}}$, for $1 < i < n-1$. Therefore, the probability that z_i are linearly independent is

$$\prod_{i=1}^{n-1}(1 - \frac{2^{i-1}}{2^{n-1}}) = \prod_{i=1}^{n-1} 1 - \frac{1}{2^{n-i}} = \prod_{i=1}^{n-1} 1 - \frac{1}{2^i} \geq 0.288 \geq \frac{1}{4}.$$

If $z_1, z_2, .., z_{n-1} \in \{0,1\}^n$ are linearly independent, we can solve the system and get some $s' \neq 0^n$, test whether is $f(0^n) = f(s')$ or not. If this is true , we know that $s = s'$ and problem has been solved. If it is not true, it must be that $s = 0^n$. From the obtained probability, we conclude that s can be determined by repeating Simon's algorithm no more than $4n$ times. Therefore, complexity of this algorithm is $O(n)$.

The following figure illustrates Simon's algorithm.

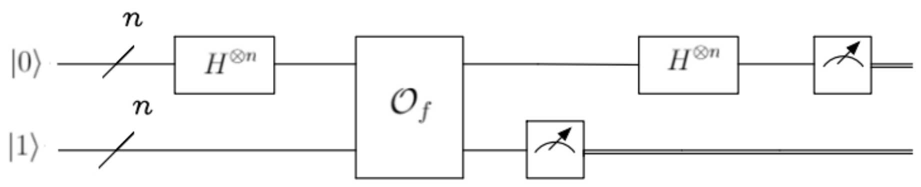

Figure 4: The quantum circuit of Simon's algorithm.

3.1 APLICATIONS OF SIMON'S ALGORITHM

3.1.1 Simon's algorithm and three-round Feistel Construction

Although Simon's algorithm was described in 1994, its first application in cryptanalysis of block ciphers was demonstrated in 2010 by H. Kuwakado and M. Morii, where it was shown that one can construct an efficient quantum distinguisher for 3-round Feistel Network [37]. In order to recall their construction, we firstly provide the definition of a block cipher based on the Feistel construction.

Definition 3.4. [42] A *Feistel cipher* is a symmetric structure that maps a $2n$-bit plaintext $L_0||R_0$, where $L_0, R_0 \in \{0,1\}^n$ and $||$ is the concatenation operator, to a $2n$-bit ciphertext $L_r||R_r$, through an r-round process where $r \geq 1$. For $i \leq 1 \leq r$, $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$, where each K_i is derived from the cipher key K and F is the round function.

In [37] authors considered a 3-round Feistel cipher (Figure 5) with internal permutations (FP) P_1, P_2, P_3 on $\{0,1\}^n$ (assumed to be publicly known). Assume $p = a||b$ be plaintext, then using the previous scheme one can obtain the ciphertext $c = p||q$ as follows:

- **Round 1:** $FP(p) = FP(a||b) = b||a \oplus P_1(b) = c_1$
- **Round 2:** $FP(c_1) = a \oplus P_1(b)||b \oplus P_2(a \oplus P_1(b)) = c_2$
- **Round 3:** $FP(c_2) = b \oplus P_2(a \oplus P_1(b))||a \oplus P_1(b) \oplus P_3(b \oplus P_2(a \oplus P_1(b))) = p||q.$

Now we define a function F which will be utilised in construction of a distinguisher later on.

Definition 3.5. Let E be $3-$round Feistel cipher with internal permutations P_i $(i = 1, 2, 3)$. Than, the function $F : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ is defined by

$$F(a||b) = b \oplus P_2(a \oplus P_1(b)), \tag{3.2}$$

where $a, b \in \{0,1\}^n$.

In order to describe the quantum distinguisher constructed by H. Kuwakado and M. Morii in [37], we recall the definition of a random permutation.

Definition 3.6. [37] Denote by $\mathcal{P}_n$ the set of all permutations on $\{0,1\}^n$. The permutation $P \in \mathcal{P}_n$ is called a *random permutation* if is chosen from $\mathcal{P}_n$ randomly.

In [37], the following problem has been considered.

Problem 3.7. [37] Let E be either the 3-round Feistel cipher (Figure 5) with internal fixed permutations (publicly known), or a random permutation on $\{0,1\}^{2n}$. Let U_E be unitary operator for computing E, defined as follows

$$U_E|x\rangle|y\rangle = |x\rangle|y \oplus E(x)\rangle,$$

$x, y \in \{0,1\}^{2n}$. By querying U_E, the goal is to determine whether E possesses the Feistel structure (with internal known permutations) or it is selected randomly from $\mathcal{P}_{2n}$. Unitary operator $U_{E^{-1}}$ for computing E^{-1} is not given.

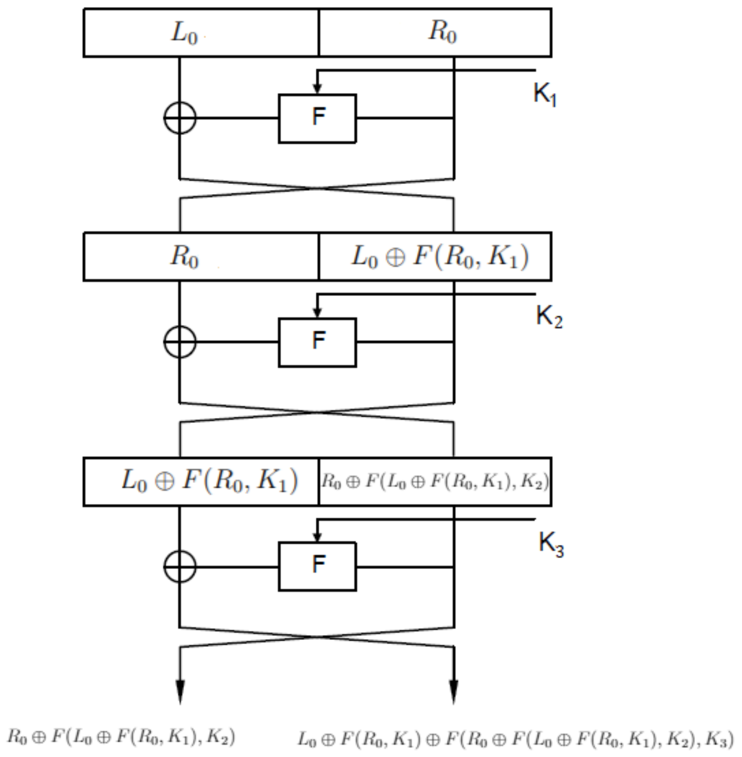
L_0
R_0
K_1
F
R_0
$L_0 \oplus F(R_0, K_1)$
K_2
F
$L_0 \oplus F(R_0, K_1)$
$R_0 \oplus F(L_0 \oplus F(R_0, K_1), K_2)$
K_3
F
$R_0 \oplus F(L_0 \oplus F(R_0, K_1), K_2)$
$L_0 \oplus F(R_0, K_1) \oplus F(R_0 \oplus F(L_0 \oplus F(R_0, K_1), K_2), K_3)$

Figure 5: The 3-round Feistel scheme.

The first step is to construct the Simon's function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ for two distinct arbitrary constants $\alpha, \beta \in \{0,1\}^n$ as

$$f(c||a) = \begin{cases} F(a||\alpha) \oplus \beta, & \text{if } c = 0, \\ F(a||\beta) \oplus \alpha, & \text{if } c = 1, \end{cases} \tag{3.3}$$

where $c \in \{0,1\}$, $a \in \{0,1\}^n$.

Lemma 3.8. *[37] Let E be 3-round Feistel cipher with (known) internal permutations. Then for any distinct strings $c||a$ and $c'||a'$ in $\{0,1\}^{n+1}$*

$$f(c||a) = f(c'||a') \iff c' = c \oplus 1 \wedge a' = a \oplus z,$$

where $z = P_1(\alpha) \oplus P_1(\beta)$.

Proof. ($\Rightarrow$) Suppose that $c = 1$ and $f(c||a) = f(c'||a')$. Using (3.2) and (3.3) we have

$$f(1||a) = F(a||\beta) \oplus \alpha = \beta \oplus P_2(a \oplus P_1(\beta)) \oplus \alpha.$$

Let us consider the following two cases:

Case 1:

$$c' = 0 \implies f(0||a') = F(a'||\alpha) \oplus \beta = \alpha \oplus P_2(a' \oplus P_1(\alpha)) \oplus \beta.$$

Since $f(1||a) = f(0||a')$, it implies

$$\beta \oplus P_2(a \oplus P_1(\beta)) \oplus \alpha = \alpha \oplus P_2(a' \oplus P_1(\alpha)) \oplus \beta, P_2(a \oplus P_1(\beta)) = P_2(a' \oplus P_1(\alpha)).$$

The P_2 is an internal permutations, thus it must be injective, and therefore

$$a \oplus P_1(\beta) = a' \oplus P_1(\alpha),$$

hence $a' = a \oplus P_1(\alpha) \oplus P_2(\beta) = a \oplus z$, where $z = P_1(\alpha) \oplus P_2(\beta)$.

Case 2:

$$c' = 1 \implies f(1||a') = F(a'||\beta) \oplus \alpha = \beta \oplus P_2(a' \oplus P_1(\beta)) \oplus \alpha.$$

From $f(1||a) = f(1||a')$ we obtain

$$\beta \oplus P_2(a \oplus P_1(\beta)) \oplus \alpha = \beta \oplus P_2(a' \oplus P_1(\beta)) \oplus \alpha.$$

Since P_2 and P_1 are permutations it must be $a = a'$.

($\Leftarrow$) Suppose now $c = 1$, $c' = c \oplus 1$ and $a' = a \oplus z$, where $z = P_1(\alpha) \oplus P_2(\beta)$. $c' = 1 \oplus 1 = 0$, conclude $c' = 1 \oplus 1 = 0$. Using (3.2) and (3.3), we are able to determine the following

$$f(1||a) = \beta \oplus P_2(a \oplus P_1(\beta)) \oplus \alpha$$

and

$$\begin{aligned} f(0||a') &= f(0||a \oplus z) \\ &= \alpha \oplus P_2((a \oplus z) \oplus P_1(\alpha)) \oplus \beta \\ &= \alpha \oplus P_2((a \oplus P_1(\alpha) \oplus P_2(\beta)) \oplus P_1(\alpha)) \oplus \beta \\ &= \alpha \oplus P_2(a \oplus P_1(\beta)) \oplus \beta. \end{aligned}$$

From the last two expression we can conclude $f(1||a) = f(0||a')$. The case when $c = 0$ can be proved in a similar way (cf. [37]). □

Lemma 3.8 shows us that the function f is periodic with the period $s = 1||z$, i.e. $f(c||a) = f((c||a) \oplus (1||z))$ which is equivalent to the condition (3.1) in Simon's problem. Therefore, $f(c||a) = f((c||a) \oplus (1||z))$. The next step is to define the oracle function $\mathcal{O}_f$ as follows

$$\mathcal{O}_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle, \tag{3.4}$$

where $x \in \{0,1\}^{n+1}$ and $y \in \{0,1\}^n$. Using this operator we are able to introduce an attack based on Simon's algorithm to determine whether E possesses the Feistel structure (with internal known permutations) or it is a random permutation:

1. Initialize X as the empty set.

2. Run Simon's algorithm.

3. If X does not contain n lineraly independent $v's$ then go back to step 2. Otherwise, find n-bit string $z = (z_0, z_1, \ldots, z_{n-1})$ by solving this linear system of equations:

$$\begin{pmatrix} v_1^0 & v_2^0 & \ldots & v_n^0 \\ v_1^1 & v_2^1 & \ldots & v_n^1 \\ \vdots & & \ddots & \vdots \\ v_1^{n-1} & v_2^{n-1} & \ldots & v_n^{n-1} \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} v_0 p \\ v_0^1 \\ \vdots \\ v_0^{n-1} \end{pmatrix} \quad (mod\ \ 2), \tag{3.5}$$

   where $v_j^i \in \{0,1\}$ and $(v_0^i, v_1^i, \ldots, v_n^i) \in X$.

4. Choose $u \in \{0,1\}^{n+1}$ at random. Let $u' = u \oplus (1||z)$. Compute $f(u)$ and $f(u')$. If $f(u) = f(u')$ then E is supposed to be with the internal permutations, otherwise it is with the random permutations.

Let us analyze the case when E possesses the Feistel structure. After applying $\mathcal{O}_f$ and measurement the second register, from Simon's algorithm (**Step 4**) implies that the state is

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus (1||z)\rangle).$$

Applying the Hadamard $H^{\otimes(n+1)}$ transform we have,

$$\begin{aligned} |\psi_1\rangle &= \frac{1}{\sqrt{2^{n+2}}} \sum_{v \in \{0,1\}^{n+1}} ((-1)^{x \cdot v} + (-1)^{x \oplus (1||z) \cdot v} |v\rangle) \\ &= \frac{1}{\sqrt{2^{n+2}}} \sum_{v \in \{0,1\}^{n+1}} (-1)^{x \cdot v} (1 + (-1)^{(1||z) \cdot v}) |v\rangle. \end{aligned}$$

We obtain the result from **Step 5** of Simon's algorithm. Measuring this state, we have $(1||z) \cdot v = 0$, which is a solution of one equation in (3.5). Since we assumed that E originates from the Feistel structure, then from Lemma 3.8 follows $f(u) = f(u \oplus (1||z))$ and the output of this algorithm is correct. On the other hand, if E is a random permutation, then the vector z will be a random vector if exists. In that case, we make an error if we do not have n independent vectors after $2n$ extractions. Therefore the probability that $f(u) = f(u' \oplus 1||z)$ is $p = \frac{2^n - 1}{2^{2n} - 1} \approx \frac{1}{2^n}$. In this way, the main result in [37] is proven.

Theorem 3.9. *There is a quantum algorithm for distinguishing the* 3*-round Feistel cipher with internal known permutations from a random permutation on* $\{0,1\}^{2n}$. *The complexity of the algorithm is* $O(n)$ *and the error probability is approximately equal to* $\frac{1}{2^n}$.

3.1.2 Even-Mansour's cipher

Let P be a random public permutation on $\{0,1\}^{\frac{n}{2}}$ and key $k = k_1||k_2$ where $k_1, k_2 \in \{0,1\}^{\frac{n}{2}}$. The encryption function $E_k : \{0,1\}^n \times \{0,1\}^{\frac{n}{2}} \to \{0,1\}^{\frac{n}{2}}$ of Even-Mansour cipher (EM) is defined as

$$E_k(p) = P(k_1 \oplus p) \oplus k_2 = c,$$

where c is obtained ciphertext. Therefore, the decryption function $D_k : \{0,1\}^n \times \{0,1\}^{\frac{n}{2}} \to \{0,1\}^{\frac{n}{2}}$ is

$$D_k(c) = P^{-1}(c \oplus k_2) \oplus k_1 = p,$$

where p is a plaintext. For more details about this cryptography construction, we refer reader to [22].

In [38], M. Mori and H. Kuwakado introduced a quantum version of Even-Mansour's cipher, which is defined as a unitary operator U_{E_k} given as

$$U_{E_k}|x\rangle|y\rangle = |x\rangle|y \oplus E_k(x)\rangle,$$

where $x, y \in \{0,1\}^{\frac{n}{2}}$. In general, the goal is to find the secret key k, and in this process the adversary is allowed to use the unitary operator U_{E_k}. In [38], authors described a quantum algorithm similar to Simon's algorithm, which is given as follows. Firstly, the define the function

$$f(x) = E_k(x) \oplus P(x) = P(k_1 \oplus x) \oplus k_2 \oplus P(x).$$

For $y = x \oplus k_1$, we have

$$f(y) = P(k_1 \oplus y) \oplus k_2 \oplus P(y) = P(k_1 \oplus x \oplus k_1) \oplus k_2 \oplus P(x \oplus k_1) = P(x) \oplus P(x \oplus k_1) = f(x),$$

i.e. f satisfies Simon's condition. Now, we consider the application of Simon's algorithm in details.

Let set Z be initialized as an empty set, then the quantum algorithm for distinguishing EM cipher consists of the following steps [22]:

1. Two $\frac{n}{2}$-qubit registers are initialized to the zero state:

$$|\psi_0\rangle = |0\rangle^{\otimes \frac{n}{2}} \otimes |0\rangle^{\otimes \frac{n}{2}}.$$

2. Apply the Hadamard transform $H^{\otimes \frac{n}{2}}$ the obtain the state

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{\frac{n}{2}}}} \sum_{x \in \{0,1\}^{\frac{n}{2}}} |x\rangle \otimes |0\rangle^{\otimes \frac{n}{2}}.$$

3. Apply the unitary operator (black box/oracle) defined as $U_f|x\rangle|y\rangle = |x\rangle \otimes |y \oplus f(x)\rangle$ and obtain
$$|\psi_2\rangle = \frac{1}{\sqrt{2^{\frac{n}{2}}}} \sum_{x\in\{0,1\}^{\frac{n}{2}}} |x\rangle \otimes |f(x)\rangle.$$

4. Measure the second register, the one that contains the values of f. Let y be the measurement result and denote by X_y set of all x such that $y = f(x)$. For simplicity, omit the second register then the resulting state is given by
$$|\psi_3\rangle = \frac{1}{\sqrt{|X_y|}} \sum_{x\in X_y} |x\rangle.$$
From the definition of f, we have that if $f(x) = y$, than $f(x \oplus k_1) = y$. Therefore, the state $|\psi_3\rangle$ can be rewritten as:
$$|\psi_3\rangle = \frac{1}{\sqrt{|X_y|}} \sum_{i=1}^{|X_y|} |x_i\rangle = \frac{1}{\sqrt{|X_y|}} \sum_{i=1}^{\frac{|X_y|}{2}} |x_i\rangle + |x_i \oplus k_1\rangle.$$

5. Applying the Hadamard transform $H^{\otimes\frac{n}{2}}$ to $|\psi_3\rangle$ gives
$$|\psi_4\rangle = \frac{1}{\sqrt{|X_y|2^{\frac{n}{2}}}} \sum_{\substack{1\le i\le \frac{|X_y|}{2}\\ z\in\{0,1\}^{\frac{n}{2}}}} ((-1)^{x_i\cdot z} + (-1)^{(x_i\oplus k_1)\cdot z})|z\rangle.$$

6. Measure the register. Notice that we can write $((-1)^{x_i\cdot z} + (-1)^{(x_i\oplus k_1)\cdot z})|z\rangle = (-1)^{x_i\cdot z}(1 + (-1)^{x_i\oplus k_1})|z\rangle$. Hence, the measurement result satisfies
$$z \cdot k_1 = 0,$$
where $z \in \{0,1\}^{\frac{n}{2}}$.

7. Append z to the set Z. If Z does not contains $\frac{n}{2}$ independent $z's$ then go back to **Step 2**. On the other hand solve the following system of equations
$$z_i \cdot k_1 = 0, \quad i \in \{1, 2, \ldots, \frac{n}{2} - 1\}.$$

8. Choose an arbitrary $p \in \{0,1\}^{\frac{n}{2}}$, and let k_2 be given by
$$k_2 = E_k(p) \oplus P(p \oplus k_1).$$
If we obtained k_1 correctly solving previous system, then one can also obtain k_2 and thus $k = k_1||k_2$.

Recall that in **Step 7** one obtains $\frac{n}{2} - 1$ linearly independent equations. In order to do this the adversary must repeat the procedure from **Step 2** to **Step 7**, say $r = \frac{n}{2} + \epsilon$ times. Then the probability to determine k_1 is

$$\prod_{i=\epsilon+2}^{r} (1 - \frac{1}{2^i}) > 1 - \frac{1}{2^\epsilon + 1},$$

which is independent of n. Thus, the time complexity of this algorithm is $O(n)$. For classical equivalent it was shown in [38] that the algorithm needs $O(n^3)$ steps. Therefore, Even-Mansour's cipher is insecure in quantum environment.

3.1.3 Application of Simon's algorithm to CBC-MAC

In this section, we describe a quantum algorithm introduced by T. Santoli in [49], which is used to attack a block cipher in the mode called cipher block chaining message authentication code (CBC-MAC). The main goal of message authentication and integrity is to verify that the message was not changed in transit and that the sent message came from a stated sender, respectively. The message authentication process is generally realized by message authentication codes (MACs), which works as follows. To mark the message m, the sender takes an arbitrary key k and uses the MAC algorithm to calculate the tag t of the message m, i.e $MAC(k, m) = t$. Then a pair (m, t) is sent to the receiver, who uses the same key k to check if t is a tag of m or not.

CBC-MAC (for more details on CBC-MAC algorithm we refer reader to [42]) is based on a pseudorandom permutations and usually has many variants. Unless otherwise stated, all definitions are from [33].

Definition 3.10. Let $F : \{0, 1\}^* \times \{0, 1\}^* \to \{0, 1\}^*$ be an arbitrary function. We say that F is an *efficient keyed function* if for any key k there exist a polynomial-time algorithm which computes $F_k(x)$ where x is an input. It is *length-preserving* if it accepts only pairs of inputs (k, x) where k, x and the output $F_k(x)$ have the same length.

In order to define pseudorandom functions, we first describe meaning of an *oracle-distinguisher*. It consists of an algorithm $D^{\mathcal{O}}$ which has an access to an oracle function denoted by $\mathcal{O}$.

Definition 3.11. Let $F : \{0, 1\}^* \times \{0, 1\}^* \to \{0, 1\}^*$ be an efficient length-preserving a keyed function. F is a *pseudorandom function* if for any probabilistic polynomial-time oracle distinguisher D, there exists a negligible function $\sigma(n)$ such that:

$$\left| Pr[(D^{F_K})(1^n) = 1] - Pr[D^f(1^n) = 1] \right| \leq \sigma(n),$$

where a negligible function is one which grows slower than any inverse polynomial in n, or more precisely, for any polynomial p there exists an $n_0 > 0$ such that for all integers $n > n_0$, $\sigma(n) < \frac{1}{p(n)}$.

Definition 3.12. Let F be a mapping $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. F is *keyed permutation* if for any $K \in \{0,1\}^k$ the function F_K is a permutation from $\{0,1\}^n$ to $\{0,1\}^n$.

Definition 3.13. A keyed permutation is called *efficient* if for any $K \in \{0,1\}^k$ there is a polynomial-time algorithm to evaluate $F_K(x)$ and $F_K^{-1}(x)$ respectively, for any input $x \in \{0,1\}^n$.

Definition 3.14. Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient keyed permutation. F is called a *pseudorandom permutation* if for any probabilistic polynomial-time distinguisher D, there exists a negligible function $\sigma(n)$ such that:

$$\left| Pr[(D^{F_K})(1^n) = 1] - Pr[D^f(1^n) = 1] \right| \leq \sigma(n),$$

where $K \in 0, 1^n$ and $P_n \in \mathcal{P}_n$ are chosen uniformly at random.

Similarly as in the case of classical definitions, a quantum pseudorandom permutations are defined. We use a quantum oracle-distinguisher algorithm $D^{|\mathcal{O}\rangle}$, which is allowed to make a quantum queries of the form

$$|x\rangle|y\rangle \to |x\rangle|y \oplus \mathcal{O}(x).$$

Definition 3.15. Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient keyed permutation. F is a *quantum pseudorandom permutation* if for any probabilistic polynomial-time quantum distignuisher D, there exists a negligible function $\sigma(n)$ such that:

$$\left| Pr[(D^{|F_K\rangle})(1^n) = 1] - Pr[D^{|P_n\rangle}(1^n) = 1] \right| \leq \sigma(n).$$

For a pseudorandom function $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, CBC-MAC construction is described as follows. Define the function $CBC^l : \{0,1\}^k \times \{0,1\}^s \to \{0,1\}^n$ as

$$CBC(K, m_1||m_2||m_3 \dots m_l) = CBC^l_{F_K}(m_1||m_2||m_3|| \dots ||m_l),$$

where $s = ln$ and

$$CBC^l_{F_K}(m_1||m_2||m_3|| \dots ||m_l) = F_K(F_K(\dots(F_K(m_1) \oplus m_2) \oplus \dots)m_l). \tag{3.6}$$

Using (3.6), a tag t of the message m of length s can be computed as $t = CBC^l(K, m)$. In [49] a quantum forgery attack on CBC-MAC is described as follows.

A message m with prefix $\alpha_1 \in \{0,1\}^n$ is given and it is tagged using CBC-MAC for which it is assumed that pseudorandom permutations F_K is replaced with a random permutation P. For $i = 1, \dots l-1$ define functions $g_i : \{0,1\}^{n+1} \to \{0,1\}^n$

$$g_i(b||x) = CBC^l_P(\alpha_b||0^{(i-1)n)}||x||0^{(l-i-1)n} = P^{l-i}P^j(\alpha_b) \oplus x), \tag{3.7}$$

where $b \in \{0,1\}$. We have the following result which regards the periods of g_i.

Lemma 3.16. *For distinct elements $u, v \in \{0,1\}^{n+1}$, it holds $g_i(u) = g_i(v)$ if and only if $v = u \oplus 1||(P^i(\alpha_0)) \oplus P^i(\alpha_1)$.*

Proof. ($\Rightarrow$) The statement if proved only for $b = 0$, since in a similar way one proves it for $b = 1$. Hence, let us denote by $z = P^i(\alpha_0) \oplus P^i(\alpha_1)$. Since $u = b||x$ and $v = b'||x'$, the claim of the lemma is equivalent to $g_i(b||x) = g_i(b'||x') \iff b' = b \oplus 1$ and $x' = x \oplus z$.

Now, let us assume $g_i(u) = g_i(v)$ for distinct elements $u, v \in \{0,1\}^{n+1}$ and $b = b' = 0$. Hence, from equation (3.7) we have

$$P^{l-i}(P^i(\alpha_0) \oplus x) = P^{l-i}(P^i(\alpha_0) \oplus x')$$

Since P is permutation we obtain $x = x'$. It follows $u = v$ which is a trivial case. Suppose now $b = 0$ and $b' = 1$. Then, we have

$$P^{l-i}(P^i(\alpha_0) \oplus x) = P^{l-i}(P^i(\alpha_1) \oplus x')$$

Since P is a permutation we have

$$x' = x \oplus P^i(\alpha_1) \oplus P^i(\alpha_1) = x \oplus z.$$

($\Leftarrow$) Suppose that $b = 0$, $b' = b \oplus 1 = 1$ and $x' = x \oplus z$. Let us determine $g_i(u)$ and $g_j(v)$, respectively. From equation (3.7) we have

$$\begin{aligned}
g_i(u) &= g_i(0||x) = P^{l-i}(P^i(\alpha_0) \oplus x) \\
g_i(v) &= g_i(1||x') = P^{l-i}(P^i(\alpha_1) \oplus x') \\
&= P^{l-i}(P^i(\alpha_1) \oplus x \oplus P^i(\alpha_1) \oplus P^i(\alpha_0) \\
&= P^{l-i}(P^i(\alpha_0) \oplus x = g_i(u).
\end{aligned}$$

□

The above property of functions g_i is essentially equivalent to the condition of Simon's problem. In [49], the quantum attack on CBC-MAC based on Simon's algorithm is described as follows:

1. Construct the unitary operator U_{g_i}, for $i = 1, \ldots, l-1$ as $U_{g_i}|x\rangle|y\rangle = |x\rangle|x \oplus g_i(x)\rangle$ and for $i = 1, \ldots, l-1$ apply Simon's algorithm to find $z^i = P^i(\alpha_0) \oplus P^i(\alpha_1) \neq 0^n$. All queries made in this process are on superpositions of messages $m_1||m_2||m_3 \ldots ||m_l$ such that $m_j = 0^n$ for at least one $j \in \{1, \ldots, l\}$.

2. Compute $t_0 = CBC_P^l(\alpha_0||0^{(l-1)n}) = P^l(\alpha_0)$ and $t_1 = CBC_P^l(\alpha_1||0^{(l-1)n} = P^l(\alpha_1)$ classically. Similarly as in the first step, these queries are made on messages $m_1||m_2||m_3 \ldots ||m_l$ such that $m_j = 0^n$ for at least one $j \in \{1, \ldots, l\}$.

3. Assume first that l is even, then forge $(m, t) = (\alpha_1||z^1||z^2|| \ldots ||z^{l-1}, t_0)$.
Suppose now that l is odd. In this case we forge $(m, t) = (\alpha_1||z^1||z^2|| \ldots ||z^{l-1}, t_1)$.

The following lemma is proved by T. Santoli in [49] using the induction method for $l \geq 3$.

Lemma 3.17. *[49] The tag t is a valid tag for the message m, which means that $t = CBC_P^l(m)$, where $t = t_0$ if l is even and $t = t_1$ if l is odd.*

Proof. Base case: For $l = 3$ (l is odd), we have $t = t_1 = P^3(\alpha_1)$, and thus

$$\begin{aligned}
&CBC_P^3(\alpha_1||z^1||z^2) = P(P(P(\alpha_1) \oplus z^1) \oplus z^2) \\
&= P(P(P(\alpha_1) \oplus P(\alpha_0) \oplus P(\alpha_1)) \oplus P^2(\alpha_0) \oplus P^2(\alpha_1)) \\
&= P((P^2(\alpha_0) \oplus P^2(\alpha_0) \oplus P^2(\alpha_1) \oplus P^2(\alpha_1)) = P^3(\alpha_1) = t_1.
\end{aligned}$$

For $l = 4$ (l is even), we have $t = t_0 = P^4(\alpha_0)$, and thus

$$\begin{aligned}
&CBC_P^4(\alpha_1||z^1||z^2||z^3) = P(P(P(P(\alpha_1) \oplus z^1) \oplus z^2) \oplus z^3) \\
&= P(P(P(P(\alpha_1) \oplus P(\alpha_0) \oplus P(\alpha_1)) \oplus P^2(\alpha_0) \oplus P^2(\alpha_1)) \oplus P^3(\alpha_0) \oplus P^3(\alpha_1)) \\
&= P(P(P^2(\alpha_0) \oplus P^2(\alpha_0) \oplus P^2(\alpha_1)) \oplus P^3(\alpha_0) \oplus P^3(\alpha_1)) \\
&= P(P^3(\alpha_1) \oplus P^3(\alpha_0) \oplus P^3(\alpha_1)) = P^4(\alpha_0) = t_0.
\end{aligned}$$

Induction hypothesis: We claim that the forging attack is successful for l. We want to prove that it holds for $l + 1$.
Induction step: Consider the following two cases:
Case 1: l being odd implies that $l + 1$ is even. In this case $CBC_P^l = P^l(\alpha_1)$, and thus

$$\begin{aligned}
&CBC_P^{l+1}(\alpha_1||z^1||z^2|| \ldots ||z^{l-1}||z^l) \\
&= P(CBC_P^l(\alpha_1||z^1||z^2|| \ldots ||z^{l-1}||z^{l-1}) \oplus z^l) \\
&= P(P^l(\alpha_1) \oplus P^l(\alpha_0) \oplus P^l(\alpha_1)) = P^{l+1}(\alpha_0) = t_0.
\end{aligned}$$

Case 2: Suppose now l is even, then $l+1$ must be odd. Hence, similarly like above we obtain

$$\begin{aligned}&CBC_P^{l+1}(\alpha_1||z^1||z^2||\dots||z^{l-1}||z^l)\\&=P(CBC_P^l(\alpha_1||z^1||z^2||\dots||z^{l-1}||z^{l-1})\oplus z^l)\\&=P(P^l(\alpha_0)\oplus P^l(\alpha_0)\oplus P^l(\alpha_1))=P^{l+1}(\alpha_1)=t_1.\end{aligned}$$

Hence, we proved by induction for $l \geq 3$ that t is a valid tag for m. □

3.1.4 Quantum slide attack

Slide attacks were first introduced in [8] by Biryukov and Wagner. They can be applied to an r-round block cipher E whose round functions F are all identical and use the same round key k.

More precisely, it is assumed that the attacker has knowledge of encryptions of $2^{\frac{n}{2}}$ plaintext. With a high probability, he obtains a *slid pair*, i.e. a pair of couples (P, C) and (P', C') such that $F(P) = P'$. This implies that $F(C) = C'$. A quantum version of this attack was first described in [12], where it has been shown that Simon's algorithm can be combined with the slide attack. The quantum version is briefly described as follows.

Let F be an unkeyed round function and E_k be a whole encryption function of an r-round block cipher. Then Simon's function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ is defined by

$$f(b||x) = \begin{cases} F(E_k(x)) \oplus x, & \text{if } b = 0,\\ E_k(F(x)) \oplus x, & \text{if } b = 1.\end{cases} \tag{3.8}$$

The slide property shows that all x satisfy $F(E_k(x)) \oplus k = E_k(F(x \oplus k))$, which means that f is a periodic function with the period $s = 1||k$. Note that this is equivalent to Simon's promise 3.1. The following lemma plays an important role in determining the

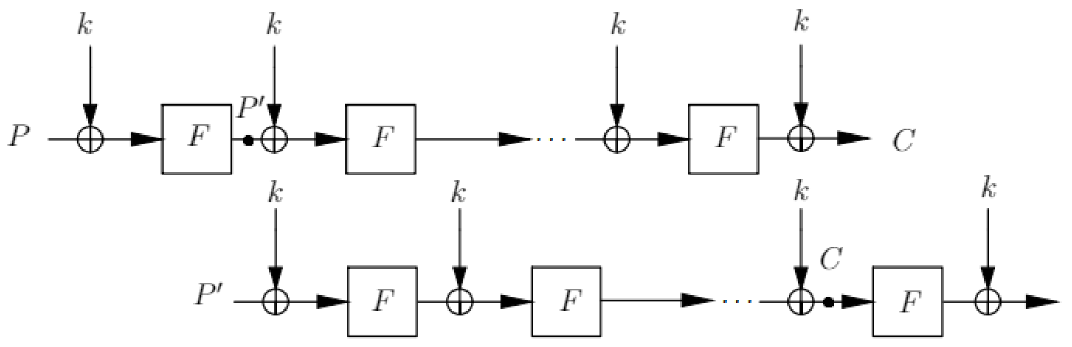

Figure 6: An illustration of the slide attack.

success probability of Simon's algorithm.

Lemma 3.18. *[53] Assume there is a periodic function f with a period s. Then there exists a, $0 < a < 1$ that satisfies*

$$\epsilon(f;s) = \max_{t \notin \{0,s\}} Pr[(f(x)) = f(x \oplus t)] \leq a.$$

Repeating Simon's algorithm cn times, s can be determined with the probability at least $1 - (2(\frac{(1+a)}{2})^c)^n$.

In order to show that Simon's algorithm can be applied to recover the secret key k, by Lemma 3.18 one bounds $\epsilon(f, 1||k)$ under the condition that both $E_k \circ P$ and $P \circ E_k$ are indistinguishable from random permutations.

If $\epsilon(f, 1||k) > \frac{1}{2}$, there exist $(t_0, t_1) \notin \{(0,0),(1,k)\}$, such that $Pr[f(b,x) = f(b \oplus t_0, x \oplus t)] > \frac{1}{2}$. Now, assuming that $t_0 = 0$ we have that $t \neq 0$ and thus

$$Pr[f(b,x) = f(b, x \oplus t)] > \frac{1}{2},$$

which is according to (3.8) equivalent to

$$Pr[F(E_k(x)) = F(E_k(x \oplus t)) \oplus t] > \frac{1}{2},$$

when $b = 0$ or

$$Pr[E_k(F(x)) = E_k(F(x \oplus t)) \oplus t] > \frac{1}{2},$$

for $b = 1$. Now, for $t_0 = 1$ we have $t \neq k$, and similarly as in the previous part for $b = 0$ we have

$$Pr[F(E_k(x)) \oplus x = E_k(F(x \oplus t)) \oplus x \oplus t] > \frac{1}{2},$$

or

$$Pr[E_k(F(x \oplus k)) \oplus k = E_k(F(x \oplus t)) \oplus t] > \frac{1}{2}$$

for $b = 1$. This means there is a differential in $E_k \circ F$ and $F \circ E_k$ with probability 1/2. On the other hand, one can conclude that $\epsilon(f, 1||k) \leq \frac{1}{2}$ unless $E_k \circ F$ or $F \circ E_k$ have differentials with probability 1/2. Therefore, according to Lemma 3.18, Simon's algorithm can be combined with slide attacks. In this way the slide attack is improved and its time complexity in quantum environment is $O(n)$, while the classical one has the complexity $O(2^{\frac{n}{2}})$.

3.1.4.1 Quantum slide attack on LED block cipher

Many existing block ciphers are based on EM structure. One such cipher is *Light Encryption Device* (shortly LED), proposed by Guo at al. in [26]. It is a 64-bit block cipher, with 64-bit or 128-bit key. In the following exposition, we are interested in 64-bit keylength (LED-64), with the corresponding number of being 32. Now we describe the specifications of a 64-bit LED block cipher with a 64-bit key [26].

A 64-bit plaintext p is divided into 16 nibbles $p_0||p_1||\dots||p_{15}$, which are represented as a 4×4 matrix

$$\begin{pmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \\ p_{12} & p_{13} & p_{14} & p_{15} \end{pmatrix}.$$

This is the initial value of the cipher state. In general, key sizes that are not a multiple of 4 bits are not included. Denote by $k_0, k_1, \dots, k_l$ the l nibbles of the key K, then the i-th subkey SK^i are represented in the matrix form as follows

$$\begin{pmatrix} sk_0^i & sk_1^i & sk_2^i & sk_3^i \\ sk_4^i & sk_5^i & sk_6^i & sk_7^i \\ sk_8^i & sk_9^i & sk_{10}^i & sk_{11}^i \\ sk_{12}^i & sk_{13}^i & sk_{14}^i & sk_{15}^i \end{pmatrix},$$

where $sk_j^i = k_{(j+i\cdot 16 \; mod \; l)}$. For a 64-bit key K ($l = 16$), represented by $K = k_0||k_1||\dots||k_{15}$, we have that all subkeys (SK) are equal to K, which is shown by the matrix

$$\begin{pmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{pmatrix}.$$

The round function of LED is described using two operations, firstly **addRoundKey**(state,SK^i) and second **step**(state) operation. The first **addRoundKey**(state,SK^i) combines nibbles of subkey SK^i with the *state*. The second operation consists of four transformations in the sequence of AddConstants, SubCells, ShiftRows and MixColumn, which are explained in [26]. The number of steps r during encryption depends on the key size, where for instance, for a 64-bit key we have $r = 8$.

Quantum slide attack on LED-64

In [19] J. Guo et al. proposed a quantum attack on LED-64 block cipher, which is similar to attacks on Feistel network and Even-Mansour constructions, described in the Subsections 3.1.1 and 3.1.2, respectively.

Define the function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ as follows:

$$f(b||x) = \begin{cases} P_{r+1}((E_1(x)) \oplus x, & \text{if } b = 0, \\ E_2(P_1(x)) \oplus x, & \text{if } b = 1, \end{cases} \tag{3.9}$$

where E_1 represents the original LED-64, while E_2 represents the altered LED-64 with a change in the initial round constant, and P_1, P_{r+1} are the first and $(r+1)$-th round function of E_1, respectively. More precisely

$$E_1(x) = (P_{k_r} \circ P_{k_{r_1}} \circ \cdots \circ P_{k_1})(x) \oplus k,$$

where $P_{k_r} = P_r(x \otimes k)$ and

$$E_2(x) = (P'_{k_r} \circ P'_{k_{r_1}} \circ \cdots \circ P'_{k_1})(x), \oplus k,$$

such that $P'_{k_r} = P'_r(x \otimes k)$ and $P'_i = P_{i+1}$ for $i = 1, \ldots, r$. Note that for LED-64 $n = 64$ and $r = 8$.

Lemma 3.19. *For a function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ defined by (3.9), it holds that*

$$f(b||x) = f(b'||x') \iff b' = b \oplus 1 \wedge x' = x \oplus k.$$

Therefore, the function f is periodic with the period $s = 1||k$. Now, we can describe the algorithm introduced in [19], to recover the key k:

1. The first step is to construct a quantum circuit suitable for Simon's algorithm as shown in Figure 8 . The idea is the same as in [2]. Let us consider a public random permutation P such that $P : |x\rangle|y\rangle \to |x\rangle|y \oplus P(x)\rangle$ and the function $g : |x\rangle|y\rangle \to |x\rangle|y \oplus g(x)\rangle$, then the quantum gate of controlled P is $CP : |X\rangle|y\rangle \to |x\rangle|y \oplus bP(x)\rangle$ and the quantum gate of controlled function g is $Cg : |x\rangle|y\rangle \to |x\rangle \oplus bg(x)\rangle$. Note that gate CE_i $(i = 1, 2)$ is constructed from oracle the E_i and

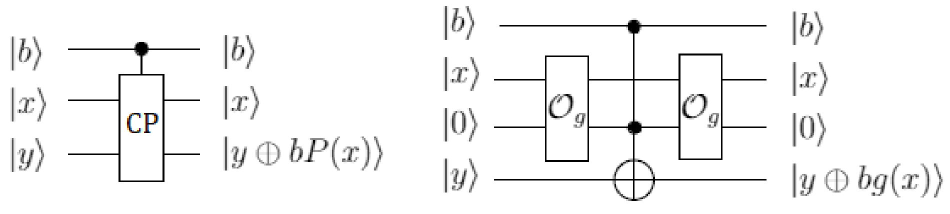

Figure 7: The CP and Cg circuits.

   $CCNOT$ gate [2].

2. Initialize X as an empty set to store the vector z. Choose $4n + 1$ qubits state and divide it into five quantum registers denoted by A, B, C, D and F, such that the first register A consists of 1 qubit and the others consist of n qubits. Apply the Hadamard transform $H^{\otimes n}$ to the second register B to obtain the state $|\psi_1\rangle$. Repeat the next two steps $c(n + 1)$ times, where c is some constant.

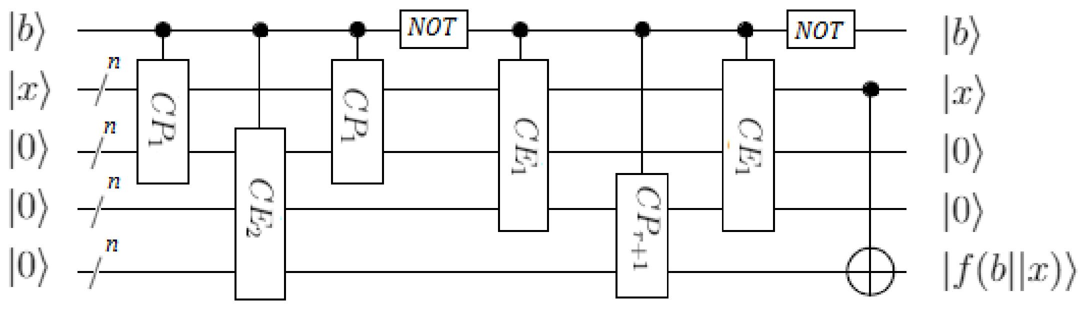

Figure 8: The quantum circuit which realizes the function f.

3. Call the oracle function $\mathcal{O}_g$ to map state $|\psi_1\rangle$ to $|\psi_2\rangle$, stored in the last register F and measure it. Similarly as in the **Step 4** in Simon's algorithm, B is reduced to
$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|y\rangle|0\rangle + |y \oplus s\rangle|0\rangle).$$

4. Apply the Hadamard transform $H^{\otimes(n+1)}$ to obtain
$$|\psi_4\rangle = \frac{1}{\sqrt{2^{n+2}}}\sum(-1)^{y\cdot z}(1+(-1)^{s\cdot z})|z\rangle.$$
This step is in details described in the 5th part of Simon's algorithm. After that the second register is measured to obtain some vector z such that $y \cdot s = 0$.

5. We want to determine classically, if the found vector z is linearly independent of the vectors in X. If it is true, denote it with z_i (i is the cardinally of X and we count from 0) and add in the set X. If $i < n-1$, return to the previous step and repeat the process. If $i = n-1$ it means we have n linearly-independent vectors. We go to the 7th step.

6. The attack is not realized if the above process ends naturally after $c(n+1)$ steps.

7. The attack is successful if this step is achieved. Add the $(n+1)-th$ vector z_n, which is linearly independent of the elements of X and not to orthogonal to s. Therefore, there is a system of $n+1$ linearly independent equations such that
$$z_i \cdot s = \begin{cases} 0, & i = 0, 1 \ldots, n-1, \\ 1, & i = n. \end{cases}$$
We can use the improved Gaussian or some other method to solve this system and determine $s = (1||k)$, where k is the secret key.

Before stating the result, which gives us an information about the complexity of the described quantum algorithm that attacks the LED-64 block cipher, we state the following lemma.

Theorem 3.20. *[19] As in the case of the quantum slide attack on LED-64, the probability of success is about* 99.9% *the required space complexity is* 2^9*, time complexity is* 2^{12} *quantum query and* 2^{26} *classical computations.*

Proof. Assuming that $c = 3$ in Step 2 and $a = \frac{1}{2}$, we repeat described procedure $3(n+1)$ times. By Lemma 3.18 the probability of success is $1 - (2(\frac{1+\frac{1}{2}}{2})^3)^{64} \approx 99.9\%$. According to the quantum circuit given in Figure 8, the attack requires $4n + 1$ qubits in total. Since, for LED-64 block cipher $n = 64$, then $4 \cdot 64 + 1 = 257 < 2^9$ qubits are required. Therefore, the space complexity is approximately 2^9. In order to determine time complexity of the quantum attack on LED-64, we consider Figure 8. There are several operations to be performed: CP_j four times, CE_i twice and $CNOT$ once. Additionally, each CE_i consists of three unit operations (see Figure 7), so it is six more transformations. In Steps 2 and 4 one applies the Hadamard transforms. Hence, the total sum of the unit operations is 15. On the other hand, we repeat the procedure at most $c(n+1)$ times, where it is assumed that $c = 3$. Therefore the total required time complexity for quantum query is $15 \cdot 3 \cdot (64 + 1) \approx 2^{12}$.

The time complexity of the classical computation is mainly determined by **Steps 5 and 7**. Based on the improved Gaussian elimination method introduced in [41] we need about $(n+1)^3$ the classical computational steps to solve Step 5. For each iteration of step 5, not only the linear dependence of the obtained vector z on the vectors from the set X is checked, but also to ensure that the matrix L generated by the set X has the simplest form $L = (z_0, z_1 \dots z_{n-1})^\top$. In **Step 7**, according to the $n \times (n+1)$ matrix L, we add the $(n+1)$-th vector z_n, which is linearly independent of the elements of X and not to orthogonal to s, then construct a system of $n + 1$ independent equations. Solving the system for $s = 1||k$, the required classical computational complexity is about $(n + 1)^2$. In total, the time complexity required for the classical computation is $c(n + 1)(n + 1)^3 + (n + 1)^2 = 3(64 + 1)(64 + 1)^3 + (64 + 1)^2 \approx 2^{26}$. □

3.1.4.2 Quantum attack on 1k-Feistel network

We start by recalling the definition of an rk-Feistel block cipher.

Definition 3.21. [12] An rk-Feistel version of Feistel cipher is a block cipher where the round keys form a periodic sequence of period r. In particular, $1k$-Feistel denotes the case where all subkeys are equal.

X. Dong et al. in [21] described an attack to $1k$-Feistel network. Let f be a round function of $1k$-Feistel block cipher, and k a secret key. For α is a random constant, let the function $F : \{0, 1\} \times \{0, 1\}^{\frac{n}{2}} \to \{0, 1\}^{\frac{n}{2}}$ is defined as follows:

$$F(b||x) = \begin{cases} E_k(x, \alpha)_R, & \text{if } b = 0, \\ E_k(\alpha, f(x) \oplus x)_L, & \text{if } b = 1, \end{cases} \tag{3.10}$$

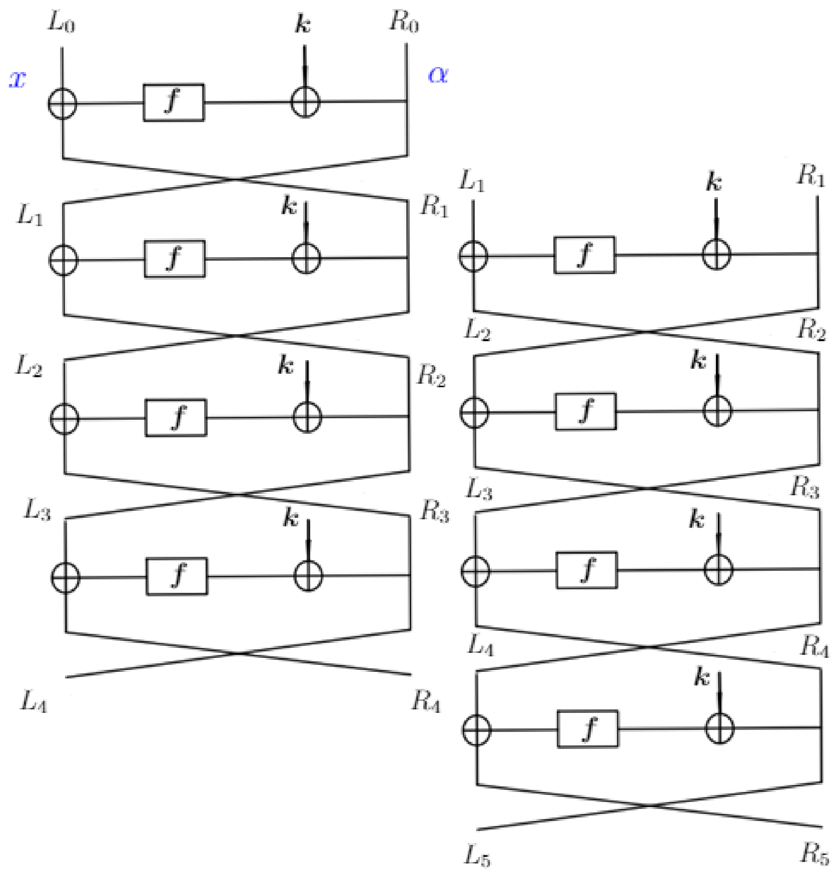

Figure 9: The quantum slide attack on $1k$-Feistel block cipher.

where n is the block size of $1k$-Feistel block cipher E_k, $E_k(\cdot)_L$ and $E_k(\cdot)_R$ are the left branch or right branch, respectively of E_k.

As shown in Figure 9, $E_k(x,\alpha)_R = R_4, E_K(L_1,R_1)_L = L_5 = R_4, L_1 = \alpha$ and $R_1 = f(k \oplus \alpha) \oplus x$ hold. Therefore, from $E_k(x,\alpha)_R = E_k(L_1,R_1)_L = R_4$, one concludes that

$$F(0||x) = E_k(x,\alpha)_R = E_k(\alpha, F(K \oplus \alpha) \oplus x)_L = F(1||x \oplus f(\alpha) \oplus f(k \oplus \alpha)).$$

Hence, $F(b||x)$ is a function with period $s = 1||f(\alpha) \oplus f(k \oplus \alpha)$, thus one can apply Simon's algorithm to attack $1k$- Feistel cipher. Using the results and analysis provided by Leander and May in [39], the time complexity is about $2(n/2 + 1 + \sqrt{n/2+1})$ repetitions of Simon's algorithm, which requires about $n+1$ qubits. Note that, the attack works for any number r of rounds of $1k$-Feistel, but in [21] authors give an example for case when $r = 4$ (cf. Figure 9).

Remark 3.22. Similar attacks can be realized on $2k$ and $4k$-Feistel block ciphers. For more details we refer to [21]. Also, note that an improved version of the attack given for the $1k$-Feistel network, can applied to $1k$-DES as well.

3.1.5 Quantum cryptanalysis of AEZ

We start by recalling the construction of *tweakable block cipeher* proposed by M. Lisov et al. in [40].

The tweakable block cipher E takes three inputs: a *key* $K \in \{0,1\}^k$, a *tweak (initialization vector)* $T \in \{0,1\}^t$ and a *plaintext* $P \in \{0,1\}^n$ and produces as output a *ciphertext* $C \in \{0,1\}^n$, i.e.

$$E : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n.$$

AEZ cryptographic system introduced by V. T. Hoang et al. in [28] is an example of a tweakable block ciphers. It is also an authentication-encryption scheme. AEZ allows keys of any length, but the given key is processed into 384 bits, and if it is not already that length, then one applies a cryptographic hash function. The key K is decomposed in three 128-bits subkeys (I, J, L).

The main building block of AEZ is a tweakable function $E_K^{i,j}$, which is a permutation on 128 bits. For more details related to the structure and design of AEZ, we refer the reader to [10, 28]. In general, there are multiples versions of AEZ, such as AEZv2, AEZv3, AEZv4, AEZv5, AEZ10. In [10], X. Bonnetain proposed a quantum attack on AEZ which is showed that all versions of AEZ can be broken in quantum cryptanalysis. Let us consider subkey functions f_I, f_J and f_L, which are defined as follows:

$$\begin{aligned} f_I(x) &= lastblock(AEZ-core(K,(\tau,N),(0,x,0,x))),\\ f_L(x) &= AEZ-prf(K,(\tau,N,(x,x)),\tau),\\ f_J(x) &= AEZ-prf(K,(\tau,x,x),\tau), \end{aligned}$$

where τ is the output length of a pseudo-random function (PRF), and N is a fixed nonce for each input. Here, AEZ-prf is a pseudo-random function that is called when the message is empty, that takes some associated data and a length τ as arguments, and that outputs a tag of the desired length that can be used to authenticate the associated data. Since the AEZ- core has a more complex description, for more details we refer to [14]. All functions f_I, f_L and f_J are periodic with the following periods $x \oplus I$, $x \oplus J$, $x \oplus 6L$, respectively, and therefore once can apply Simon's algorithm.

In the AEZv4 version there is a small difference for f_I , since the period is not on the whole AEZ-core but only on the last block. One can construct an oracle of f_I from an oracle of the AEZ-core by subtracting out the last block and taking only that. Using this method, one query to f_I costs two queries to AEZ-core. For each case, functions of $n = 128$ bits are queried. To get a success probability of $\frac{1}{2}$, one needs 80% of success for each subkey, which is achieved in 157 queries. The total query complexity of the attack is $628 = 2^{9.3}$. For $k \in \{I, J, K\}$, the attack has the following steps:

1. Query 157 times Simon's routine with f_k.
2. Solve classically the Boolean equation system to get the period of f_k.
3. If this period was a multiple of k, invert to retrieve k.

In this quantum attack, the different functions have the same hidden period. The only condition for the nonce is that it is not entangled with the input value. For f_L, one needs to perform a quantum query with a nonce superposition. If one does not want to allow this, one can always use $f'_L = AEZ - prf(K, (\tau, N, x, x), \tau)$, which satisfies $f'_L(x) = f'_L(x \oplus 12L)$. This has the same query complexity, but a slightly larger data complexity (about $2^{11.5}$).

3.1.6 Quantum attacks on SoEM

One of the interesting applications of Simon's algorithm is the application to Sum of Even-Mansour construction, called SoEM, which is the combination of a sum of permutations and Even-Mansour cipher [15]. More precisely, for permutations $P_1, P_2 \in \mathcal{P}_n$, one can consider a generic construction $SOEM : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as

$$SOEM(k_1, k_2, x) = P_1(x \oplus k_1) \oplus k_1 \oplus P_2(x \oplus k_2) \oplus k_2.$$

There are three variants of SoEM: SoEM1, where permutations P_1 and P_2 are equal, SoEM21, where permutations P_1 and P_2 are independent, but keys k_1 and k_2 are identical (the key space is of n bits) and SoEM2, where both permutations P_1 and P_2 and keys k_1 and k_2 are independent [15]. In [51] Shingawa and Iwata described quantum attack based on Simon's algorithm to SoEM1 and SoEM21, since attack to SoEM22 is a combination of Simon's and Grover's algorithm, which we describe in Chapter 4.

A quantum attack on SoEM1

For $SoEM1$ it holds that $P_1 = P_2 = P$. Therefore, a function $f : \{0,1\}^n \to \{0,1\}^n$ can be define as follows:

$$f(x) = SOEM(k_1, k_2, x) = P(x \oplus k_1) \oplus k_1 \oplus P(x \oplus k_2) \oplus k_2. \tag{3.11}$$

It is not difficult to verify that the following lemma holds.

Lemma 3.23. *Let f be a function defined by (3.11). Then for any two distinct vectors $x, y \in \{0,1\}^n$ it holds that*

$$f(x) = f(y) \iff y = x \oplus k_1 \oplus k_2.$$

Thus, according to Lemma 3.23, f is a periodic function with the period $k = k_1 \oplus k_2$, i.e. f satisfies Simon's condition (3.1). Therefore one can apply Simon's algorithm to determine $k = k_1 \oplus k_2$, by using $O(n)$ qubits and $O(n)$ quantum gates. In order to find k_1 and k_2, a function $g : \{0,1\}^n \to \{0,1\}^n$ is defined as:

$$\begin{aligned} g(x) &= f(x) \oplus P(x) \oplus P(x \oplus k') \\ &= P(x \oplus k_1) \oplus P(x \oplus k_2) \oplus k_2 \oplus P(x) \oplus P(x \oplus k'). \end{aligned}$$

It easy to check that g has the period k_1 which can be computed by applying Simon's algorithm again. Since, we have knowledge about k' and k_1 we can derive $k_2 = k' \oplus k_1$. Therefore the following result holds.

Theorem 3.24. *[51] There exists a quantum attack against SoEM1 that recovers the secret keys k_1 and k_2 with $O(n)$ qubits and $O(n)$ quantum queries.*

A quantum attack on SoEM21

Quantum attack to SoEM21 is quite similar to SoEM1, apart from a few differences. Since, $k_1 = k_2 = k$ and $P_1 \neq P_2$, the function $f : \{0,1\}^n \to \{0,1\}^n$ is defined by

$$\begin{aligned} f(x) &= SOEM21(k.k, x) \oplus P_1(x) \oplus P_2(x) \\ &= P_1(x \oplus k) \oplus P_2(x \oplus k) \oplus k \oplus P_1(x) \oplus P_2(x). \end{aligned} \tag{3.12}$$

In order to determine the periodicity of f, we state the following lemma.

Lemma 3.25. *Let f be a function defined as in relation (3.12). Then for any two distinct vectors $x, y \in \{0,1\}^n$ it holds*

$$f(x) = f(y) \iff y = x \oplus k.$$

Therefore, f given by (3.12) is a periodic function with the period k, i.e. $f(x) = f(x \oplus k)$. Since f satisfies Simon's condition, one is able to apply Simon's algorithm to recover the secret key k in polynomial time $O(n)$.

4 GROVER'S ALGORITHM

In this section, we introduce a quantum algorithm which solves the so-called Grover's problem, which is stated in general as Problem 4.1 below. Lov K. Grover [25] showed that a quantum computer can solve a given problem quadratically faster than any classical computer. In what follows we recall Grover's problem and the algorithm which solves this problem.

Problem 4.1. Given an unstructured system X of $N = 2^n$ elements. The goal is to find elements that satisfy some ordered conditions. Without loss of generality, we assume that there is a unique such element, and we denote it by y.

For $N = 2^n$, this problem can be represented by using the function $f : \{0,1\}^n \to \{0,1\}$

$$f(x) = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{if } x \neq y, \end{cases} \tag{4.1}$$

where one has to find all inputs which map to 1. The probability of finding an element y in N random elements is uniformly distributed, therefore $p_y = \frac{1}{N}$, and thus when $N \to \infty$ probability is equal to zero. Since classical computers can solve this problem in $O(N)$ steps, it requires a lot of time for a large enough N. Now we recall Grover's algorithm, using the similar notation as E. Strubell in [55].

Grover's algorithm: Grover showed that there is a quantum algorithm that solves the previously mentioned problem in $O(\sqrt{N})$ steps:

1. **Input:**

   - A quantum oracle $\mathcal{O}_f$ which is implemented as a unitary operation preforming the transformation $\mathcal{O}_f|x\rangle = (-1)^{f(x)}|x\rangle$, where $f(x)$ satisfies the condition (4.1).
   - n qubits initialized to the state $|0\rangle$ and the oracle qubit q to the state $|1\rangle$, which is flipped if $f(x) = 1$, and is unchanged otherwise.

   Therefore initial state is

$$|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle.$$

2. **Apply a Hadamard transform:** In this step we transform $|\psi_0\rangle$ applying the Hadamard transform $H^{\otimes(n+1)}$:

$$\begin{aligned}|\psi_1\rangle &= H^{\otimes n}|0\rangle^{\otimes n} \otimes H|1\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.\end{aligned}$$

3. **Repeat Grover iterations $r \approx \frac{\pi}{4}\sqrt{2^n}$ times:** Here we omit the last qubit for the simplicity. The first step in the Grover iteration is a call to a quantum oracle $\mathcal{O} : |x\rangle \to (-1)^{f(x)}|x\rangle$, Where $f(x) = 1$ if $x = y$ and $f(x) = 0$ otherwise.
The next part of iteration is the *diffusion transform*, that consists of Hadamard transforms $H^{\otimes n}$, conditionally shift phase and another Hadamard transform $H^{\otimes n}$. The conditional phase shift can be represented by the unitary operator

$$2|0\rangle\langle 0| - I.$$

It is not difficult to see that this operator shifts every state except $|0\rangle$ by -1, i.e.

$$(2|0\rangle\langle 0| - I)|0\rangle = |0\rangle, \;\; (2|0\rangle\langle 0| - I)|x\rangle = -|x\rangle.$$

In summary, this part of the algorithm can be written as

$$H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2H^{\otimes}|0\rangle\langle 0|H^{\otimes} - I = 2|\psi\rangle\langle\psi| - I,$$

where $|\psi\rangle$ is state of the first n qubits in $|\psi_1\rangle$. Thus we can write $|\psi_2\rangle = [2(|\psi\rangle\langle\psi| - I)\mathcal{O}]^r|\psi\rangle \approx |y\rangle$.

4. **Measurement:** We measure the quantum register and obtain the probability, to find y, is equal to 1. So, we conclude that the probability of success is $O(1)$.

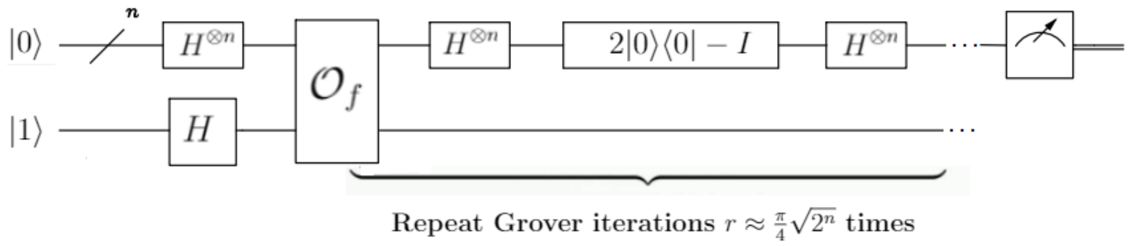
$|0\rangle$
n
$H^{\otimes n}$
$|1\rangle$
H
$\mathcal{O}_f$
$H^{\otimes n}$
$2|0\rangle\langle 0| - I$
$H^{\otimes n}$
Repeat Grover iterations $r \approx \frac{\pi}{4}\sqrt{2^n}$ times

Figure 10: The quantum circuit Grover's algoritm.

Proposition 4.2. *[1] After r iterations of Grover's operator the system is in the superposition*

$$\sin(2r + 1)\theta|\phi_1\rangle + \sin(2r + 1)\theta|\phi_0\rangle,$$

where θ is the angle such that $\sin\theta = \frac{1}{\sqrt{2^n}}$, $|\phi_1\rangle$ and $|\phi_0\rangle$ denotes the projection onto the good and onto to bad subspace, respectively.

Hence, we conclude that after r iterations we have angle $2(k+1)\theta$. If this angle is approximately $\frac{\pi}{2}$, then $|\phi_0\rangle$ and $|\phi_1\rangle$ are almost co-linear. Since for large θ it holds that $\sin\theta \approx \theta = \frac{1}{\sqrt{2^n}}$, we iterate until

$$(2r+1)\theta \cong \frac{\pi}{2} \implies r \approx \frac{\pi}{4}\sqrt{2^n}.$$

This explains the choice of the number of iterations r in **Step 3**.

4.1 APPLICATIONS OF GROVER'S ALGORITHM

The first application of Grover's algorithm in cryptanalysis of block ciphers was discussed and described by A. Jahiko et al in [1]. The idea is to improve the *known-plaintext attack* by Grover's algorithm to be able to recover a secret key.

1. Let $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ be a block cipher. Assume that an attacker has knowledge of the plaintext P and corresponding ciphertext $C = E(P,k')$, where k' is a secret key. Define function $f : \{0,1\}^k \to \{0,1\}$ in the form of (4.1)

$$f(k') = \begin{cases} 1, & \text{if } E(P,k') = C, \\ 0, & \text{otherwise,} \end{cases}$$

   and the oracle function $\mathcal{O}_f|k'\rangle = (-1)^{f(k')}|k'\rangle$.

2. The next step is an initialization of all possible keys of the set $\{0,1\}^k = K$ with the same amplitude, putting them into a superposition (in [1] authors assumed that $n = k = 2^{64}$)

$$|K\rangle = \frac{1}{\sqrt{2^k}} \sum_{k' \in K} |k'\rangle.$$

3. After Grover's iterations are applied r times, then the measurement is performed to obtain key k_i such that $F(k_i) = 1$. Therefore it takes $O(\sqrt{2^n})$ steps to find a secret key. After measurement, the state $|k_i\rangle$ is obtained with a probability at least $\frac{1}{2}$.

From Proposition 4.2, one obtains the key $k_i = \phi_1$. Since the measurement is classical, then the probability to obtain $|\psi_1\rangle$ is equal to $|\sin(2r+1)\theta|^2 \geq \frac{1}{2}$ for $0 \leq \theta \leq \frac{\pi}{2}$ and $\theta \approx \frac{1}{\sqrt{2^n}}$. Hence

$$r \geq \pi\frac{\sqrt{2^n}}{8}.$$

Thus, when $n = 64$ we need $r \approx \pi 2^{29}$ iterations.

Similarly, M. Kaplan in [32] described a quantum attack against 4-round iterated block ciphers, that is equivalent to well known Meet-in-the middle attack, where the attacker has on disposal of enough many pairs (P_i, C_i). In order to describe the attack proposed by M. Kaplan, we follow the same notation as in [32]. We denote by $[N]$ the set of integers $\{1, 2, \dots, N\}$ and by $\mathcal{P}_{[N]}$ the set of permutations of $[N]$. The space of keys is $[N]$ and the space of blocks is $[M]$. We consider the following problem.

Problem 4.3. [32] The 4-*Key Extraction problem* (KE_4) with $P = (P_1, P_2, P_3) \in [M]^4$ and $C = (C_1, C_2, C_3, C_4) \in [M]^4$ takes input F where $F = \{F_1, F_2, \dots, F_N\}$ is a collection of permutations $F_i \in \mathcal{P}_M$. The goal of the problem is to find a quadruple (k_1, k_2, k_3, k_4) such that $C_i = F_{k_4}(F_{k_3}(F_{k_2}(F_{k_1}(P_i))))$ for $i \in \{1, \dots 4\}$.

In [32], Kaplan first considered the decision version of the problem denoted by $d - \mathrm{KE}_4$.

Problem 4.4. *The decision* 4-*Key Extracton problem* with a pair (P, C), permutations $F = \{F_1, \dots, F_N\}$ is to decide if there exist a 4-typle key (k_1, k_2, k_3, k_4) such that $C = F_{k_4}(F_{k_3}(F_{k_2}(F_{k_1}(P))))$.

The d-$(\mathrm{KE})_4 = h$ was expressed in the compositions form:

$$h = \mathbf{SEARCH} \circ (f_0 \wedge g_0, f_1 \wedge g_1, \dots, f_N \wedge g_N),$$

where $\wedge$ is logical AND, and the function $f_X : (\mathcal{P}_{[M]})^N \to \{0, 1\}$ is given by

$$\{F_1, \dots F_N\} = \begin{cases} 1, & \text{if } \exists\, k_1, k_2 : F_{k_2}(F_{k_1}(P)) = X, \\ 0, & \text{otherwise.} \end{cases}$$

and function $g_X : (\mathcal{P}_{[M]})^N \to \{0, 1\}$ such that

$$\{F_1, \dots F_N\} = \begin{cases} 1, & \text{if } \exists\, k_1, k_2 : F_{k_2}(F_{k_1}(X)) = C, \\ 0, & \text{otherwise.} \end{cases}$$

By **SEARCH** we denote the function **SEARCH**: $\{0, 1\}^M \to \{0, 1\}$ with the following properties

$$\mathbf{SEARCH}(y) = \begin{cases} 1, & \text{if } \exists\, i : y_i = 1, \\ 0, & \text{otherwise.} \end{cases}$$

In the other word, the idea is the make compositions of the meet-in-the-middle attack and Grover's search operator. A quantum query complexity is multiplicative under function composition and it is $O(\sqrt{M}\sqrt[3]{N^2})$. The main result of [32] is given as follows.

Theorem 4.5. *There exists a quantum algorithm that solves KE_4 in time $O(\sqrt[6]{7})$ and using memory $O(\sqrt[3]{N^2})$. The time-space product for this attack is $O(\sqrt[6]{N^{11}})$.*

Since the proof is quite lengthly and requires many new definitions and facts, it is omitted here.

4.1.1 Application of Grover's algorithm to a version of Simplified AES

The Advanced Encryption Standard (AES) is a symmetric block cipher established by U.S. National Institute of Standards and Technology (NIST) [4]. Due to the importance of the AES block cipher (and its design philosophy) in symmetric-key cryptography, in this subsection we recall its simplified version and show the application of Grover's algorithm. It is important to note that full version of AES, and in general its design approach, appears to be quantum resistant against the quantum attacks known today. A detailed description of AES is given in [4].

Regarding the application of Grover's algorithm, in 2015 an article [24] was published in which Grover's quantum attack on the AES was discussed for the fist time. For simplicty, in this subsection we present similar results of applying Grover's algorithm to a simplified version of AES (shortly, S-AES). The S-AES has the same structure as AES with difference in the key size (16 bits) and block size (16 bits) [45].

4.1.1.1 The Quantum S-AES

The qaunatum simplified version of AES is introduced by M. Almazrooie et al. in [3]. To describe quantum S-AES it takes 64 qubits and 8 additional qubits to implement an S-Box. Therefore the initial state is $|\psi_0\rangle = |k\rangle^{\otimes 16}|0\rangle^{\otimes 48}$ where the first register represents the master key k. The plaintext is introduced by applying Pauli-X gates to the first 16 qubits. The number of Pauli- X gates is equal to the number of ones in the plaintext.

Substitute Nibbles (SN): In the classical form one nibble is 4 bits or half of byte, so quantum nibble is constitued out of 4 qubits. Hence in the S-AES the initial state $|\psi_0\rangle$ can be represented by 16 nibbles $N_0, N_1, \dots, N_{15}$. Firstly, we describe this step for classical S-AES.

Take a nibble a and substitute it by some other nibble b-obtained as follows. Consider the nibble a in the form of 4-bit column vector, i.e $a = (a_0, a_1, a_2, a_3)^\top$, $a_i \in \{0, 1\}$ for $i = 0, 1, 2, 3$, and compute its inverse $c = a^{-1}$ in the finite field $GF(2^4) = \mathbb{F}_2[x]/(x^4+x+1)$, where x^4+x+1 is an irreducible polynomial over $GF(2)^4$. For more details about properties of finite fields and their constructions see [44].

Remark 4.6. The elements of $GF(2^4)$ are provided in the Apendix A of the end of the thesis.

After computing c, one finds a new nibble b using the following affine transformations:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{4.2}$$

There are a few methods to compute inverse element in a finite field[1]. Since for every finite field $\mathbb{F}$ of order p^k, where p is prime holds $g^{p^k-1} = 1$ for any $g \in \mathbb{F}$, we conclude $g^{-1} = g^{p^k-2}$. Hence, in our case we have:

$$a^{-1} = a^{2^4-2} = a^{14}.$$

Thus, the element a^{14} can be expressed as follows

$$a^{14} = a^2 \cdot (a^2)^2 \cdot ((a^2)^2)^2. \tag{4.3}$$

Therefore, in order to determine a multiplicative inverse, the two multiplications and three square operations are required. The multiplier circuit is used first. Its quantum version is proposed by Cheung et al. in [16] for $GF(2^4)$. The next step is to construct

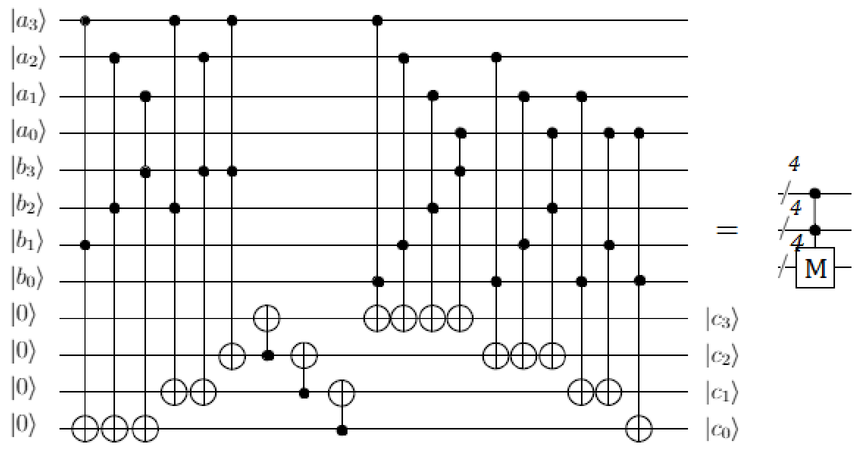

Figure 11: The multiplier quantum circuit of the $GF(2^4)$.

a quantum squarer circuit. Let $a' = a^2 = a \cdot a$, then a' can be computed as follows

$$\begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}. \tag{4.4}$$

The justification is the same as for (4.6). Every element of $GF(2^4)$ can be expressed in the form $a_3x^3 + a_2x^2 + a_1x + a_0$. The matrix in the expression (4.4) can be implemented

[1]The methods for computing inverse element in a finite field are usually based on Euclidian algorithm or on Fermat's little theorem.

in the quantum circuit applying a set of C-NOT gates and using additional auxiliary qubits, due to the dependency among the qubits in the matrix. In order to remove the dependency, CNOT synthetic algorithm proposed in [48] is used. In the following

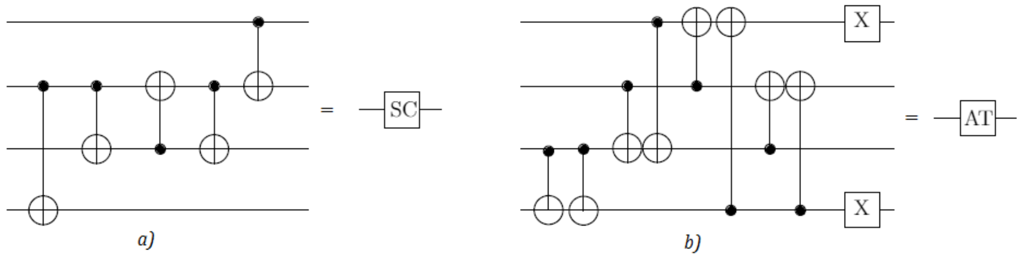

Figure 12: a) The squarer circuit of $GF(2^4)$; b) The circuit of the affine transformation given in the relation (4.2).

example, we show that based on the relation (4.4), for instance, one can find the square of the element $x^2 \in GF(2^4)$.

Example 4.7. Let us consider the element $a = x^2 \in \mathbb{F}_2/(x^4+x+1)$. Since every element in this field can be written in the form $a_3x^3 + a_2x^2 + a_1x + a_0$, or as a matrix $(a_0\ a_1\ a_2\ a_3)^\top$, then $a = x^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}^\top$. According to relation (4.4), we can compute a^2 as follows

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Hence, $a^2 = (1\ 1\ 0\ 0)^\top$, or in a polynomial form $a^2 = x+1$.

After creating the quantum multiplier circuit and the square circuit, the next step is to construct a quantum circuit for the matrix of affine transformation. For this purpose, the synthesis algorithm of C-NOT gates [48] is used again. In order to implement XOR-ed vector we use two Pauili-X gates (see Figure 12 b)). According to (4.3) the complete quantum circuit of substitute nibbles is composition of these three quantum circuits. The next step is shifting the rows.

Shifting the rows: In the quantum circuit we can shift the rows arbitrarily. For instance, in Figure 14, the nibble N_0 is shifted by N_{10} (state $|\psi_1\rangle$).

Mix columns(MC): In the classical S-AES we make the state matrix $S = \begin{pmatrix} S_0 & S_2 \\ S_1 & S_3 \end{pmatrix}$, where S_i represents nibbles of a plaintext. The MixColumns transformation in S-AES is expressed as follows:

$$\begin{pmatrix} \widetilde{S}_0 \\ \widetilde{S}_1 \end{pmatrix} = \begin{pmatrix} 1 & x^2 \\ x^2 & 1 \end{pmatrix} \cdot \begin{pmatrix} S_0 \\ S_1 \end{pmatrix} \tag{4.5}$$

where all operations are in the $\mathbb{F}_2/(x^4+x+1)$.

Remark 4.8. If we consider an element $a \in \mathbb{F}_2/(x^4+x+1)$ in the form $a = a_3x^3 + a_2x^2 + a_1x + a_0$ and multiply by x^2, then their product is of the form $d = (a_3x^3+a_2x^2+a_1+a_0)\cdot x^2 = a_0x^2+a_1x^3+a_2x^4+a_3x^5 = a_0x^2+a_1x^3+a_2(x+1)+a_3(x^2+x) = a_2+(a_2+a_3)x+(a_0+a_3)x^2+a_1x^3$. Hence, we can write :

$$\begin{pmatrix} d_0\\ d_1\\ d_2\\ d_3 \end{pmatrix} = \begin{pmatrix} 0&0&1&0\\ 0&0&1&1\\ 1&0&0&1\\ 0&1&0&0 \end{pmatrix} \cdot \begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3 \end{pmatrix}. \tag{4.6}$$

Now, we have to make a quantum circuits that implements the matrix in (4.6) into the matrix in (4.5). In relation (4.5), we replace 1 with the identity matrix and x^2 with the matrix expressed in relation (4.6):

$$\begin{pmatrix} d_0\\ d_1\\ d_2\\ d_3\\ d_4\\ d_5\\ d_6\\ d_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&1&0\\ 0&1&0&0&0&0&1&1\\ 0&0&1&0&1&0&0&1\\ 0&0&0&1&0&1&0&0\\ 0&0&1&0&1&0&0&0\\ 0&0&1&1&0&1&0&0\\ 1&0&0&1&0&0&1&0\\ 0&1&0&0&0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3\\ a_4\\ a_5\\ a_6\\ a_7 \end{pmatrix}. \tag{4.7}$$

In order to implement matrix from the expression (4.7) into the quantum circuit we can use CNOT gates. However, it will require ancilla qubits and therefore we use CNOT synthetic [48] to construct a quntum circuit of mix columns.

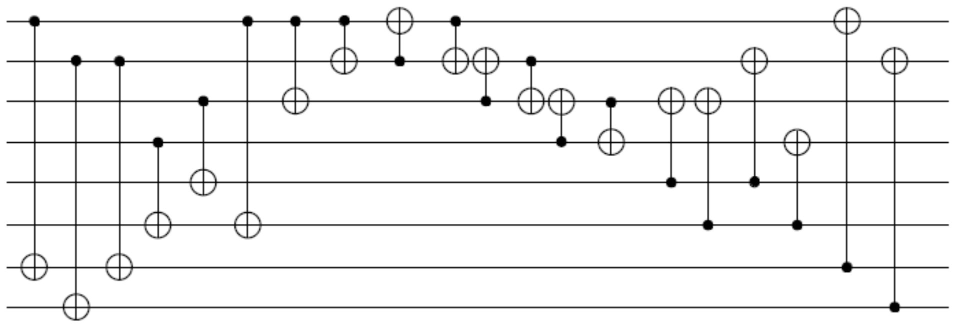

Figure 13: The Quantum Circuit of Columns Mixing of S-AES.

Key Expansion: As mentioned earlier S-AES has a 16-bit keysize, and the key is divided in the four nibbles N_0, N_1, N_2, N_3. Two nibbles make one byte B, i.e. $B_0 = N_0||N_1$ and $B_1 = N_3||N_4$, and the main key $k = (B_0, B_1)$. The S-AES has the three subkeys expanded from the main key: the whitening key obtained directly from the main key, the first encryption round subkey (B_2, B_3) and the second round subkey (B_4, B_5) are generated from k using ***Subkey generation algorithm*** [45], as explained in the following example.

Example 4.9. Let us assume that the input key is $k = 1010101010110001$. The first subkey k_1 is in the fact just the input key divided in the two bytes: $B_0 = 10101010$ and $B_1 = 10110001$.
The other subkeys are expanded as follows:

1. $B_i = B_{i-2} \oplus RCON(\frac{i}{2}) \oplus \text{SubNibRotNib}(B_{i-1})$ if $i \equiv 0 (\text{mod } 2)$, where $i = 2, 3, 4, 5$, $RCON(i) = (x^{i+2})||0000$, where x^{i+2} is the element of $GF(2^4)$ the function RotNib is to rotate the two nibbles and SubNib is introducet at (4.2).

2. if $i \not\equiv 0(\text{mod } 2)$, then $B_i = B_{i-2} \oplus B_{i-1}$.

Hence $B_2 = B_0 \oplus 10000000 \oplus \text{SubNib}(00011011)$. The nibble 0001 is equivalent to 1 in $GF(2^4)$ thus it is inverse to itself. Using (4.2) we obtain a new nibble given as 0111. The nibble 1011 is equivalent to $x^2 + x + 1$ and its inverse is $x^2 + x = 0110$. Using (4.2) we obtain a new nibble 1010. Thus, $B_2 = 10101010 \oplus 10000000 \oplus 01101010 = 1000000$. Moreover, $B_3 = B_1 \oplus B_2 = 10110001 \oplus 1000000 = 00110001$. In the similar way, we can determine the third subkey (B_4, B_5).

In Figure 14, we have that first 16 qubits represent the main key. Hence, $B_0 = N_0||N_1$ and $B_1 = N_2||N_3$. The first round subkey starts at the state $|\psi_2\rangle$, where RC is actually a round constant implemented for free. At the state $|\psi_3\rangle$ we have that the nibbles N_2 and N_3 are exchanged using (SN), and their outputs are at position N_5 and N_4 respectively, then they are Xored by the constant 10000000 by placing a single Pauli-X gates to the last qubit of the nibble N_5. In order to obtain first byte B_2 of the subkey nibbles N_0 and N_1 are XOR-ed with N_4 and N_5, respectively (the state $|\psi_4\rangle$.) The byte B_1 is XOR-ed with B_2 to produce B_3 which is XOR-ed with the second nibble of ciphertext (see $|\psi_5\rangle$). The process to obtain the second round subkey is similar to the first one. The difference is that one uses two Pauili-X gates implemented on qubits 0 and 1 of the nibble N_7. Note that $RCON(2) = (x^{2+2}||0000) = (x + 1||0000) = 00110000$.

4.1.1.2 Grover's attack on S-AES

Choose a pair (P, C), where P is a plaintext and C is the corresponding ciphertext. The main goal it to recover the secret key K. The description of Grover's attack to S-AES, as well as its design, we recall from [3]. Namely, in [3], M. Almazrooie et al.

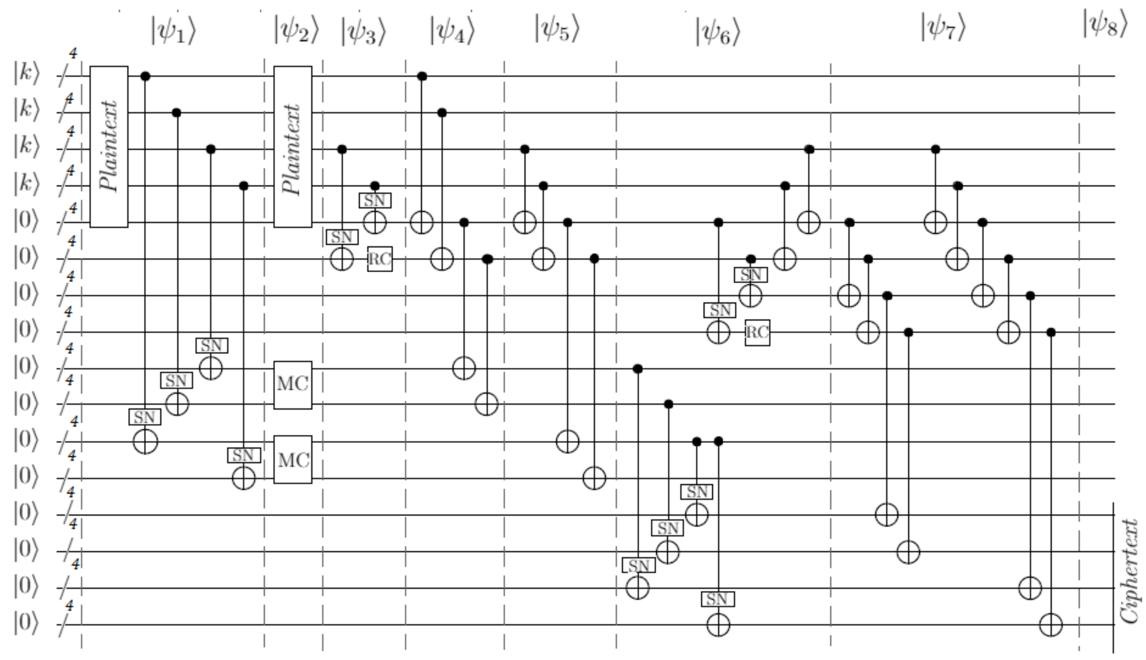

Figure 14: The Quantum Circuit of S-AES.

firstly defines the function f as

$$f(K) = \begin{cases} 1, & \text{if } SAES(K,P) = C, \\ 0, & \text{otherwise,} \end{cases}$$

and then defined the oracle function $\mathcal{O}_f|k\rangle = (-1)^{f(k)}|k\rangle$. The quantum attack consists the following steps:

1. Initializing 16 qubits to the state $|0\rangle$ and put the oracle qubit to the state $|0\rangle$.

$$|\psi_0\rangle = |0\rangle^{\otimes 16} \otimes |0\rangle.$$

2. Apply Pauli-X gates to the last qubit $X|0\rangle = |1\rangle$. After that one uses a Hadamard transform $H^{\otimes 17}$ to obtain a superposition of the key qubits and thus one obtains

$$|\psi_1\rangle = H^{\otimes 16}|0\rangle \otimes |H\rangle|1\rangle = \frac{1}{\sqrt{2^{16}}} \sum_{k \in \mathbb{F}_2^1 6} |k\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

3. In this step ciphertext C is obtained, applying quantum S-AES described in the previous section $|\psi_2\rangle = |C\rangle$. In this state, the oracle or the Black-box $\mathcal{O}_f$, is queried. Here, 63 additional qubits were used for the workspace. We omit it in the first two steps for simplicity.

4. After solution is being marked, the reversing S-AES is applied and results to plaintext P, i.e. $|\psi_3\rangle = |P\rangle$.

5. Finally Grover operator (the conditional phase shift) is applied to determine $|\psi_4\rangle \approx |K\rangle$.

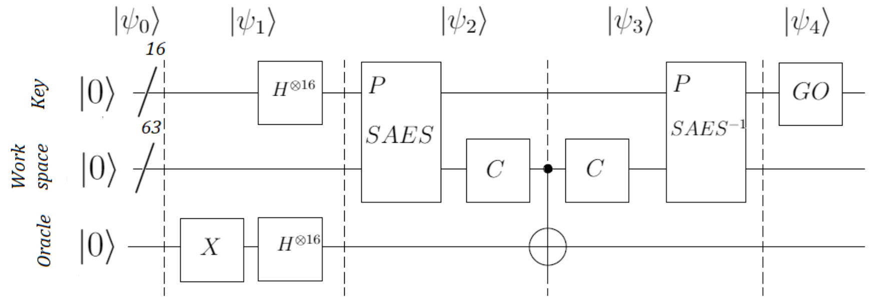

Figure 15: The Quantum Circuit of Grover Attack on the S-AES

Remark 4.10. Note that in [3], authors introduced experimental results which show that Grover's attack to S-AES recovers the secret key in $\frac{\pi}{4}2^{\frac{k}{2}}$, where k is the number of the qubits of the key.

4.1.2 Combining Simon's and Grover's Algorithm

4.1.2.1 The FX construction

The FX- construction was proposed by Killian and Rogaway in [34] as a generalization of the DESX scheme. This construction was build using an n-bit block cipher E with m-bit key K and two additional n-bit keys K_1 and K_2 such that

$$FX_{K,K_1,K_2}(P) = E_K(P \oplus K_1) \oplus K.$$

In [39], Leander and May introduced for a first time a combination of Grover's and Simon's algorithms, which has been applied to the FX-construction. The main result of [39] is given with the following theorem.

Theorem 4.11. *Let $f : \{0,1\}^m \times \{0,1\}^{3n} \to \{0,1\}^n$ with*

$$f_{k_0,k_1,k_2}(x) = g_{k_0}(x \oplus k_1) \oplus k_2, \tag{4.8}$$

where $g : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$. Given quantum oracle access to f and g, the tuple (k_0, k_1, k_2) can be computed with success probability at least $\frac{2}{5}$ using $m + 4n(n + \sqrt{n})$ qubits and $2^{\frac{m}{2}}O(m + n)$ oracle queries.

Lemma 4.12. *Let $\boldsymbol{u_1}, \ldots, \boldsymbol{u_{n-1}} \in \{0,1\}^n$ be linearly independent vectors. Then one can compute in time $O(n^3)$ a unique vector $\boldsymbol{v} \in \{0,1\}^n \backslash \{\boldsymbol{0}\}$ orthogonal to $\boldsymbol{u_i}$ for all $i \in \{1, \ldots, n-1\}$.*

The proof of Theorem 4.11 is based on the construction of a quantum algorithm which is a combination of Simon's and Grover's quantum results. As the proof is lengthy, we will focus only on its parts that are closely related to the mentioned construction.

Note that the function f actually represents the FX block cipher, where $f_{k_0,k_1,k_2}(x) = FX_{K,K_1,K_2}(P)$ and $g_{k_0}(x \oplus k_1) = E_K(P \oplus K_1)$. Let us now define the function $f' : \{0,1\}^m \otimes \{0,1\}^n \to \{0,1\}^n$ with

$$f'(k,x) = f_{k_0,k_1,k_2}(x) \oplus g_k(x).$$

For $k = k_0$, we have

$$f'(k_0,x) = f_{k_0,k_1,k_2}(x) \oplus g_{k_0}(x) = g_{k_0}(x \oplus k_1) \oplus k_2 \oplus g_{k_0}(x) = f'(k_0, x \oplus k_1),$$

and therefore the function $f'(k_0,x)$ is periodic with the period k_1 in its second components, which is equivalent to the property (3.1). Therefore, by applying Simon's algorithm we are able to find k_0 for which f' is a periodic function. In the case when $k_1 = 0^n$, $f'(k_0,x) = g_{k_0}(x \oplus 0^n) \oplus k_2 \oplus g_{k_0}(x) = g_{k_0}(x) \oplus k_2 \oplus g_{k_0}(x) = k_2$, hence f' is constant for trivial k_1.
By relation (4.8) we have that if k_0 is known, then one computes $k_2 = f_{k_0,k_1,k_2}(x) \oplus g_{k_0}(x)$ for any x.

The next step is to define the function $h : \{0,1\}^m \times \{0,1\}^{nl} \to \{0,1\}^{nl}$ as follows

$$h : (k, x_1, x_2, \ldots, x_l) \to f'(k,x_1)||f'(k,x_2)||\ldots||f'(k,x_l),$$

where $l = 2(n + \sqrt{n})$. By $\mathcal{O}_h$ we define the quantum oracle as

$$\mathcal{O}_h : |k, x_1, \ldots, x_l, 0, \ldots, 0\rangle \to |k, x_1, \ldots, x_l, h(k, x_1, \ldots, x_l)\rangle,$$

where $\mathbf{0}$ is the all quantum state of length n. Now we describe the quantum algorithm which is a parallelized version of Simon's algorithm, which uses $l \approx O(n)$ queries and $m + 2nl = m + 4n(n + \sqrt{n})$ qubits.

The quantum algorithm $\mathcal{A}$:

1. Prepare the initial state
$$|\psi_0\rangle = |0\rangle^{\otimes m+2nl}.$$

2. Apply the Hadamard transform $H^{\otimes(m+nl)}$ to obtain a state
$$|\psi\rangle = \frac{1}{\sqrt{2^{m+nl}}} \sum_{k\in\{0,1\}^m, x_1,\ldots,x_l\in\{0,1\}^n} |k\rangle|x_1\rangle\ldots|x_l\rangle|0\rangle^{\otimes nl}.$$

3. Call the oracle $\mathcal{O}_h$ and apply it to $|\psi_2\rangle$:
$$|\psi_3\rangle = \frac{1}{\sqrt{2^{m+nl}}} \sum_{k\in\{0,1\}^m, x_1,\ldots,x_l\in\{0,1\}^n} |k\rangle|x_1\rangle\ldots|x_l\rangle|h(k, x_1, \ldots, x_l)\rangle).$$

4. Apply now the Hadamard transform $H^{\otimes nl}$ to $|x_1\rangle \dots |x_l\rangle$ which gives the state

$$|\psi_5\rangle = \frac{1}{\sqrt{2^{m+2nl}}} \sum_{\substack{k\in\{0,1\}^m, z_1,\cdot z_l\in\{0,1\}^n \\ x_1,\dots,x_l\in\{0,1\}^n}} |k\rangle(-1)^{z_1\cdot x_1}|z_1\rangle \dots (-1)^{z_1\cdot x_l}|z_l\rangle|h(k, x_1, x_2, \dots x_l)\rangle.$$

Since we are combining Simon's and Grover's algorithms, we introduce the definition of the classifier $\mathcal{B}$ as follows:

Classifier $\mathcal{B}$: Let us define a function $\mathcal{B} : \{0,1\}^{m+nl} \to \{0,1\}$ that maps $(k, z_1, \dots, z_l) \to \{0,1\}$, which satisfies the following two tests:

1. **Test 1:** If the dimension of liner span of all $z_i \in \{0,1\}^n$ is not equal to $n-1$, we set $\mathcal{B}(k, z_1, \dots, z_l) = 0$. Otherwise, one applies Lemma (4.12) to compute a unique non-zero vector $k_1' \in \{0,1\}^n$.

2. **Test 2:** Let $m_i, m_i' \in \{0,1\}^m$ be random and distinct elements, and check the identities $y_i = f_{k_0,k_1,k_2}(m_i) + f_{k_0,k_1,k_2} = g(k_0, m_i + k_1) + g(k_0, m_i' + k_1), \forall i = 1, \dots, \lceil \frac{3m+nl}{n} \rceil$. If all identities hold, then $\mathcal{B}(k, z_1, \dots, z_l) = 1$, otherwise 0.

Classifier $\mathcal{B}$ partitions the state $|\psi_5\rangle = |\phi_1\rangle + |\phi_2\rangle$ in a good subspace $|\phi_1\rangle$ if both tests are satisfied, and a bad subspace $|\phi_2\rangle$ otherwise. Thus, we classify the state $|k\rangle|z_1\rangle \dots |z_l\rangle$ as good if and only if $\mathcal{B}(k, z_1, \dots, z_l) = 1$.

Lemma 4.13. *[39] If $k = k_0$ then test $\mathcal{B}$ outputs 1 by probability at least $\frac{1}{5}$. If $\mathcal{B}$ outputs 1, then $k_0 = k$ holds with the probability at least $1 - \frac{1}{2^{2m+nl-4}}$.*

Therefore, the initial probability p of producing a good state in $\mathcal{A}$ is equal to

$$\begin{aligned} p &= P_r[|k\rangle|z_1\rangle \dots |z_l\rangle \text{is good}] \\ &= P_r[k = k_0] \cdot P_r(\mathcal{B}(k, z_1, \dots, z_l) = 1 | k = k_0) \geq \frac{1}{5} \cdot \frac{1}{2^m}. \end{aligned} \tag{4.9}$$

The classifier $\mathcal{B}$ defines a unitary operator $S_{\mathcal{B}}$ that changes the signs of states as :

$$S_{\mathcal{B}} : |k\rangle|z_1\rangle \dots |z_l\rangle \to \begin{cases} -|k\rangle|z_1\rangle \dots |z_l\rangle, & \text{if } \mathcal{B}(k, z_1, \dots, z_l) = 1, \\ \ \ |k\rangle|z_1\rangle \dots |z_l\rangle, & \text{if } \mathcal{B}(k, z_1, \dots, z_l) = 0. \end{cases}$$

The application of Grover's algorithm is realized by applying the unitary operator $Q = \mathcal{A}S_0\mathcal{A}^{-1}S_{\mathcal{B}}$ to the initial state $|\psi_5\rangle = \mathcal{A}|0\rangle^{\otimes m+2nl}$, i.e. we compute $Q^K|\psi_5\rangle$ and measure system for k iterations. In order to determined probability of a good state the following theorem is used.

Theorem 4.14. *[13] Let $\mathcal{A}$ be any quantum algorithm on q qubits that use no measurement. Let $\mathcal{B} : \{0,1\}^q \to \{0,1\}$ be a function that classifies outcomes $\mathcal{A}$ as good or bad. Let $p > 0$ be the initial success probability that a measurement of $\mathcal{A}|0\rangle^{\otimes q}$ is good.*

Set $k = \lceil \frac{\pi}{4\theta} \rceil$, where θ is defined via $\sin^2(\theta) = p$. Moreover, define the unitary operator $Q = -\mathcal{A}S_0\mathcal{A}^{-1}s_{\mathcal{B}}$, where the operator $S_{\mathcal{B}}$ changes the sign of the good state

$$|x\rangle \to \begin{cases} -|x\rangle, & \text{if } \mathcal{B}(x) = 1, \\ \ \ |x\rangle, & \text{if } \mathcal{B}(x) = 0, \end{cases}$$

while S_0 changes the sign of the amplitude only for the zero state $|0\rangle^{\otimes q}$. Then, after the computation of $Q^k\mathcal{A}|0\rangle^{\otimes q}$, a measurement yields good with probability at least max$\{1-p, p\}$.

Initially, the angle between $|\psi_5\rangle$ and its bad subspace $|\phi_0\rangle$ is θ, where

$$\sin^2(\theta) = p = \langle\phi_1|\phi_1\rangle.$$

Therefore, $\theta = \arcsin(\sqrt{p}) \geq \arcsin(\frac{1}{\sqrt{5}} \cdot \frac{1}{\sqrt{2^m}})$, where the lower bound follows from (4.9). Every Grover iteration increase the angle θ to $(2k+1)\theta$ after $k = \lceil \frac{\pi}{4\arcsin\frac{1}{\sqrt{2^m}}} \rceil$. After k iterations a final measurement produces a good state with the probability $p_{good} = \sin^2((2k+1)\theta)$, since $\theta \geq \arcsin(\frac{1}{\sqrt{5}} \cdot \frac{1}{\sqrt{2^m}})$, and thus

$$p_{good} \geq \sin^2\Big(\frac{\pi}{2} \cdot \frac{\arcsin(5^{\frac{-1}{2}} \cdot 2^{\frac{-m}{2}})}{\arcsin(2^{\frac{-m}{2}})}\Big).$$

Since $\arcsin(x) \approx x$ for small x, then for $m \gg 0$ one can write $p_{good} \geq \sin^2(\frac{\pi}{2} \cdot \frac{(5^{\frac{-1}{2}} \cdot 2^{\frac{-m}{2}})}{2^{\frac{-m}{2}}}) = \sin^2(\frac{\pi}{2\sqrt{5}}) \approx 0.42$.

Notice that for $\theta = \frac{\pi}{2}$ we have that the resulting state is almost equal to the good state $|\phi_1\rangle$, and in that case one obtains a good state with a high probability after measurement.

4.1.2.2 SMS4 block cipher

The application of Simon's and Grover's algorithms has been demonstrated to 7/8-rounds of SMS4 block cipher in [31] by Hodžić and Knudsen. We firstly recall the structure of the main building elements presented in the round function of SMS4 cipher. In general, it is a 32 round unbalanced Feistel network. Let us define the linear function $L : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ as

$$L(u) = u \oplus (u \lll 2) \oplus (u \lll 10) \oplus (u \lll 18) \oplus (u \lll 24),$$

where " $\lll i$" denotes left rotation by i bits, and the function $F_r : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$

$$F_r(x) = L(S(x \oplus k_r)),$$

where $S : (\mathbb{F}_2^8)^4 \to (\mathbb{F}_2^8)^4$ is 8-byte function defined as

$$S(u \oplus k_r) = (\mathcal{S}(u_1 \oplus k_1^r), \dots, \mathcal{S}(u_4 \oplus k_4^r)),$$

where $u = (u_1, \dots, u_4) \in (\mathbb{F}_2^4)^8$ and $k^r = (k_1^r, \dots, k_4^r)$ and $\mathcal{S} : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is a nonlinear S-box function. Now, the round function $RF_r : (\mathbb{F}_2^{32})^4 \times \mathbb{F}_2^{32} \to (\mathbb{F}_2^{32})^4$ is defined by

$$RF_3(x_0, x_1, x_2, x_3, k_r) = (x_1, x_2, x_3, x_0 \oplus F_r(x_1 \oplus x_2 \oplus x_3)),$$

where $x_i \in \mathbb{F}_2^{32}$. The main idea in [31] is to consider the output of the first branch at

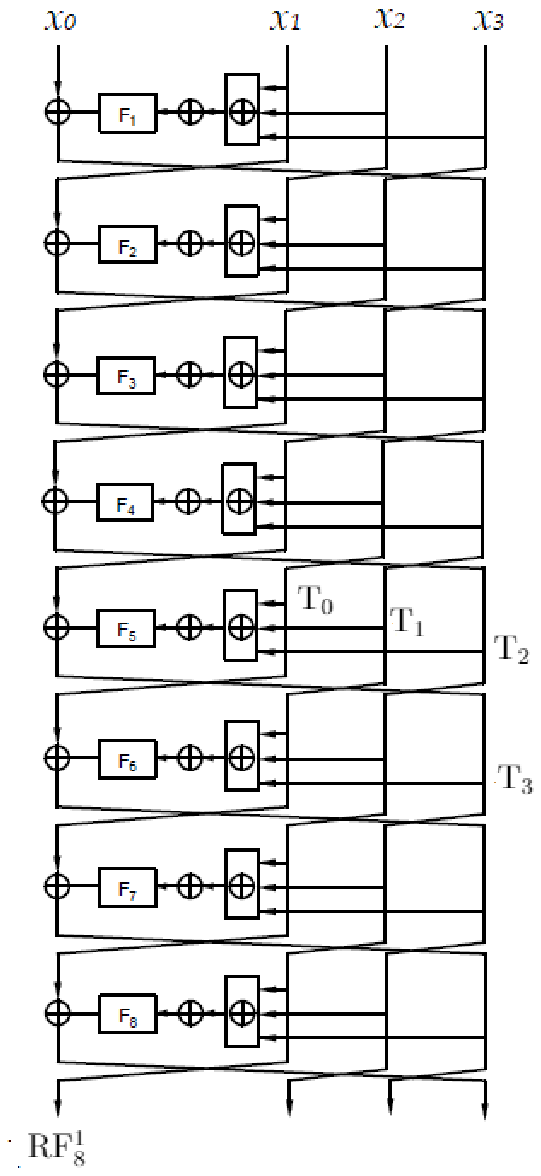

Figure 16: The eight round SMS4 block cipher

round 8, denoted by RF_8^1, of round function RF_8 and using it define the function for SMS4 block cipher, with similarly properties as the function f defined in the case of the FX-construction. In Figure 16, we can see that RF_8^1 has output given by

$$RF_8^1(\overline{x}) = T_0(\overline{x}) \oplus F_5(T_1(\overline{x}) \oplus T_2(\overline{x}) \oplus T_3(\overline{x})), \tag{4.10}$$

where $\overline{x} = (x_0, x_1, x_3)$ and

$$\begin{aligned} T_0(\overline{x}) &= x_0 \oplus F_1(x_1 \oplus x_2 \oplus x_3), \\ T_1(\overline{x}) &= x_1 \oplus F_2(x_2 \oplus x_3 \oplus T_0(\overline{x})), \\ T_2(\overline{x}) &= x_2 \oplus F_3(x_3 \oplus T_0(\overline{x}) \oplus T_1(\overline{x})) \\ T_3(\overline{x}) &= x_3 \oplus F_4(T_0(\overline{x}) \oplus T_1(\overline{x}) \oplus T_2(\overline{x})). \end{aligned}$$

Notice that that the term $T_3(\overline{x})$ in (4.10) does not have x_3 present together with $T_0(\overline{x})$ in the function F_4. Therefore instead $\overline{x} = (x_0, x_1, x_2, x_3)$ we write $\overline{x} = (x, \alpha, \alpha, \alpha) = (x, \Lambda)$, where $\Lambda = (\alpha, \alpha, \alpha)$ for some fixed $\alpha \in \mathbb{F}_2^{32}$. Furthermore, we have

$$\begin{aligned} T_1(\overline{x}) \oplus T_2(\overline{x}) \oplus T_3(\overline{x}) = \alpha &\oplus F_2(x \oplus F_1(\alpha)) \oplus F_3(x \oplus F_1(\alpha) \oplus F_2(x \oplus F_1(\alpha))) \\ &\oplus F_4(x \oplus F_1(\alpha) \oplus F_2(x \oplus F_1(\alpha))) \\ &\oplus F_3(x \oplus F_1(\alpha) \oplus F_2(x \oplus F_1(\alpha))) = \alpha \oplus D(x \oplus F_1(\alpha)), \end{aligned}$$

and thus the output of RF_8^1 can be written as

$$RF_8^1(x, \Lambda) = x \oplus F_1(\alpha) \oplus F_5(\alpha \oplus D(x \oplus F_1(\alpha)).$$

In order to apply the Simon-Grover algorithm, we introduce the function $f : \mathbb{F}_2^{2n+1} \to \mathbb{F}_2^n$ as follows:

$$f(y, b, x) = \begin{cases} g(y, x) = \alpha_0 \oplus A[(\phi(x, y)) \oplus RF_8^1(x, \Lambda_0)], & \text{if } \ b = 0, \\ h(y, x) = \alpha_1 \oplus A[(\phi(x, y)) \oplus RF_8^1(x, \Lambda_1)], & \text{if } \ b = 1, \end{cases}$$

where $\phi(x, y) = x \oplus L(S(\alpha_i \oplus y))$ and $\Lambda_i = \{\alpha_i, \alpha_i, \alpha_i\}$ for $i = 0, 1$. Function A represents the composition functions of inverse mapping of functions S and L respectively, i.e $A = (S^{-1} \circ L^{-1})$. It is not difficult to verify that for $y = k_1$ then $g(k_1, x) = h(k_1, x \oplus F_1(\alpha_0) \oplus F_1(\alpha_1))$, and thus the function $f(k_1, b, x)$ is a periodic function.

Denoting by $z = (b, x) \in \mathbb{F}_2 \to \mathbb{F}_2^{32}$, we define function $h_1 : \mathbb{F}_2^n \times (\mathbb{F}_2^{n+1})^l \to \mathbb{F}_2^{ln}$ by

$$h_1 : (y, z_1, \ldots, z_l) \to f(y, z_1) || \ldots || f(y, z_l).$$

The next step is to introduce the quantum oracle $\mathcal{O}_{h_1}$ and apply the quantum algorithm $\mathcal{A}$, test classifier $\mathcal{B}$ and realize Grover's algorithm $Q = \mathcal{A}S_0\mathcal{A}S_{\mathcal{B}}$. As this process is described in detail for FX constructions, we will state only the final result.

Hence, in [31] it was shown that Simon-Grover application to RF_8^1 requires $n + l(n+1) + ln$ qubits and $2^{\frac{n}{2}} \cdot O(n)$ oracle queries. Similarly like for FX-construction the key k_1 is extracted.

4.1.2.3 Quantum Key-recovery Attacks to 5-Round Feistel Structures

Based on the work of Leander and May, which we described in Subsection 4.1.2.1, Dong and Wang in [21] derived a similar attack on 5-round Feistel network. We consider the following function:

$$f(b, R_0) = F_2(k_2, R_0 \oplus F_1(k_1, \alpha_b)) = \alpha_b \oplus R_3 = \alpha_b \oplus F_4(k_4, F_5(k_5, R_5) \oplus L_5) \oplus R_5, \quad (4.11)$$

where $b \in \{0,1\}, \alpha_b \in \{0,1\}^{\frac{n}{2}}$ is an arbitrary constant and $\alpha_0 \neq \alpha_1$, $E(\alpha_b || R_0) = (L_5 || R_5)$. The following result describes the periodicity of f.

Lemma 4.15. *Let E be 5-round Feistel cipher, then for any $x = b||R_0 \in \{0,1\}^{\frac{n+2}{2}}$*

$$f(x) = f(x \oplus s) \iff s = 1||F_1(k_1, \alpha_0) \oplus F_1(k_1, \alpha_1).$$

By Lemma 4.15, we have that f has the period $s = 1||F_1(k_1, \alpha_0) \oplus F_1(k_1, \alpha_1)$, and thus satisfies the assumption (3.1).

Theorem 4.16. *Let $g : \{0,1\}^{\frac{n}{2}} \times \{0,1\}^{\frac{n}{2}} \times \{0,1\}^{\frac{n}{2}+1} \to \{0,1\}^{\frac{n}{2}}$ with*

$$(k_4, k_5, y) \to f(y) = f(b, R_0) = \alpha_b \oplus F_4(k_4, F_5(k_5, R_5) \oplus L_5) \oplus R_5,$$

where $y = b||R_0$, α_0, α_1 are two arbitrary constants, and $(L_5||R_5) = Enc(\alpha_b||R_0)$. Given quantum oracles of g and Enc, (k_4, k_5) and $F_1(k_1, \alpha_0) \oplus F_1(k_1, \alpha_1)$ could be computed using $n + (n+1)(n + 2 + 2\sqrt{n/2+1})$ qubits and approximately $2^{\frac{n}{2}}$ quantum queries.

In order to prove Theorem 4.16, the following function $h : \{0,1\}^n \times \{0,1\}^{(\frac{n}{2}+1^l)} \to \{0,1\}^{(\frac{n}{2})^l}$ is defined by

$$h : (k_4, k_5, y_1, \dots, y_l) \to g(k_4, k_5, y_1)|| \dots ||g(k_4, k_5, y_l),$$

and a quantum oracle $\mathcal{O}_h$ is defined as

$$\mathcal{O}_h : |k_4, k_5, y_1, \dots, y_l, 0, \dots, 0\rangle \to |k_4, k_5, y_1, \dots, y_l, h(k_4, k_5, y_1, \dots y_l)\rangle.$$

The next step is to apply the quantum algorithm $\mathcal{A}$, test classifier $\mathcal{B}$ and apply Grover's algorithm $Q = \mathcal{A}S_0\mathcal{A}S_{\mathcal{B}}$. This process has been described in detail for the FX construction, and thus the details are omitted here.

Remark 4.17. [21] The quantum key-recovery attacks, described in this section can be applied to 7-/8-/15-/31-/32-round Feistel structures with minor modifications. These results are summarized in Table 2.

Table 2: Summary of the key-recovery attacks on Feistel schemes in quantum chosen-plaintext attack settings [21].

Rounds	Trivial bound	Dong et al.
5	$2^{1.25n}$	$2^{0.5n}$
7	$2^{1.75n}$	2^{n}
8	2^{2n}	$2^{1.25n}$
15	$2^{3.75}$	2^{3n}
31	$2^{7.75n}$	2^{7n}
32	2^{8n}	$2^{7.25n}$

4.1.2.4 A quantum Attack on 30-round GOST Block Cipher

GOST [58] block cipher was developed in the 1970s by Soviet Union as an alternative to the American DES. It is a 32 round symmetric block cipher standardized by the Russian government with a block size of 64 bits, having a Feistel structure with 8 S-boxes and 256-bit keysize. The round function f consists of a key addition, eight 4×4 S-boxes and 11-bit left rotation. The key k of bit size 256 is divided into eight 32-bit words, i.e., $K = (k_0, k_1, k_2, \ldots, k_8), k_i \in \{0,1\}^{32}$. The GOST round function is shown in Figure 17, where $L_i||R_i$ is the input state of the i-th round function, such that L_i and R_i are the left and right branches of the i-th round function for $i = 0, 1, 2, \ldots 31$, and "$\lll$" represents a cyclic left rotation by j bits. Before describing the quantum

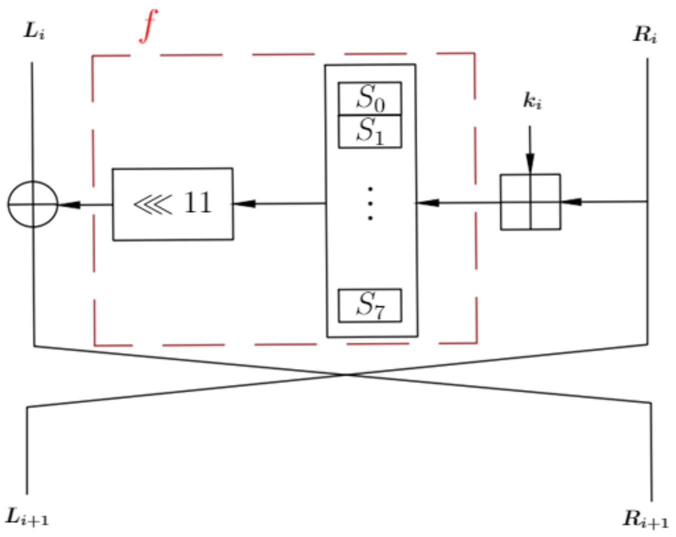

Figure 17: One round of the GOST block cipher.

attack presented in [21], let us recall some properties of the GOST block cipher.

- **Property 1:** [21] For a two round GOST holds the following. If we know $(L_0||R_0)$

and $(L_2||R_2)$, then $k_0 = S^{-1}((L_0 \oplus L_2) \ggg 11) - R_0)$, $k_1 = S^{-1}((R_0 \oplus R_2) \ggg 11) - L_2$.

- **Property 2 (*Reflection Property*):** [47] If the input state of the 25th round meets condition $L_{24} = R_{24}$, then the last 16-round of 32-round GOST acts as an identity by ignoring the last swap function, i.e., the input of 17th round is $(L_{16}||R_{16})$, and the output of 32th round is $(L_{32}||R_{32}) = (L_{16}||R_{16})$.

In [21], an attack on 30 rounds of the cipher has been provided, where the authors consider from the 3-rd round (as an input) to the 32-nd round as an output, i.e. the input is $L_2||R_2$ and the output is $L_{32}||R_{32}$. Similarly as the Feistel construction, one firstly applies Algorithm $\mathcal{A}$ that we described earlier, and then the combination of Simon's and Grover's algorithms. More precisely, we have the following steps.
Algorithm $\mathcal{A}$:

1. Prepare the initial 32×7-bit register
$$|\psi_0\rangle = |0\rangle^{\otimes 224}.$$

2. Apply the Hadamard transform $H^{\otimes 224}$ to obtain a state
$$|\psi_1\rangle = \frac{1}{2^{112}} \sum_{L_2,k_2,\ldots,x_7 \in \{0,1\}^{32}} |L_2\rangle|k_2\rangle \ldots |k_7\rangle,$$
where L_2 is the left half of the input of the 30-round GOST, the right half R_2 is a constant.

3. By Property 2, if $L_{24} = R_{24}$, the last 16-round is an identical transformation if the last swap function is ignored. Thus, given 2^{32} inputs $L_2||R_2$, we can expect that there is an $L_2||R_2$ that satisfies the condition $L_{24} = R_{24}$, i.e., $L_{16}||R_{16} = R_{32}||L_{32}$.

4. Once the right $L_2||R_2$ is obtained, one can guess $k_2, k_3, \ldots, k_7$ and then encrypts for round 3-8 to obtain the internal state $L_8||R_8$, decrypts $L_16||R_16$ for round 11-16 to obtain $L_{10}||R_{10}$. According to Property 1, k_0 and k_1 can be derived from $L_8||R_8$ and $L_{10}||R_{10}$.

To partition the space $|\psi_1\rangle$ into a good $|\phi_1\rangle$ and a bad $|\phi_0\rangle$ subspaces, a classifier $\mathcal{B} : \{0,1\}^{32\times 7} \to \{0,1\}$ that maps $(L_2, k_2, \ldots, k_7)$ to 0 or 1 is constructed as follows:

1. Derive $L_{32}||R_{32}$ for $L_2||R_2$ from the 30-round encryption oracle, note that R_2 is a randomly given constant.

2. Use $(k_2, k_3, \ldots, k_7)$, $L_2||R_2$ and $L_{32}||R_{32}$ to derive k_0, k_1 from Property 1.

3. Check the derived $(k_0, k_1, k_2, \ldots, k_7)$ by 5 plaintext ciphertext pairs using the 30-round encryption oracle. If the check is correct, output 1, otherwise 0.

The classifier $\mathcal{B}$ outputs good state if it satisfies the following two tests:

1. **Test 1:** $L_2||R_2$ satisfies Property 2. According to the FX construction, it is correct with probability 2^{-32}.

2. **Test 2:** $k_2, k_3, \dots k_7$ are the correct subkeys. It is correct with probability 2^{-192}.

Classifier $\mathcal{B}$ defines a unitary operator $S_{\mathcal{B}}$, which conditionally changes the sign of the quantum state ψ_1:

$$\begin{cases} -|L_2\rangle|k_2\rangle\dots|k_7\rangle, & \text{if } \mathcal{B}(L_2, k_2, \dots, k_7) = 1, \\ \ \ |L_2\rangle|k_2\rangle\dots|k_7\rangle, & \text{if } \mathcal{B}(L_2, k_2, \dots, k_7) = 0. \end{cases}$$

The application of Grover's algorithm is realized by applying the unitary operator $Q = \mathcal{A}S_0\mathcal{A}^{-\infty}S_{\mathcal{B}}$ to the initial state $\mathcal{A}|\psi_1\rangle$. Similarly to the FX construction, one can obtain that after 2^{112} Grover iterations Q, the angle between resulting state and the bad subspace is approximately $\frac{\pi}{2}$. The probability to obtain a good state is about $\sin^2(\frac{\pi}{2}) = 1$.

4.1.2.5 Quantum Attack on SoEM22

In Chapter 2 we defined and introduced Sum of Even-Mansour construction and described the application of Simon's algorithm to two variants (SoEM1 and SoEM2) of this construction. In this section, we consider a quantum attack on SoEM22 based on the combination of Simon's and Grover's algorithms. Let $g : \{0,1\}^n \otimes \{0,1\}^n \to \{0,1\}^n$ be a function $g(x,k) = P_1(x) \oplus p_2(X \oplus k)$, where P_1, P_2 are permutations in $\mathcal{P}_n$. Define the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as follows:

$$\begin{aligned} f(k,x) &= SOEM22(k_1,k_2,x) \oplus g(k,x) \\ &= P_1(x \oplus k_1) \oplus k_1 \oplus P_2(x \oplus k_2) \oplus k_2 \oplus P_1(x) \oplus P_2(x \oplus k). \end{aligned} \tag{4.12}$$

For the case that $k = k_2$, we have $f(k_2, x) = P(x \oplus k_1) \oplus k_1 \oplus k_2 \oplus P_1(x)$. It is not difficult to verify that $f(k_2, x)$ is a periodic function with the period k_1. Following the steps of the proof of Theorem 4.11, we assume that g behaves like a random function for every $k \in \{0,1\}^n$. For $k \neq k_2$, f is not a periodic function with high probability. Then, in the application of Grover's algorithm, we have the steps which are the same as for th FX construction (and thus the details are omitted). We recall the following theorem for SoEM22, which is equivalent to Theorem 4.11 for the FX construction.

Theorem 4.18. *[51] For any fixed $k \in \{0,1\}^n$, $g(k,x)$ is assumed to behave like a random function $\{0,1\}^n \to \{0,1\}^n$. With quantum oracle access to $SOEM22(k_1,k_2,x)$ and $g(k,x)$, there is a quantum attack against SoEM22 that recovers k_1 and k_2 with $O(n^2)$ qubits and $O(n \cdot 2^{\frac{n}{2}})$ queries.*

5 BERNSTEIN-VAZIRANI'S ALGORITHM

In this section we describe Bernstein-Vazirani's algorithm [6] and some of its applications. In general, this algorithm can be considered as an extension of the Deutsch–Jozsa algorithm [20]. In cryptanalysis of block ciphers, Bernstein-Vazirani's algorithm is used for the similar purpose as Simon's algorithm (in relation to linear structures of a given function), as well as in derivation of a quantum version of differential attacks. We now state Bernstein-Vazirani's problem, and later the algorithm which solves this problem is provided.

Problem 5.1. *Bernstein-Vazirani's Problem:* Let $f : \{0,1\}^n \to \{0,1\}$ be a function with the promise that for all input strings $x \in \{0,1\}^n$ it holds that

$$f(x) = x \cdot s, \tag{5.1}$$

where $s \in \{0,1\}^n$ is a secret string. The goal is to determine s.

Example 5.2. Let the function $f : \{0,1\}^2 \to \{0,1\}$ be defined by the following table of values

x	$f(x)$
00	0
01	0
10	1
11	1

Note that f satisfies the property (5.1). The goal is to find a secret string $s = s_1s_2$, where $s_1, s_2 \in \{0,1\}$. Since the function f satisfies (5.1), we have for all $x = (x_1, x_2) \in \{0,1\}^2$, it holds that $f(x) = x \cdot s = x_1s_1 + x_2s_2$. Let us consider two elements $x_1 = 01$ and $x_2 = 10$. From the table of values of f we have that $f(x_1) = f(01) = 0$ and $f(x_2) = f(10) = 1$. Thus, one obtains the following system of equations

$$\begin{aligned} f(01) &= 0s_1 + 1s_2 = 0, \\ f(10) &= 1s_1 + 0s_2 = 1. \end{aligned} \tag{5.2}$$

By solving the system (5.2) one deduces that $s_1 = 1$ and $s_2 = 0$, and thus $s = 10$.

In Example 5.2, one needs two queries to determine s in the classical way. Similarly, for $n > 2$ we can conclude that we need at least n queries of the function to find s, while only one query is required for the quantum computation, as shown by the following algorithm.

Bernstein-Vazirani's Algorithm [6]:

1. **Input:** Prepare the state:
$$|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle.$$

2. **Apply a Hadamard transform:** In this step we transform input $|\psi_0\rangle$ applying the Hadamard transform $H^{\otimes(n+1)}$. We can write it as follows:
$$|\psi_1\rangle = H^{\otimes n}|0\rangle^{\otimes n} \otimes H|1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

3. **Call the oracle or black-box $\mathcal{O}_f$:** Now we call the oracle $\mathcal{O}_f$, that is implemented as a unitary operation that preforms the transformation $\mathcal{O}_f|x\rangle = (-1)^{f(x)}|x\rangle$
$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

4. **Apply a Hadamard transform:** Here we omit the last qubit for the simplicity. Notice, if we aplly a Hadamard transform to the last qubit $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ we obtain state $|1\rangle$. In this step, the Hadamard transforms $H^{\otimes n}$ are applied to the first n qubits
$$|\psi_3\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x))} H^{\otimes n}|x\rangle = \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)\oplus y\cdot x}|y\rangle.$$
By (5.1), we have that $f(x) = x \cdot s$ and $x \cdot s \oplus y \cdot x = (s \oplus y) \cdot x$, and thus
$$|\psi_3\rangle = \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} (-1)^{(s\oplus y)\cdot x}|y\rangle.$$

5. **Measurement:** Now we consider
$$c = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{(s\oplus y)\cdot x}.$$
Clearly, for $s = y$ we have that $c = 1$. Therefore, the probability that $s = y$ is 1. Since the total probability must add up to 1, this means that all the other amplitudes must be zero.

As we mentioned earlier, we used only one query to find s and $O(n)$ gates. In general, Berstein-Vazirani's algorithm has found application in determining the linear structure of Boolean's functions, as well as in quantum differential analysis. So now we define the Walsh transform of Boolean's function, which actually represents the connection between this algorithm and its applications.

Definition 5.3. The *Walsh transform* of a function $f : \{0,1\}^n \to \{0,1\}$ is defined by

$$S_f(u) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus x \cdot u}. \tag{5.3}$$

Let us define the following two sets:

$$\mathcal{N}_f^0 = \{u \in \{0,1\}^n | S_f(u) = 0\}, \quad \mathcal{N}_f^1 = \{u \in \{0,1\}^n | S_f(u) \neq 0\}. \tag{5.4}$$

By (5.3), the state $|\psi_3\rangle$ of the BV algorithm can be written as follows

$$|\psi_3\rangle = \sum_{y \in \{0,1\}^n} S_f(y)|y\rangle.$$

Therefore, if we measure it in the computational basis we will find y with the probability $S_f^2(y)$, since $S_f(y)$ is an amplitude of the corresponding state. The probability of obtaining $y \in \mathcal{N}_f^0$ is always zero according to (5.4). Thus, we always determine the y such that $y \in \mathcal{N}_f^1$. This fact is very important for BV applications, which we will show in the following section.

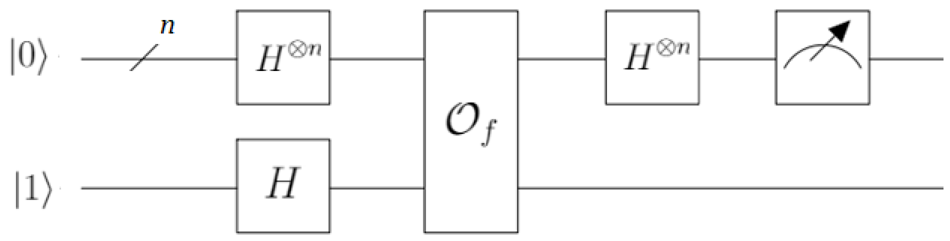

Figure 18: The quantum circuit of BV algorithm.

5.1 A QUANTUM ALGORITHM FOR FINDING LINEAR STRUCTURES OF BOOLEAN FUNCTIONS

There is no known efficient classical algorithm to determine the linear structure of Boolean functions with higher number of input variables. However, the quantum algorithm based on Bernstein-Vazirani's algorithm introduced in [29] can provide some information in an efficient way. Before we provide all details, we firstly list some important facts, properties, and definitions related to the linear structures of Boolean functions.

Definition 5.4. The vector $s \in \{0,1\}^n$ is a *linear structure* of a function $f : \{0,1\}^n \to \{0,1\}$ if

$$f(x \oplus s) \oplus f(x) = f(s) \oplus f(0), \quad \forall x \in \{0,1\}^n. \tag{5.5}$$

Let U_f denote the collection of the linear structure of $f(x)$, then

$$U_f = U_f^0 \cup U_f^1,$$

where

$$U_f^i = \{s \in \{0,1\}^n | f(x \oplus s) + f(x) = i, \ \ \forall x \in \{0,1\}^n\}, \quad i = 0, 1.$$

For any $s \in \{0,1\}^n$ and $i = 0, 1$ let

$$V_{f,s}^i = \{x \in \{0,1\}^n | f(x \oplus s) \oplus f(x) = i\}. \tag{5.6}$$

Definition 5.5. A vector $s \in \{0,1\}^n$ is called a *quasi linear structure* of a function $f \in \{0,1\}^n$ if

$$1 - \frac{|\{x \in \{0,1\}^n | f(x \oplus s) \oplus f(x) = i\}|}{2^n} < \sigma(n), \tag{5.7}$$

where $\sigma(n)$ is a negligible function.

The relation (5.7) means that (5.5) holds except for the negligible number of vectors x.

Definition 5.6. The *relative differential uniformity* of $f : \{0,1\}^n \to \{0,1\}$ is

$$\delta_f = \frac{1}{2^n} \max_{0 \neq a \in \{0,1\}^2} \max_{i \in \{0,1\}} |x \in \{0,1\}^n | f(x \oplus a) \oplus f(x) = i|. \tag{5.8}$$

Notice that $U_f \neq \{0\}$ if and only if $\delta_f = 1$.

Lemma 5.7. *[29]*

$$\mathcal{C}_f(s) = \sum_{x \in \{0,1\}^2} (-1)^{f(x)+f(x \oplus s)} = 2^n \left(\sum_{u \cdot s = 0} S_f^2(u) - \sum_{u \cdot s = 1} S_f^2(u) \right),$$

where $\mathcal{C}_f(s)$ is the correlation function of f.

Theorem 5.8. *[29] Let $f : \{0,1\}^n \to \{0,1\}$, the for all $s \in \{0,1\}^n$ and for all $i \in \{0,1\}$*

$$\sum_{u \cdot s = i} S_f^2(u) = \frac{|V_{f,s}^i|}{2^n} = \frac{|\{x \in \{0,1\}^n | f(x \oplus s) + f(x) = i\}|}{2^n}.$$

Proof. For $\mathcal{C}_f$ it holds:

$$\begin{aligned} \mathcal{C}_f(s) &= |\{x \in \{0,1\}^n | f(x \oplus s) + f(x) = 0\}| - |x \in \{0,1\}^n | f(x \oplus s) + f(x) = 1\}| \\ &= |V_{f,s}^0| - |V_{f,s}^1|. \end{aligned} \tag{5.9}$$

Therefore, from the Lemma 5.7 and relation (5.9), we have

$$2^n \left(\sum_{u \cdot s = 0} S_f^2(u) - \sum_{u \cdot s = 1} S_f^2(u) \right) = |V_{f,s}^0| - |V_{f,s}^1|,$$

which occurs in the following equation

$$\sum_{u\cdot s=0} S_f^2(u) - \sum_{u\cdot s=1} S_f^2(u) = \frac{|V_{f,s}^0|}{2^n} - \frac{|V_{f,s}^1|}{2^n}. \tag{5.10}$$

On the other hand, Parseval's relation gives (see [43])

$$\sum_{u\cdot s=0} S_f^2(u) + \sum_{u\cdot s=1} S_f^2(u) = \sum_{u\in\{0,1\}^n} S_f^2(u) = 1. \tag{5.11}$$

By (5.6), we have

$$|V_{f,s}^0| + |V_{f,s}^1| = 2^n. \tag{5.12}$$

Dividing, the relation (5.12) by 2^n we have

$$\frac{|V_{f,s}^0|}{2^n} + \frac{|V_{f,s}^1|}{2^n} = 1. \tag{5.13}$$

Combining relations (5.11) and (5.13), we obtain

$$\sum_{u\cdot s=0} S_f^2(u) + \sum_{u\cdot s=1} S_f^2(u) = \frac{|V_{f,s}^0|}{2^n} + \frac{|V_{f,s}^1|}{2^n}. \tag{5.14}$$

Now, by (5.10) and (5.14) we have

$$\sum_{u\cdot s=0} S_f^2(u) = \frac{|V_{f,s}^0|}{2^n}, \quad \sum_{u\cdot s=1} S_f^2(u) = \frac{|V_{f,s}^1|}{2^n}, \tag{5.15}$$

which is equivalent to the statement of the theorem. □

Theorem 5.8 is very important since it represents the relationship between linear structures and the Walsh spectrum of Boolean functions, as well as Bernstein-Vazirani's algorithm. In other words, if we run the BV algorithm, the probability of getting u such that $u \cdot s = i$ will be equal to $\frac{|V_{f,s}^i|}{2^n}$.

By the following lemma we have that if one can determine $\mathcal{N}_f^1$, then one can obtain U_f^i. Recall that by applying Bernstein-Vazirani's algorithm we can find a subset of $\mathcal{N}_f^1$.

Lemma 5.9. *Let a function* $f : \{0,1\}^n \to \{0,1\}$, *be given, then* $\forall i \in \{0,1\}$, *then the set* U_f^i *is equal to*

$$U_f^i = \{s \in \{0,1\}^n | u \cdot s = i, \ \forall u \in \mathcal{N}_f^1\}.$$

Before we introduce the quantum algorithm for finding a linear structure of the Boolean function f, which is provided in [29], we recall the following problem.

Problem 5.10. For a given function $f : \{0,1\}^n \to \{0,1\}$, decide whether a function has non-zero linear structures or not.

Algorithm 1: A quantum algorithm for finding a linear structures of the Boolean function f [29].

```
   Input: The Boolean function $f : \{0,1\}^n \to \{0,1\}$
   Output: Sets $S^0$ and $S^1$
1  Initialize $U := X$ and $p := p(n)$
2  for $r$ many times
3      Run the Bernstein-Vazirani algorithm to the function $f$ for $n+1$ times to
        get $n+1$ vectors $u_1, u_2, \dots, u_{n+1} \in \mathcal{N}_f^1$
4      Put $U := U \cup \{u_1, \dots, u_{n+1}\}$
5      Solve the equation $x \cdot U = i$ to get the solution $S^i$
6      if $S^0 = \{0\}$ and $S^1$=$X$ then
7          output no and stop
8  Report $f$ has quasi linear structures and output $S^0$ and $S^1$
```

For an arbitrary polynomial $p(n)$, and an empty set X is an empty set, Algorithm 1 solves the Problem 5.10. The complexity of this algorithm is described with Theorem 5.12 below.

Remark 5.11. By Theorem 5.9 it must be the case that $U_f^i \subseteq S^i$, but the reverse does not necessarily hold. Clearly, if $S^0 = \{0\}$ and $S^1 = X$ (**Step 6**), then s must be 0, i.e. f has no non-zero linear structure.

In order to find out whether a given Boolean function f has non-zero linear structures or not, we recall the following results from [29].

Theorem 5.12. *If there exist the relative differential uniformity δ_f such that $\delta_f < 1$, an average of $O(\frac{n+1}{1-\delta_f})$ times running the Bernstein-Vazirani algorithm in Algorithm 1 (i.e $r = O(\frac{1}{1-\delta})$) will give an answer that f has no non-zero linear structure.*

Proof. Assume that there exist $\delta_f < 1$. Then there are $\zeta \in \{0,1\}^n$ and $\eta \in \{0,1\}$ such that

$$\begin{aligned}\delta_f &= \frac{1}{2^n}|\{x \in \{0,1\}^n | f(x \oplus \zeta) \oplus f(x) = \eta\}| \\ &= \frac{1}{2^n} \max_{0 \neq a \in \{0,1\}^2} \max_{i \in \{0,1\}} |\{x \in \{0,1\}^n | f(x \oplus s) \oplus f(x) = i\}| = \delta < 1.\end{aligned}$$

Then,

$$\begin{aligned}&\frac{1}{2^n}|\{x \in \{0,1\}^n | f(x \oplus \zeta) \oplus f(x) = \overline{\eta}\}| \\ &= \frac{1}{2^n} \min_{0 \neq a \in \{0,1\}^2} \min_{i \in \{0,1\}} |\{x \in \{0,1\}^n | f(x \oplus s) \oplus f(x) = i\}| = 1 - \delta > 0.\end{aligned}$$

Thus, for any $0 \neq a \in \{0,1\}^n$ and for any $i \in \{0,1\}$, it holds that

$$0 < 1-\delta \leq \frac{1}{2^n}|\{x \in \{0,1\}^n | f(x \oplus a) \oplus f(x) = i\}| \leq \delta < 1.$$

Let us denote by $C_{s,i}$ the set of all vectors $u \in \{0,1\}^n$ such that $u \cdot s = i$ i.e. $C_{s,i} = \{u \in \{0,1\}^n | u \cdot s = i\}$. Assume that after $r \approx m$ times ruining of BV algorithm we have given the set U. The probability that $U \subseteq C_{s,i}$ is at most

$$\delta^m = ((1-(1-\delta)))^m \leq e^{-m(1-\delta)}.$$

If $m = \frac{1}{\delta-1}$, then $\delta^m \leq e^{-1}$. Particularly, if $m > \frac{c}{1-\delta}$ it implies $\delta^m < \frac{1}{e^c}$, where $c > 1$ is a constant. Therefore, the probability that U is not a subset of $C_{s,i}$ is at least

$$1-\delta^m > 1-\frac{1}{e^c}.$$

Suppose now, after some time we found k linearly independent vectors $u_1, u_2, \dots, u_k$ such that $a \in \{0,1\}^2$ is a solution of the equation $u_i \cdot x = 0$ and $0 < k < n$. Then after another expected number of order $O(\frac{1}{\delta-1})$ we get u_{k+1} such that $u_{k+1} \cdot s = 1$. u_{k+1} must be linearly independent with u_i, $0 \leq i \leq k$, since $u_i \cdot s = 0$ (s is solution of $u_i \cdot x = 0$). Repeating the process we can find n linearly independent vectors $u_1, u_2, \dots, u_n$, through an expected number of order $O(\frac{n}{1-\delta})$ measures. Thus we conclude $S^0 = U_0^f = \{0\}$. Since there exists exactly one possible solution b of the equations $u_j \cdot b = 1$, another expected number of order $O(\frac{1}{1-\delta})$ measures will give a u_{n+1}, such that $u_{n+1} \cdot b = 0$ thus it means $U_f^1 = S^1 = X$. Thus we need $O(\frac{n+1}{1-\delta_f})$ time complexity for running Bernstein-Vazirani's algorithm to give an answer that f has no non-zero linear structure. □

Another result (given in [29]), plays an important role in determining the running time of Algorithm 1, is given as follows.

Theorem 5.13. *[30] Given an oracle access to $f : \{0,1\}^n \to \{0,1\}$, Algorithm 1 gives an answer that f has no non-linear structure or outputs vector sets S^0 and S^1. Assuming one has run the Bernstein–Vazirani algorithm $p = r(n+1)$ times, then for every $s \in \{S^i\}$, $(i = 0,1)$, and for every ϵ, such that $0 < \epsilon < 1$ holds*

$$P_r\Big(1 - \frac{|\{x \in \{0,1\}^n | f(x \oplus s) \oplus f(x) = i\}|}{2^n} < \epsilon\Big) > 1 - e^{-2p\epsilon^2}, \tag{5.16}$$

where $P_r(E)$ is the probability of the event E happens.

Let us consider some special cases of δ. Let us assume that $p(n)$ is a polynomial function of n, such that $\delta < 1 - \frac{1}{p(n)}$. Thus, $\frac{n+1}{1-\delta} < \frac{n+1}{1-1+\frac{1}{p(n)}} = (n+1)p(n)$. By Theorem 5.12 we have that the complexity of running Algorithm 1 to obtain an answer that f has no non-zero structure is $O(\frac{n+1}{1-\delta}) < O((n+1)p(n))$. The other special case is when $\delta = \frac{1}{2}$, in which Algorithm 1 has $O(2(n+1)) = O(2n)$ time complexity.

Let us assume $e(n)$ is an exponential function of n and $1 > \delta > 1 - \frac{1}{e(n)}$. By Theorems 5.12 and 5.13, we need at least $O(e(n))$ times to run the algorithm. Special case is when $\delta = 2^{n-1}$, in which case we have $O(n2^{n-1})$ steps. All this cases are considered in [29].

The previous discussion shows that Algorithm 1, used to find the linear structures of Boolean functions is a polynomial-time algorithm.

5.2 THE QUANTUM DIFFERENTIAL CRYPTOANALYSIS

Differential cryptanalysis, proposed by Biham and Shamir in 1990 [7], is one of the most popular cryptanalyitic methods for block ciphers. In the following text, we briefly describe the main concept of differential cryptanalysis.

Let us consider a function $f : \{0,1\}^n \to \{0,1\}^m$ with input $x = (x_0, x_1, \ldots, x_{n-1})$ and output $y = (y_0, y_1, \ldots, y_{m-1})$. Let x' and x'' be two inputs such that $f(x') = y'$ and $f(x'') = y''$. The input and output differences are given by

$$\Delta x = x' \oplus x'' = (\Delta x_0, x_1 \ldots, \Delta x_{n-1}),$$

and

$$\Delta y = y' \oplus y'' = (\Delta y_0, y_1 \ldots, \Delta y_{m-1}),$$

respectively, where $\Delta x_i = x'_i \oplus x''_i$ and $\Delta y_i = y'_i \oplus y''_i$. The pair $(\Delta x, \Delta y)$ is called a *differential.*

A *differential characteristic* is composed of input and output differences, where the input difference to one round is determined by the output difference of the last round. Based on their previous work, L. Hongwei et al. [30] described two methods based on BV algorithm to derive a quantum version of differential cryptanalysis. In our work we introduce the first one.

Let us assume that there is a block cipher with block length $k = ln$ i.e the its plaintexts and the ciphertexts are of the length k. Suppose that every S-box function $F = (F_1, F_2, \ldots, F_m)$, where $F_i : \{0,1\}^n \to \{0,1\}$, for each $i \in \{1, \ldots, m\}$, maps $\{0,1\}^n \to \{0,1\}^m$. Steps of the first part of the quantum algorithm, which proposes differential cryptanalysis, are given by Algorithm 2. Firstly, one has to analyse obtained outputs S_i^j (where $j = 0, 1$) in order to get some high probability differentials of a S-Box function F. Now, assume that $p(n) = cn$ for some constant $c \geq 2$. Consequently, every vector $s \in S_i^j$ satisfies

$$\frac{|\{x \in \{0,1\}^n | f_i(x \oplus s) + f_i(x) = j\}|}{2^n} > \frac{1}{2}$$

with high probability, according to the Theorem 5.13. This means that for any vector $s \in S_i^j$, (s, i) is a differential of F_i with the probability grater than $\frac{1}{2}$. If most of

Algorithm 2: The first algorithm for the quantum differential cryptoanalysis [30]

```
Input: An S-Box $F = \{F_1, \dots F_m\}$.
Output: Some high probability differentials of each $F_i$, $i = 1, \dots, m$
1 Prepare $U$ and $W$ as a empty sets.
2 for $i = 1, \dots, m$
3     for $p = 1, \dots, p(n)$
4         Run the BV algorithm, and get an $m$-bit output $u$
5         Let $U = U \cup \{u\}$
6     Solve the equations $UX = 0$ and $UX = 1$ zo get sets $S_i^0$ and $S_i^1$,
       respectively
7     Output $S_i^0$ and $S_i^1$
```

the S_i^j consists of many vectors, then one should take $p(n)$ to be larger (for example, $p(n) \approx n^2$). The purpose of this is to prevent the set S_i^j from having too many elements. On the other hand, Algorithm 3 (given below) can be utilized to find differentials of F with high probability. The outputs of this algorithm are vectors of the form $(s, j_1 j_2 \dots j_m)$ or $(0, 0)$, and they can used to construct differential characteristics.

Algorithm 3: The second quantum algorithm for finding differentials [30].

```
Input: $S_i^j$, $i = 1, 2 \dots, m, j = 0, 1$
Output: Differentials of $f$ with high probability.
1 for each $S_1^{j_1}, j_1 = 0, 1$
2     for $j = 1, \dots, n$
3         for $j_i = 0, 1$
4             if $s \in S_i^{j_i}$ then
5                 $(x_s, y_s) = (s, j_1 \dots j_i)$
6         else if $s \notin S_i^0$ and $a \notin S_i^1$ then
7             $(x_s, y_s) = (0, 0))$
8 Output $(x_a, y_a)$
```

5.2.1 The efficiency of the quantum differential algorithm

In order to evaluate an S-Box function f, the running time of the BV algorithm is $mp(n) = p$, where the time to solve the system of linear equations $UX = 0$ and $UX = 1$ is $mq(n) = mn$, with $q = q(n)$ is a polynomial of n. Hence, the total time of the first algorithm is $m(p+q)$.

Let us consider the running time of the second algorithm, which has been derived in [30]. Obviously the upper bound of the time complexity is $O(2^m)$. Now we want to see the success probability.

Theorem 5.14. *[30] If $(s, j_1, j_2, \dots, j_n)$ is obtained by running Algorithm 3, then for any* $0 < \epsilon < 1$

$$P_r\Big(1 - \frac{|\{x \in \{0,1\}^n | f(x \oplus a) \oplus f(x) = j_1 \dots j_m\}|}{2^n} < m\epsilon\Big) > (1 - e^{-2p\epsilon^2})^m. \tag{5.17}$$

Proof. By Theorem 5.13, the probability of

$$\frac{|\{x \in \{0,1\}^n | f_i(x \oplus a) \oplus f_i(x) = j_i\}|}{2^n} > 1 - \epsilon \tag{5.18}$$

is greater than $1 - e^{-2p\epsilon^2}$. If the equation holds in 3.3 then the number of x that satisfies the above equation for two distinct i is at least $2^n(2(1-\epsilon)-1) = 2^n(1-2\epsilon)$. Similarly for three distinct i, the number of x satisfies equality in (5.18) is at least $2^n((1-2\epsilon)+(1-\epsilon)-1) = 2^n(1-3\epsilon)$. By induction of m we can prove that for all $i = 1, \dots, m$ holds that the number of x satisfying equality in (5.18) is $2^n(1-m\epsilon)$. Therefore, the probability that

$$1 - \frac{|\{x \in \{0,1\}^n | f(x \oplus a) + f(x) = j_1 \dots j_m|}{2^n} < m\epsilon$$

is greater than $(1 - e^{-2p\epsilon^2})^m$. □

Suppose now, $\epsilon = \frac{1}{c_1 m}$, where $c_1 \geq 2$ is a constant, $p = \frac{c_2}{\epsilon^2}$, where $c_2 \geq 1 + \frac{\ln m}{2}$ is also a constant, then from (5.17) we have

$$(1 - e^{-2p\epsilon^2})^m \geq (1 - e^{-2p\frac{c_2}{p}})^m \geq 1 - ne^{-2c_2} \geq 1 - ne^{-2(1+\frac{\ln n}{2})} = 1 - \frac{1}{e^2}.$$

Therefore, after a total time of $m(p+q) + O(2^n)$, where $p = \max\{p(m), c_2 c_1^2 n^2\}$, we obtain a set of non-zero differentials that consists of vectors of the form $(a, j_1 \dots j_m)$ which satisfy

$$P_r\Big(1 - \frac{|\{x \in \{0,1\}^n | f(x \oplus a) + f(x) = i\}|}{2^n} < \frac{1}{c_1}\epsilon\Big) > 1 - \frac{1}{e^2}. \tag{5.19}$$

Since the classical algorithm needs 2^{m+n} steps, and as the S-Box used in the block ciphers are not large, i. e. m and n are small, then 2^{m+n} is also small and therefore this quantum algorithm essentially does not show any higher speed compared to the classical one.

5.3 APPLICATIONS OF BERSTEIN-VAZIRANI's ALGORITHM

5.3.1 Application to a 3-round Feistel network

In Section 3.1.1 we defined the Feistel network and described applications of Simon's algorithm. Now, we introduce an application of Bernstein-Vazirani algorithm to a Feistel network [56]. We start by defining the function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ as follows

$$f(a||b) = \begin{cases} P_2(b \oplus P_1(\alpha_0)), & \text{if } a = 0, \\ P_2(b \oplus P_1(\alpha_1)), & \text{if } a = 1, \end{cases} \tag{5.20}$$

where $a \in \{0,1\}$, $b \in \{0,1\}^n$ and $\alpha_0, \alpha_1\{0,1\}^n$ are distinct arbitrary constants. The next step is to construct the oracle function $\mathcal{O}_f^{\alpha_0,\alpha_1}$:

$$\mathcal{O}_f|a\rangle|b\rangle = |a\rangle|y \oplus \alpha_a\rangle$$

Lemma 5.15. *For every $x = (a||b) \in \{0,1\}^{n+1}$ and $y = (a'||b') \in \{0,1\}^{n+1}$ holds*

$$f(x) = f(y) \iff a = a \oplus 1 \text{ and } b' = b \oplus z,$$

where $z = P_1(\alpha_0) \oplus P_1(\alpha_1)$.

The proof of Lemma 5.15 is similar to the proof of Lemma 3.8. We note that for the function f (by Definition 5.5), the vector $s = (1||z) = (1||P_1(\alpha_0) \oplus P_2(\alpha_1))$ is a non-zero linear structure, where $(1||z) \in \mathcal{O}_f^{(0,0\dots,0)}$ can be determined by Algorithm 1. Now, suppose that the access to the quantum oracle, which computes three-rounds function of the Feistel network or a random permutation over $\{0,1\}^{2n}$, is given. Let $p(n) = n + 1$ be an arbitrary polynomial function of n and initialize the set U as an empty set. A quantum distinguisher for 3-round Feistel network, based on Bernsein-Vzirani's algorithm, is given by Algorithm 4 (given below). The following two results show the validity of Algorithm 4.

Lemma 5.16. *[56] Let $f = (f_1, f_2, \dots, f_n)$ be a function defined as (5.20). Then for all $i = 1, \dots, n$, it holds*

$$\delta_{f_i}(1||z) = \frac{1}{2^{n+1}} \max_{(s_1,s_2)\in\{0,1\}^n+1} |\{(b||x) \in \{0,1\}^{n+1} | f_i(b||x) = f(b \oplus s_1||x \oplus s_2)\}| \leq \frac{2}{3}$$

holds except a negligible probability. Here, $\delta_{f_i}(1||\alpha)$ is still a random variable since f_i is determined by random functions P_1 and P_2 and $(s_1, s_2) \notin \{(0,\dots,0),(1||\alpha)\}$.

Theorem 5.17. *[56] Algorithm 4 successfully distinguishes the 3-round Feistel function from a random permutation except a negligible probability.*

Algorithm 4: [56] The quantum algorithm for distinguishing 3- Feistel network.

```
   Input: Arbitrary distinct constants $\alpha_0, \alpha_1 \in \{0,1\}^n$. The function
          $f = (f_1, f_2, \dots, f_n)$ defined by (5.20).
1  for  $j = 1, \dots n$
2      for $p = 1, \dots, p(n)$
3          Apply BV algorithm to $f_j$ to get an $u \in \mathcal{N}^1_{f_j}$
4          Set $U = U \cup \{u\}$
5      Solve the system of linear equations $\{x \cdot u = 0 | u \in U\}$ to get the solution
        set $S^j_0$.
6      if $S^0_j \subseteq \{(0, 0, \dots, 0)\}$ then
7          Output NO and stop the process.
8      else
9          Let $U = \emptyset$
10 if $S^0_1 \cap \dots \cap S^0_{n+1} \subseteq \{(0, \dots, 0)\}$ then
11     Output NO and stop.
12 else
13     Choose an arbitrary nonzero vector $a \in S^0_1 \cap \dots \cap S^0_{n+1}$ and choose an
        $(n+1)$-bit string $v$ uniformly at random. Let $v' = v \oplus a$. Make classical
        queries for $v$ and $v'$.
14 if $f(v) = f(v')$. then
15     Output YES
16 else
17     Output NO
```

Proof. (Theorem 5.17) If the given oracle computes a random permutation, the vector a obtained after running Algorithm 4 is random (if exists). Therefore, since we have 2^n elements, the probability to obtain a such that $v' = v \oplus a$ is $\frac{1}{2^n}$. Hence, the probability that $f(v) = f(v')$ is $\frac{1}{2^n}$. On the other hand, if the given oracle computes the 3-round Feistel function, then according to Lemma 5.16 we have that

$$\delta_{f_i}(1||z) = \frac{1}{2^{n+1}} \max_{(s_1,s_2)\in\{0,1\}^{n+1}} |\{(b||x) \in \{0,1\}^{n+1} | f_i(b||x) = f(b \oplus s_1 || x \oplus s_2)\}| \leq \frac{2}{3}$$

holds with overwhelming probability [1]. According to Theorem 2 in [56] we have

$$P_r[a \neq (1||z)] \leq \Big(\frac{2}{3}\Big)^{n+1},$$

[1]We say that an event E holds with overwhelming probability if, for every fixed $a > 0$, it holds with probability $1 - O_a(n^{-a})$, i.e $P_r(E) \geq 1 - C_a n^{-a}$ for some C_a independent of n.

and thus the probability that $f(v) \neq f(v')$ is the most $\left(\frac{2}{3}\right)^{n+1}$. □

Since Algorithm 4 requires $n(n+1)$ times for quantum oracle and classical oracle for 2 times, it has complexity $O(n^2)$ [56]. In comparison to Simon's quantum attack on the Feistel network described in Section 3.1.1, which requires $O(n)$ times complexity, this algorithm appears to be less efficient (since it requires more time).

5.3.2 Application to Even-Mansour construction

Recall that in Section 3.1.2 we introduced Even Mansour's cipher and described an application of Simon's attack to it. Recall that the encryption function E_k is defined as

$$E_k(p) = P(k_1 \oplus p) \oplus k_2,$$

where $k = (k_1, k_2)$ is a secret key and $P : \{0,1\}^n \to \{0,1\}^n$ is a public permutation. In [56], authors proposed a new quantum algorithm for recovering the key k, but they used Bernstein-Vazirani's algorithm in the place of Simon's. In order to construct this quantum attack, we define a function $f : \{0,1\}^n \to \{0,1\}^n$ as

$$f(x) = E_k(x) \oplus P(x) = P(x \oplus k_1) \oplus P(x).$$

If an oracle access of E_k is given, one can construct the quantum oracle function $\mathcal{O}_f$. Recall from Section 3.1.2 that for $y = x \oplus k_1$ $f(x) = f(y)$, i.e period of function f is k_1. Similarly, since we can write $f(x \oplus k_1) \oplus f(k_1) = f(k_1) \oplus f(0)$, it implies that k_1 is a linear structure of function f. Hence, by applying Algorithm 1, we are able to obtain k_1, while on the other hand Algorithm 5 (introduced in [56], which is given below) can determine k_1 with an overwhelming probability. The two following results justifies the validity of Algorithm 5.

Lemma 5.18. *Let $f = (f_1, f_2, \ldots, f_n)$ be an arbitrary function. Applying Algorithm 5 with n^2 queries ($p(n) = n$) to f gives output NO or some vector. It holds, if*

$$\delta'_f \leq p_0 < 1$$

and f has no linear structure, then Algorithm 5 returns NO with probability grater than $1 - p_0^n$.

Lemma 5.18 is part of Theorem 3 in [56], where it was assumed, that cn^2 queries is required, where c is a constant, but for our purpose we set $c = 1$.

Theorem 5.19. *Applying Algorithm 5 with n^2 ($p(n) = n$) queries to f gives a key k_1, except for negligible probability.*

Proof. The result of Lemma 5.16 can be modified for this purpose such that

$$\delta_{f_i}(k_1) = \frac{1}{2^{n+1}} \max_{s\in\{0,1\}^n} |\{x \in \{0,1\}^n | f_i(x) = f_i(x \oplus s)\}| \leq \frac{2}{3} \tag{5.21}$$

and $s \notin \{((0,\dots,0)), k_1\}$ holds except a negligible probability. Here, $\delta_{f_i}(k_1)$ is still a random variable since f_j is determined by random functions P_1, P_2. For

$$\delta'_f = \frac{1}{2^n} \max_{a\in\{0,1\}^2, a\notin U_f^1} \max_{i\in\{0,1\}^2} |\{x \in \{0,1\}^2 | f(x\oplus a) \oplus f(x) = i\}|,$$

we have that $\delta'_f = \max_j \delta'_{f_j} \leq \frac{2}{3}$ holds except for a negligible probability. Then according to Lemma 5.18, Algorithm 5 recovers k_1 with the probability greater than $1 - (\frac{2}{3})^n$. □

Let us assume the oracle access to $f = (f_1, \dots, f_n)$ is given and $p(n)$ is an arbitrary polynomial of n. Then the quantum algorithm for distinguishing Even-Mansour construction is described in Algorithm 5.

Algorithm 5: [56] The quantum algorithm for distinguishing EM scheme.

```
1  for  $i = 1, \dots n$
2      for $p = 1, \dots, p(n)$
3          Apply BV algorithm on $f_j$ to get an $u \in \mathcal{N}_{f_j}^1$
4          Put $U = U \cup \{u\}$
5      Solve the system of linear equations $\{x \cdot u = 0 | u \in U\}$ to get the solution
       $S_0^j$,
6      if $S_j^0 \subseteq \{(0,0,\dots,0)\}$ then
7          Output NO and stop the process.
8      else
9          Let $U = \emptyset$.
10 if $S_1^0 \cap \dots \cap S_{n+1}^0 \subseteq \{(0,\dots,0)\}$ then
11     Output NO and stop.
12 else
13     Choose an arbitrary nonzero vector $a \in S_1^0 \cap \dots \cap S_n^0$ and output $a$.
```

Time complexity of Algorithm 5 is $O(n^2)$, which is higher than the polynomial complexity $O(n)$ of Simon's algorithm.

6 KUPERBERG'S ALGORITHMS

In this section we recall a few variants of Kuperberg's algorithm (other versions can be found in [36]), which has quite similar applications as Simon's algorithm, but for the case when non-binary operations are involved. Equivalently, it is an algorithm for solving the well-known *The hidden subgroup problem.* Although it does not have many applications in cryptanalysis of block ciphers (since most of block ciphers utilize binary operations), our intention is to recall known applications and its relation to the problem of finding a period of a given function (i.e. to shows its connection to Simon's problem). We start by recalling *The hidden subgroup problem (HSP).*

Problem 6.1. Given a group G, a finite set X and a black box/oracle function $f : G \to X$ with promise that there exist subgroup $H \leq G$ such that for all $a, b \in G$, $f(a) = f(b)$ if and only if $Ha = Hb$. The goal is to find H.

In [36], G. Kuperberg introduced and described a quantum algorithm with subexponential complexity $2^{O(\sqrt{logN})}$ for the dihedral hidden subgroup problem (DHSP), where G is the dihedral group D_N. The definition of the dihedral group is given as follows.

Definition 6.2. The *dihedral group* D_N is the group of symmetries of a regular polygon with N vertices. The group order of D_N is $2N$.

There are two types of symmetries of the N-gon: rotations of angle $\frac{2k\pi}{N}$ and reflections through a line which makes an angle of $\frac{\pi k}{N}$ with the horizontal axis. One group presentation for the dihedral group is:

$$D_N = \langle x, y : x^2 = y^N, xyx = y^{-1} \rangle. \tag{6.1}$$

Using this group presentation, we can write any element of D_N in the form $g = y^t x^s$, where $t \in \mathbb{Z}_2$ and $s \in \mathbb{Z}_N$. For $t = 0$, $g = x^s$ is *rotation* and for $t = 1$, $g = yx^s$ is *reflection element.*

Example 6.3. Let us consider the dihedral group D_6. The order of this group is $2 \cdot 6 = 12$. The set of rotation elements is given as $\rho = \{1, x, x^2, x^3, x^4, x^5\}$, where 1 is the identity element of the group D_6. The set of reflection elements is $\sigma = \{y, yx, yx^2, yx^3, yx^4, yx^5\}$. Let x be a rotation of the angle $\frac{2\pi}{6}$. In Figure 19, we

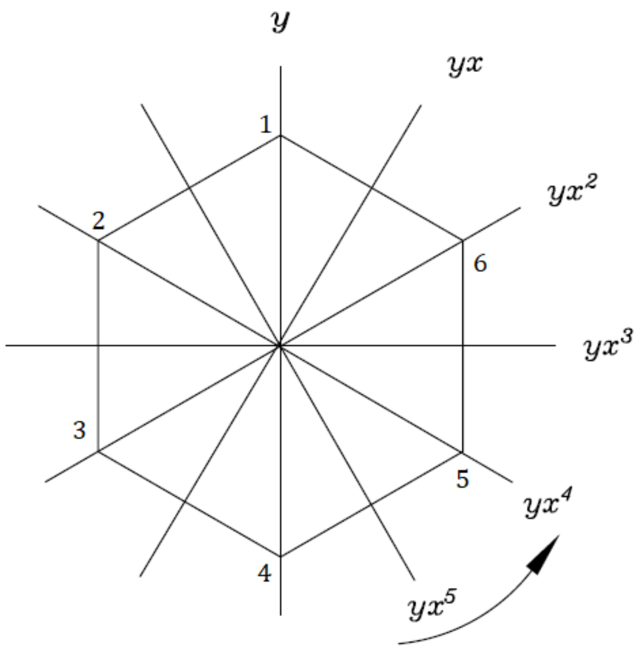

Figure 19: Dihedral group D_6.

conclude that

$$x : \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix}, \quad yx : \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}.$$

Similarly, we can determine the other elements of D_6.

In [36], it was assumed that $H = \langle xy^s \rangle$, where s is the *slope* of a reflection. Therefore, in order to solve HSP we want to find s. This is justified by the following proposition.

Proposition 6.4. *[36] Finding an arbitrary hidden subgroup H of D_N reduces to finding the slope s of a hidden reflection.*

Proof. The subgroups of D_N are either cyclic or dihedral. Assume that H is not a reflection. Then it is the trivial subgroup or it has a non-trivial intersection with the cyclic subgroup $C_N = \langle x \rangle$. Finding the hidden subgroup $H_1 = H \cap C_N$ is easy if we know the factors of N. Note that we can factor N using Shor's algorithm [52]. Then the quotient group is H/H_1 is either trivial or a reflection in the quotient group G/H_1. If H is trivial, then this will be determined by the fact that an algorithm to find the slope of a hidden reflection must fail. $\square$

In [36], G. Kuperberg firstly described the quantum algorithm when n is the power of 2 i.e. $N = 2^n$, and then introduced another algorithm that works for every N. We now recall the case when $N = 2^n$.

An algorithm for solving DHSP when $N = 2^n$ [36]

The main goal of this algorithm is to find the parity of s. When we find out its parity, the main part can be repeated with a subgroup of D_N isomorphic to $D_{N/2}$. There are two such subgroups:

$$H_0 = \langle x^2, y\rangle, \quad H_1 = \langle x^2, yx\rangle.$$

Depending on whether $s\ (mod\ 2)$ is equal to 0 or 1, the subgroup H is contained in H_0 or H_1, respectively. Now, we describe define a *hidden subgroup state.* This result is recalled from [17]. For any finite set X we can define the *constant pure state* $|X\rangle$

$$|X\rangle = \frac{1}{\sqrt{|X|}} \sum_{x \in X} |x\rangle.$$

Using this, we are able to prepare a uniform superposition over group elements:

$$|G\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle.$$

Then, we call the black box /oracle function $\mathcal{O}_f : |g\rangle \to |g\rangle \otimes |f(g)\rangle$ and obtain:

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |f(g)\rangle.$$

Now we measure the second register. If we get some value $x \in S$, then state is projected to the uniform superposition of those $g \in G$ such that $f(g) = x$. By definition of f, $f(a) = f(b)$ if and only if a and b are in the same coset of H, those g make some coset of H. Since every coset has the same cardinality, they occur with the same probability. Thus, we obtain the *coset state*:

$$|Hg\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |hg\rangle.$$

This result is equivalent to *hidden subgroup state*

$$\rho_{G/H} = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |Hg\rangle\langle Hg|. \tag{6.2}$$

In our special case $G = D_N$. We mentioned earlier that every element g of D_N (the relation (6.1)) can be written in the form $g = y^t x^s$, where $t \in \mathbb{Z}_N$ and $s \in \mathbb{Z}_2$. Since $N = 2^n$, it means that we need n qubits to describe s and one qubit to describe t.

Now, we are able to introduce the quantum algorithm proposed in [36].

Kuperberg's first algorithm (Algorithm 1):

1. **Input:** An oracle $f : G \to X$, where $G = D_N$ with the hidden subgroup $H = \langle yx^s\rangle$ and $N = 2^n$.

2. In this step we make a list L_0 of copies of $\rho_{D_N/H}$ by applying the unitary operator to the constant pure state $|D_N\rangle$. Explicitly, we apply the quantum Fourier transform (QFT) to $|s\rangle$ as

$$QFT|s\rangle = \frac{1}{N}\sum_{k\in\mathbb{Z}_N} e^{\frac{2\pi isk}{N}}|k\rangle.$$

   and measure it. This gives us the measured value with a uniform probability, and collapses the remaining qubit in the state

$$|\psi_k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi iks}{N}}|1\rangle), \tag{6.3}$$

   We always prepare the same state $\rho_{D_N/H}$ and make the same measurement, hence we can assume that we have $2^{O(\sqrt{n})}$ states of the form $|\psi_k\rangle$, where k is chosen independently and uniformly at random but different for every state obtained in this way. Since

$$|\psi_{-k}\rangle = X|\psi_k\rangle,$$

   where X is a Pauli-X gate, the state $|\psi_{-k}\rangle$ has the equivalent information about s as $|\psi_k\rangle$.

3. Let $m = \lceil n-1 \rceil$, then for each $i \in \{0, \dots, m-1\}$, a list L_i of qubit state such that k has at least mi the last bits in the row equal to zero. Divide L_i into pairs $|\psi_k\rangle$, $|\psi_l\rangle$, so that they have at least m common low bits or $n-1-mi$ bits if $m = i-1$. To combine states we do the following:

$$\begin{aligned}|\psi_k\rangle \otimes |\psi_l\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi iks}{N}}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi ils}{N}}|1\rangle)\\ &= \frac{1}{2}(|00\rangle + e^{\frac{2\pi ils}{N}}|01\rangle + e^{\frac{2\pi iks}{N}}|10\rangle + e^{\frac{2\pi s(k+l)}{N}}|11\rangle).\end{aligned}$$

   Applying CNOT gate we obtain:

$$\frac{1}{2}(|00\rangle + e^{\frac{2\pi ils}{N}}|01\rangle + e^{\frac{2\pi iks}{N}}|11\rangle + e^{\frac{2\pi s(k+l)}{N}}|10\rangle) = \frac{1}{\sqrt{2}}(|\psi_{k+l}\rangle|0\rangle + |\psi_{k-l}\rangle|1\rangle).$$

   After measurement in the classical basis $\{|0\rangle, |1\rangle\}$, we get a classical state, out of superposition, i.e. we get states $|\psi_{k+l}\rangle$, and $|\psi_{k-l}\rangle$, respectively. This is procedure how to extract the new qubit $|\psi_{k\pm l}\rangle$ from the old qubits $|\psi_k\rangle$ and $|\psi_l\rangle$. Make a new list L_{i+1} that contains only states of the form $|\psi_{l-k}\rangle$.

4. In this step the final list L_m consists of the remaining states of the form $|\psi_0\rangle$ and $|\psi_{2^{n-1}}\rangle$, as follows. In order to obtain state $|\psi_{2^{n-1}}\rangle$, observe $2^{n-1} = \frac{2^n}{2} = \frac{N}{2}$ and relation (6.3). Therefore, state $|\psi_{2^{n-1}}$ is equal to $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{\frac{2\pi iN}{2N}s}) = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{\pi is}|1\rangle)$. Note that if s is even i.e the lowest bit of s is 0, we have $|+\rangle$, otherwise it is $|-\rangle$. Measure it in the $\{|+\rangle, |-\rangle\}$ basis to determine the parity of the slope s.

5. Repeat steps $1 - 3$ with the subgroup of D_N which is isomorphic to $D_{N/2}$ and which contains H.

In **Step 3** we described the combinations of states $|\psi_k\rangle$ and $|\psi_l\rangle$. For all k and l they can be used once, and we want to produce a special state $|\psi_{2^{n-1}}\rangle$. Thus, it is natural that we need an algorithm to choose which elements to combine. For more details on this part, we refer the reader to [11]. For this purpose G.Kuperberg has constructed a sieve that begins with $2^{\theta(\sqrt{n})}$ qubits. Each stage of the sieve will determine $|\psi_k\rangle$ and $|\psi_l\rangle$ such that k and l agree in $\theta(\sqrt{n})$ low bits in additional to their trailing zeros. The probability that the label $k \pm l$ of the extracted qubit has $\sqrt{n}$ more trailing zeros than k or l is $\frac{1}{2}$. If the sieve has $\theta(\sqrt{n})$ steps, we can expect it to result $|\psi_{2^{n-1}}\rangle$ (cf. [36]).

Another algorithm described in [36] solves DSHP for $N \neq 2^n$. There are a few differences in comparison to the first Kuperberg's algorithm.

Kuperberg's second algorithm (Algorithm 2): [36]

1. **Input:** An oracle $f : D_N \to X$ with a hidden subgroup $H = \langle yx^s \rangle$.

2. Similarly as in the first Kuperberg's algorithm we make a list L_0 of copies of $\rho_{D_N/H}$. Using a QFT on $\mathbb{Z}_N$ and a measurement, we extract a qubit state $|\psi_k\rangle$, here $0 \leq k \leq \frac{N}{2}$. Since N is not a power of 2, the QFT is more complicated on $\mathbb{Z}_N$. For more details we refer to [35].

3. For each $0 \leq j < m$, where $m = \lceil \sqrt{(\log_2 N) - 2} \rceil$, we suppose a list L_j of qubit states $|\psi_k\rangle$ such that $0 \leq k \leq 2^{m^2 - mi + 1.}$ Randomly divide L_j into pairs of qubits $|\psi_k\rangle$ and $|\psi_l\rangle$ such that $|k - l| \leq 2^{m^2 - m(i+1)+1}$. The new list L_{j+1} consist states of the form $|\psi_{k-l}\rangle$.

4. The final list L_m contains the states $|\psi_0\rangle$ and $|\psi_1\rangle$.

5. We supposed that N is not of the power of 2, therefore we can write $N = 2^\alpha b$, where $b \neq 1$ is odd. By the well known Chinese remainder theorem

$$C_N \cong C_{2^\alpha} \times C_b,$$

where C_N si a cyclic group. For each $1 \leq j \leq \lceil \log_2 N \rceil$, apply the first Kuperberg's algorithm to produce many $|\psi\rangle$ with $2^{(\alpha,j)}|k$. Then repeat **Steps** $1 - 4$ after applying the group authomorphisam $x \to x^{2-i}$ to the factor C_b of D_N. This produces copies of $|\psi_2^j\rangle$, thus cosine observations $\cos(\pi i 2^j (s-t)/N)^2$. These observations determine s.

In [36], G. Kuperberg also introduced an algorithm for solving HSP when $N = p^n$. Let us a consider function $\xi(k)$, whose value represents the number of factors of r in k, with the exception that $\xi(0) = 0$. Within the list L of qubit states available only at any given time, we will pick $|\psi_k\rangle$ and $|\psi_l\rangle$ to maximize $\xi(k \pm l)$.

Kuperberg's third algorithm (Algorithm 3): [36]

1. **Input:** An oracle $f : D_N \to X$ with a hidden subgroup H and $N = p^n$
2. Within the sublist L' of L that minimizes ξ, repeatedly extract $|\psi_{(k\pm l)}\rangle$ from a pair of qubits $|\psi_k\rangle$ and $|\psi_l\rangle$ that maximize $\xi(k \pm l)$.
3. After enough qubits $|\psi_k\rangle$ appear with $\frac{N}{p}|k$, measure $s \ (mod\, p)$. Then repeat the algorithm with a subgroup D_N isomorphic to $D_{N/r}$.

G. Kuperberg proved the next two results for *Algorithms 1, 2 and 3.*

Theorem 6.5. *[36] Algorithms 1 and 2 find a hidden reflection in the dihedral group $G = D_N$ with the time and query complexity $2^{O(\sqrt{\log N})}$.*

The complexity in Theorem 6.5 is subexponential, while on the other hand any classical algorithm requires at least $2\sqrt{N}$ queries. We note that the 3-rd algorithm has higher time and query complexity.

Theorem 6.6. *[36] Algorithm 3 requires $\widetilde{O}(3^{\sqrt{2\log_3 N}})$ queries and quasilinear time in the number of queries.*

For the proofs of Theorems 6.5 and 6.6 we refer the reader to [36]. In general, there are other problems which are quivalent to $DHSP$, such as the abelian hidden shift problems and the hidden reflection problem in G. Let G be an abelian group and let e^G denote the multiplicative form of the same group. Let $C_n = e^{\mathbb{Z}_n}$ be the multiplicative cyclic group of order n. Since G is an abelian group, we can define generalized dihedral group to be the semidirect product

$$D_G \cong C_2 \ltimes G,$$

with the conjugation relation $x^{-1} = yxy$, for any $x \in e^G$ and for the non-trivial $y \in C_2$. Note that any element of the form yx is the reflection in D_G.

Definition 6.7. Let G be an abelian group. Then the function $f : G \to X$ is a function which is injective except for

$$f(g) = f(s - g),$$

were s is hidden in G.

Remark 6.8. In what follows the terms Kuperberg's algorithm or basic algorithm represents Kuperberg's first algorithm.

6.1 KUPERBERG'S ALGORITHM FOR SOLVING THE HIDDEN SHIFT PROBLEM

In [11] authors have introduced new results for solving the hidden shift problem, that is a natural variant of the hidden subgroup problem, using Kuperberg's algorithm. Let us first recall this problem.

Problem 6.9. (*Hidden shift problem*) Let $(G, \cdot)$ be a finite group. Given two injective functions f, g with the promise that there exist $s \in G$ such that for all x, it holds that $f(x) = g(x \cdot s)$. The goal is to find s.

Notice that if the group operation is $\oplus$, then $f(x) = g(x \oplus s)$ and the given problem can be solved by Simon's algorithm in polynomial time. More precisely, *Hidden Shift Version of Simon's Problem* is defined as follows.

Problem 6.10. [12] Let $f, g : \{0,1\}^n \to \{0,1\}^n$ be two permutations with promise that there exists $s \in \{0,1\}^n$, that for all x, it holds $f(x) = g(x \oplus s)$. The goal is to find s.

We saw earlier that the Kuperberg's algorithm works for $N = 2^n$ i.e. it works for modular addition which is power of 2. Before we recall the algorithm for solving Problem 6.9, which is presented in [11], we define an oracle function $\mathcal{O}$ as follows:

$$\mathcal{O} : |a\rangle|x\rangle|y\rangle \to \begin{cases} |0\rangle|x\rangle|y \oplus f(x)\rangle, & \text{if } \ a = 0, \\ |1\rangle|x\rangle|y \oplus g(x)\rangle, & \text{if } \ a = 1. \end{cases} \tag{6.4}$$

The quantum circuit for Kuperberg's algorithm, with oracle is given in the Figure 20. Notice, that first Hadamard transform is applied to the first two registers, and then the oracle function $\mathcal{O}$ is called followed by measurement of the second register. To estimate the complexity, in [11] authors consider the case when $n = \log_2 |G|$.

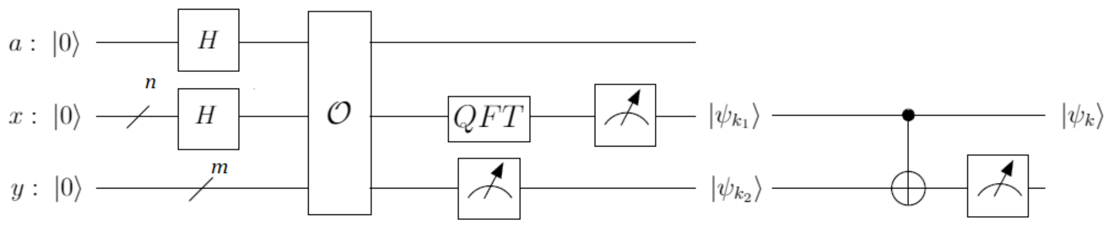

Figure 20: The quantum circuit of Kuperberg's algorithm [11].

Kuperberg's algorithm for hidden shift problem [11]

1. Prepare three quantum registers in the zero state:
$$|\psi_0\rangle = |0\rangle|0\rangle^{\otimes n}|0\rangle^{\otimes m}.$$
2. Applying Hadamard transforms to the first two register in order to obtain state:
$$\begin{aligned}|\psi_1\rangle &= (H|0\rangle) \otimes (H^{\otimes n}|0\rangle^{\otimes n}) \otimes |0\rangle^{\otimes m}\\ &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |0\rangle^{\otimes m}\\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (|0\rangle|x\rangle|0\rangle^{\otimes m} + |1\rangle|x\rangle|0\rangle^{\otimes m}).\end{aligned}$$
3. Apply the oracle $\mathcal{O}$. By (6.4) one obtains the state:
$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (|0\rangle|x\rangle|f(x)\rangle + |1\rangle|x\rangle|g(x)\rangle).$$
4. Measure the third register to find a vector y_0 such that $f(y_0) = g(x)$. For simplicity in the obtained state we omit amplitudes and last register. Therefore, we have
$$|\psi_3\rangle = \sum_{f(x)=y_0} |0\rangle|x\rangle + \sum_{g(x)=y_0} |1\rangle|x\rangle.$$
Thanks to the promise given in (6.9) and the fact that f and g are injective functions, we have that $f(x_0) = g(x_1)$ implies $x_0 = x_1 = x_0 + s$. Since we want to consider modular addition we write operation $+$. Therefore, we have
$$|\psi_3\rangle = |0\rangle|x_0\rangle + |1\rangle|x_0 + s\rangle.$$
5. Then we apply the QFT on the second register and measure the result. This gives us a k with a uniform probability, and collapses the remaining qubits into the state
$$|\psi_k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi iks}{N}}|1\rangle).$$
The remaining steps are the same as in Kuperberg's algorithm described before.

In [11] the authors proposed a new variant of this algorithm with improved time complexity. It still has a subexponential time, but it needs only one iteration to find all qubits instead of n. In order to find a more precise complexity estimations, the authors in [11] have simulated the classical part of the algorithm, replacing the quantum measurement by random outcomes. In Table 3, are summarized results of these simulations for different values of n, with 90% success probability. From these simulations authors

Algorithm 6: Improved variant of Kuperberg's algorithm [11]

```
1 Generate $N$ random numbers in $\mathbb{Z}_N$, where $N = 2^n$.
2 Separate them in lists $L_i$ of elements divisible by $2^i$ and not $2^{i+1}$
3 for $i = 1, \dots, n-2$
4     while $|L_i| \geq 3$ do
5         Pop two elements $(k, l)$ of $L_i$ where $k+l$ or $k-l$ has the highest
            possible divisibility by 2 (and is not 0)
6         Chose element $c$ randomly in $\{k+l, k-l\}$ and put it in the
            corresponding $L_j$
7         if $\forall i \in \{0, \dots, n-1\}$, $L_i \neq \emptyset$ then
8             return Found
9 return Failure
```

Table 3: Results of the simulation of the improved variant of Kuperberg's algorithm [11]

n	*queries*	$\log_2(queries)$	$1.8 \cdot \sqrt{n} - 0.5$	number of tests
16	118	6.9	6.7	10^6
32	826	9.7	9.7	10^6
64	14975	13.9	13.9	$5 \cdot 10^5$
80	49200	15.6	15.6	10^5
128	$9.8 \cdot 10^5$	19.9	19.9	$5 \cdot 10^4$

concluded that an approximate complexity is $0.7 \cdot 2^{1.8\sqrt{n}}$ for a 90% success probability. This matches the exponential complexity of $\widetilde{O}(2^{\sqrt{2\log_2(3)n}})$. In the same work, authors proposed a new algorithm which is a combination of Simon's and Kuperberg's algorithms for the first time.

An attack on Poly 1305-AES: This improved version of Kuperberg' algorithm found its application in the quantum cryptanalysis of Poly 1305-AES message-authentication code, proposed by D. Bernstein in [5]. It computes a 128-bit authenticator $Poly1305_r$ $(m, AESk(n))$ of a message m, using a 128-bit AES key k, a 128-bit additional key r, and a 128-bit nonce n. The function is defined as:

$$Poly1305_r(m_i, AESk(n)) = \Big(\Big(\sum_{i=1}^{q}(c_{q-i+1}+2^{128})r^i \mod 2^{130}-5\Big)+AES_k(n)\Big) \pmod{2^{128}},$$

where $q = \lceil \frac{l}{128} \rceil$ and c_i are constants, which are defined as in [5]. The length l is any non-negative number, which can be different for different messages(cf. [5] for more details).

In order to make a quantum attack, Bonnetain and Naya-Plasencia in [11] considered the function

$$Poly_n : |m_1\rangle|m_2\rangle|0\rangle \to |m_1\rangle|m_2\rangle|Poly1305_r(m_1, m_2, AESk(n))\rangle.$$

It was shown that this attack, which is based on improved Kuperberg's algorithm, has complexity of about 2^{38} in time and queries. So, $Poly1305 - AES$ is affected by quantum cryptoanalysis. For more detail we refer reader to [11].

6.2 QUANTUM ATTACK ON KEY-ALTERNATING CIPHER

In [12] X. Bonnetain et al. proposed the quantum slide attack on the key-alternating cipher using modular additions instead of XOR. This bock ciphers generalizes the Even-Mansour construction over multiple rounds. We begin by recalling the design of an idealized model of a key-alternating cipher presented in [9].
Let be permutations $P_1, P_2, \dots P_t : \{0,1\}^n \to \{0,1\}^n$,where $t \geq 1$ and keys $k_0, k_1, \dots k_t \in \{0,1\}^n$ are given. The key-alternating block cipher $E = E_{k_0,\dots,k_t} : \{0,1\}^n \to \{0,1\}^n$ is defined by

$$E(x) = E_{k_0,\dots,k_t} = P_t(\dots P_2(P_1(x \oplus k_0)\dots) \oplus k_t$$

for $x \in \{0,1\}^n$. An illustrated version of this is shown in Figure 21.

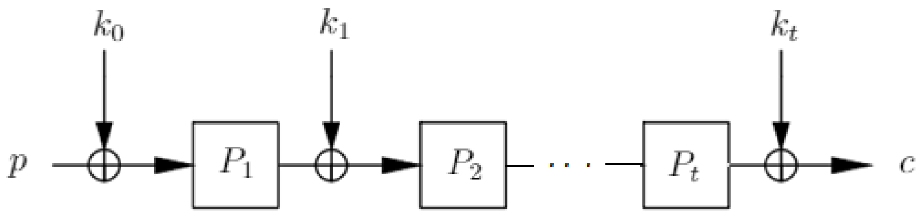

Figure 21: A key-alternating cipher.

In the case of XOR, we are able to construct a quantum attack based on Simon's algorithm that distinguishes the key-alternating cipher. Now suppose that instead of XOR, we have modular addition, and based on the definition of the quantum slide attack, we assume that all keys are equal, i.e. $k = k_0 = k_1 = \dots = k_t$.

Now, define the function $G : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ as follows:

$$G(b||x) = \begin{cases} f(x) = P(E_k(x)) - x, & \text{if } b = 0, \\ g(x) = E_k(P(x)) - x, & \text{if } b = 1. \end{cases} \tag{6.5}$$

In Section 3.1.4, we defined a quantum slide attack, and thus for all x the condition $P(E_k(x)) + k = E_k(P(x + k))$ is satisfied. Therefore, the following lemma holds.

Lemma 6.11. *Let f and g be functions defined as in the relation (6.5). Then for any arbitrary distinct vectors $x, y \in \{0,1\}^n$ holds*

$$f(x) = g(y) \iff y = x + k.$$

Proof. ($\Rightarrow$) Let us assume first that for any distinct vectors $x, y \in \{0,1\}^n$ holds $f(x) = g(y)$, i.e. $P(E_k(x)) - x = E_k(P(y)) - y$. Since it holds $P(E_k(x)) + k = E_k(P(x+k))$, we can write $P(E_k(x)) = E_k(P(x+k)) - k$. Hence

$$P(E_k(x)) - x = E_k(P(x+k)) - (k+x) = E_k(P(y)) - y,$$

which implies that $y = x + k$.
($\Leftarrow$) Now suppose $y = x + k$, then by the relation (6.5) it holds $f(x) = P(E_k(x)) - x$ and $g(y) = E_k(P(y) - y = E_k(P(x+k)) - (x+k)$. Since $E_k(P(x+k)) = P(E_k(x)) + k$, we can write. $g(x+k) = P(E_k(x)) + k - (x+k) = P(E_k(x)) - x = f(x)$. □

By Lemma 6.11, we deduce that G satisfies the conditions of the *Hidden shift problem* 6.9, that is

$$G(0||x) = P(E_k(x)) - x = E_k(P(x+k)) - (k+x) = G(1||x+k).$$

Also, we note that one can apply Kuperberg's algorithm as well, which recovers the key k with the complexity of$2^{1.78\sqrt{n}}$ [12].

6.3 QUANTUM ATTACK ON ONE ROUND FEISTEL NETWORK

In Section 3.1, we defined Feistel cipher and described a quantum attack based on Simon's algorithm [37] which breaks 3-round Feistel network in polynomial time $O(n)$. We now describe a quantum slide attack on the first round Feistel network with modular addition instead of XOR, proposed in [12]. According to the definition 3.4, for one round of Feistel network, the right part of the plaintext R_0 is equal to the left part of the second plaintext L_1, i.e., $R_0 = L_1$. Since R_0 is fixed, in this quantum attack one can consider a variable $k' = F(R_0 + k)$ (see Figure 22) as an equivalent key [12] where F is a weak round function. Let E_k be an encryption function and T_L and T_R be the functions that truncate a Feistel state only to its left and right parts, respectively. Define the function $G : \{0,1\} \times \{0,1\}^{\frac{n}{2}} \to \{0,1\}^{\frac{n}{2}}$ by

$$G(b||x) = \begin{cases} f(x) = T_R(E_k(x, R_0)), & \text{if } b = 0, \\ g(x) = T_L(E_k(R_0, x)), & \text{if } b = 1. \end{cases} \tag{6.6}$$

In Figure 22, one concludes that $f(x) = g(x + k')$, which implies that the function G defined in (6.6) satisfies the *Hidden shift problem.* Consequently, the key $k' = F(R_0+k)$

can be recovered by Kuperberg's algorithm, and since F is a weak function, a secret key k can be determined. This attack has complexity $2^{1.2\sqrt{n}}$, which is worse than in Simon's case.

Remark 6.12. One of the improved variants of the slide attack is the complementation slide attack [8]. Similar to the previous work for, authors in [12] illustrated this on a Feistel cipher with 2-round self-similarity and then applied Kuperberg's algorithm.

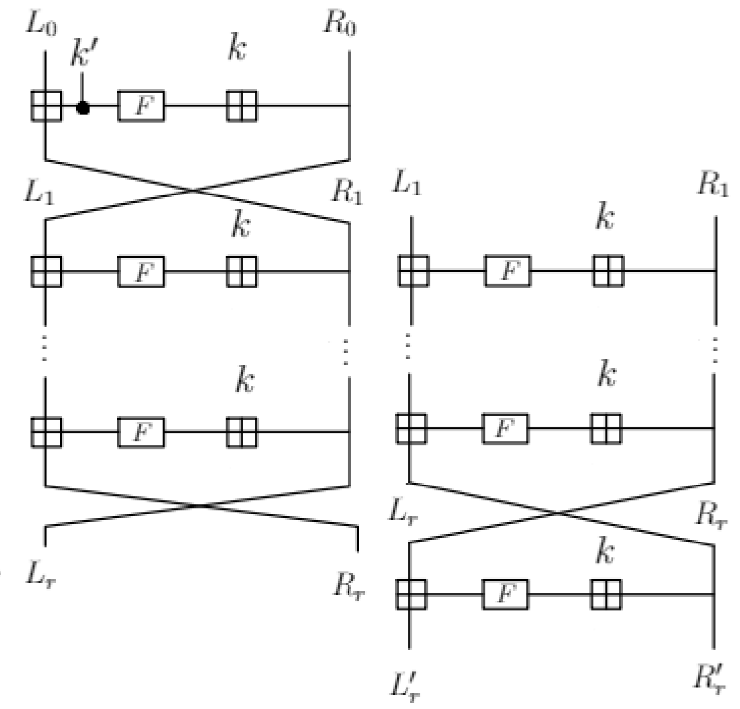

Figure 22: The slide attack on Feistel scheme with one round self-similarity and modular addition [12].

7 CONCLUSION

It is well know that Shor's algorithm is going to compromise the classical public-key cryptography in quantum environment. On the other hand, symmetric-key cryptography (referring to block ciphers) is appearing to be quite unexplored. Base on the works published in the last decade, it has been shown that the development of new quantum attacks is becoming more efficient. Not only that one is able to combine classical and quantum attacks, but also the quantum algorithms can be combined as well. In this context, it was initially an opinion of the community that the resistance to Grover's algorithm is achieved just by doubling the secret key. However, by combining it with Simon's algorithm, it turns out not the be sufficient. It has been presented throughout the thesis that various block ciphers can be easily broken by using just the Simon-Grover algorithm, which is possible to combine with other known classical attacks. We note that during the writing of this thesis, various new articles have been published which are supporting the fact that the security of block ciphers has to be re-evaluated. Although in most of the cases it has been shown that the Simon-Grover algorithm is more efficient to encryption schemes based on Feistel network, there is still no guarantee that the SPN-based schemes will stay secure. At this moment, it appears that the AES-like schemes are resistant to known quantum attacks. To these schemes, an attack based on Grover's algorithm is the only one that one can consider.

8 DALJŠI POVZETEK V SLOVENSKEM JEZIKU

Skozi zgodovino so se ljudje vedno zbirali v skupine, tvorili plemena, imperije, kraljestva, države. Zato so potrebovali zaščito informacij tako pred zunanjimi sovražniki kot tudi pred samimi člani skupine. Posledično so sčasoma razvili znanost, ki zaščiti sporočila in jo danes poznamo pod imenom *kriptografija* (kryptós - skrito, graphein - pisati). Začetek njenega razvoja je povzročila iznajdba črk. Prvi znaki obstoja kriptografije najdemo na zidu grobnice iz časa starega kraljestva v Egiptu, ki sega 1900 let pred našim štetjem. Pred njenim modernim razvojem, ki predstavlja točko preloma v znanosti in tehnologiji, je bila poglavitna naloga kriptografije v zaščiti zaupnih sporočil med vojno, religijskimi revolucijami ter pred političnimi tekmeci. Medtem se moderna kriptografija uporablja pri zaščiti bančnih računov, bazah podatkov, računalnikov, elektronske pošte itd.

Na drugi strani se je z namenom izboljšanja kriptografskih shem razvila tudi *kriptoanaliza*, ki je znanost analiziranja in vdiranja v kriptografske sisteme. Obe znanosti skupaj tvorita *kriptologijo*, ki zajema tako metode zaščite kot tudi metode vdiranja v skritost sporočila pri komunikaciji med dvema deležnikoma. Slednja danes predstavlja eno najpomembnejših znanosti. Čeprav se je skozi zgodovino močno spremenila, je njen poglavitni namen o skritosti ostal enak. Z namenom boljšega razumevanja moderne kriptografije opišimo dve njeni glavni področji, ki se razlikujeta glede na tip skritega ključa. To sta *kriptografija javnega ključa* in *simetrična kriptografija* (več informacij najdemo v [54]). Recimo, da Alice in Bob komunicirata preko nezaščitenega kanala (npr. preko telefonske linije, računalniškega omrežja ali radia). Glavni cilj kriptografije je preprečiti zunanjemu nasprotniku, ki ga pogosto imenujemo Eve/Oscar, da bi videl vsebino sporočila v komunikaciji. Sporočilo, ki si ga Alice in Bob želita izmenjati imenujemo *navadni tekst*. Pri onemogočanju nasprotnika Alice izbere skrivni ključ K (tj. enkripcijski ključ) in spremeni navadni tekst P v skrivno sporočilo, ki ga imenujemo *šifriran tekst* C. Šifriran tekst nato Alice pošlje Bobu, ki uporabi dekripcijski ključ K', da pride do originalnega sporočila (ključa K in K' nista nujno enaka). Transformacijo, ki navadni tekst spremeni v šifriran tekst, imenujemo *enkripcija*, obratni postopek pa *dekripcija*.

Glede na to, ali sta enkripcijski in dekripcijski ključ med sabo enaka, ločimo dve vrsti kriptografije. V primeru enakosti ključev (tj. $K = K'$) govorimo o *simetrični* *kriptografiji.* Sicer, če velja $K \neq K'$, imamo opravka z *asimetrično kriptografijo.* V slednjem primeru je enkripcijski ključ javni, dekripcijski pa zasebni.

V zadnjem času je poleg klasične moderne kriptografije veliko govora o post-kvantni kriptografiji, ki se nanaša na poskus razvoja kvantnih računalnikov. Ideja o razvoju kvantnega računalniku pripada slovitemu fiziku Richardu Feynmanu (1982) [23], čigar delo je vplivalo na številne kvantne algoritme. Prvi poskusi konstrukcije kvantnega računalnika so se pojavili v zadnjih dveh desetletjih z namenom doseči kvantno premoč, ki bi jo predstavljala naprava, ki jo je moč programirati, in je sposobna v realnem času rešiti določene probleme, ki jih klasični superračunalniki ne zmorejo.

V splošnem se pojem algoritma nanaša na serijo korakov, ki so potrebni za razrešitev določenega problema. Pri kvantnih algoritmih se vsaj en izmed teh korakov zanaša na kvantna stanja kot sta superpozicija ali prepletenost. Tovrstni algoritmi so običajno opisani preko kvantnega vezja, ki predstavlja model kvantnega računanja, kjer so koraki, ki rešujejo posamezen del problema, kvantna vrata uporabljena na določenih kubitih.

V določenih primerih so kvantni algoritmi precej hitrejši in efektivni v primerjavi s klasičnimi algoritmi. Zato se za (bližnjo) prihodnost, ko naj bi po nekaterih napovedih kvantni računalniki postali del vsakodnevnega življenja, pojavi vprašanje o varnosti naših (skritih) podatkov. Aplikacije kvantnih algoritmov v zadnjih dveh desetletjih so nakazale, da klasične sheme iz kriptografije javnega ključa ne nudijo potrebne zaščite. Medtem se zaščita shem v simetrični kriptografiji zmanjša v primerjavi s klasičnem kontekstom. Leta 1994 je npr. Peter Shor [52] podal polinomski algoritem za faktorizacijo celih števil ter diskretne logaritemske probleme. Slednje pomeni, da lahko Shorov algoritem vdre v RSA kriptosisteme. Na drugi strani Groverjev algoritem [25] zahteva, da se dolžina skrivnega ključa podvoji. Kasneje so aplikacije kvantnih algoritmov, kot so Simonov [53], Bernstein-Vaziranijev [6], Kuperbergov [36] algoritem, ter njihove kombinacije naznanile, da razumevanje zaščite shem v simetrični kriptografiji še ni na zadovoljivi ravni.

Za boljše razumevanje poglavitnih ciljev magistrskega dela, potrebujemo nekaj oznak, ki so vezane na sheme simetrične kriptografije. Le-te tipično ločimo na tokovne šifre in bločne šifre, vendar bomo v delu obravnavali le slednje. Pojem n-bitne bločne šifre je ponazorjen s funkcijo $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$, za katero velja, da je za vsak fiksen $K \in \mathcal{K}$ ($\mathcal{K}$ je *prostor ključev*) *enkripcijska funkcija* $E(P, K) = E_K(P)$ permutacija. Slednja priredi ključu K, z dolžino k, ter bločnemu sporočilu P (*navadni tekst*) dolžine n, n-bitni *šifriran tekst* C. Inverzna, tj. dekripcijska funkcija $D : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ je definirana s predpisom $D(C, K) = E_K^{-1}(C) = D_K(C)$, kjer je $D_K(E_K(P)) = P$ za vsak $K \in \mathcal{K}$.

Najpogostejši dizajnerski pristop bločnih šifer je dandanes sestavljen iz iteriranih produktnih šifer, katerih struktura temelji bodisi na Feistelovem omrežju (FN) bodisi na zamenjalno permutacijskem omrežju (SPN). Do enkripcije pri iteriranih produktnih bločnih šifrah pride z aplikacijo iste rundne funkcije (aplicirane večkrat), ki uporablja t.i. rundne ključe. Slednji so dobljeni s pomočjo glavnega skrivnega ključa in algoritma, ki razvršča ključe.

V magistrskem delu opišemo in analiziramo zgoraj omenjene kvantne algoritme v kontekstu kriptoanalize bločnih šifer, ki temeljijo na FN ali SPN strukturi. Znani so številni rezultati na tem področju, ki v zadnjem času pridobiva na veljavnosti. Čeprav je bil Simonov algoritem opisan že leta 1994, njegovo prvo aplikacijo v kriptoanalizi bločnih šifer zasledimo šele v delu [37] iz leta 2010, kjer je bilo pokazano, da je 3-rundno Feistelovo šifro mogoče zlomiti. Kasneje so pokazali, da t.i. Even-Mansourjeva konstrukcija prav tako ni varna [38]. Poleg tega sta Leander in May [39] prva uporabila kombinacijo Simonovega in Groverjevega algoritma na FX-konstrukciji. Pokazala sta, da t.i. “key whitening” procedura, ki povečuje prostor ključev, ni enako efektivna kot v klasičnem kontekstu. V magistrskem delu sta opisani tudi aplikaciji Berstein-Vaziranijevega in Kuperbergovega algoritma v konstekstu konstrukcije kvantnih razlikovalcev, ki temeljijo na periodičnem iskanju. Magistrsko delo je razdeljeno na osem poglavij, pri čemer poglavja 2-6 predstavljajo njeno jedro. Poglavju 2, kjer so predstavljena vsa orodja, ki jih v delu potrebujemo, sledijo štiri poglavja, ki so posvečena Simonovemu, Groverjemu, Berstein-Vaziranijevemu ter Kuperbergovemu algoritmu.

9 REFERENCES

[1] Y. Akihiro and I. Hirokazu, Quantum cryptoanalysis of block ciphers. *Algebraic Systems, Formal Languages and Computations. RIMS Kokyuroku* 1166 (2000) 235-243. *(Cited on pages 41 in 42.)*

[2] K. Aoki and A. Hosoyamada, On quantum Related-key attacks on Iterated Even-Mansour Ciphers. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences* 102(1) (2019) 27–34. *(Cited on page 33.)*

[3] M. Almazrooie, R. Abdullah, A. Samasudin and K. N. Mutter, Quantum Grover Attack on the Simplified-AES. *ICSCA 2018: Proceedings of the 2018 7th International Conference on Software and Computer Applications* (2018) 204–2011. *(Cited on pages 44, 48 in 50.)*

[4] E. Barker, L. Bassham, J. Dray, M.Dworkin, j.Foti, J. Nechvatal and E. Roback, Advanced Encryption Standard (AES). *Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD* (November,2001) . *(Cited on page 44.)*

[5] D. J. Bernstein, The Poly1305-AES Message-Authentication Code. *Fast Software Encryption. Lecture Notes in Computer Science* 3557 (2005) 3557. *(Cited on page 82.)*

[6] E.Bernstein and U.Vazirani, Quantum Complexity Theory. *SIAM Journal on Computing* 26 (1997) 1411–173. *(Cited on pages 3, 60, 61 in 88.)*

[7] E. Biham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology* 4 (1991) 3–72. *(Cited on page 67.)*

[8] A. Biryukov and D. Wagner, Advanced Slide Attacks. *In Bart Preneel, editor, Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium* 1807 LNCS (2000) 589—606.. *(Cited on pages 30 in 85.)*

[9] A. Bogdanov, L. Knudesen, G. Leander, F. X. Standaert, J. Steinberger and E. Tischhauser, Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations . *IACR Cryptology ePrint Archive.* 2012 (2012) . *(Cited on page 83.)*

[10] X. Bonnetain, Quantum Key-Recovery on full AEZ. *Selected Areas in Cryptography – SAC* (2017) 394–406. *(Cited on page 37.)*

[11] X. Bonnetain and M. Naya-Plasencia, Hidden Shift Quantum Cryptanalysis and Implications. *24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part I.* (2018) 560–592. *(Cited on pages VI, VII, 78, 80, 81, 82 in 83.)*

[12] X. Bonnetain, M. N. Naya-Plasencia and A. Schrottenloher, On Quantum Slide Attacks. (2020) 492-519. *(Cited on pages VII, 30, 35, 80, 83, 84 in 85.)*

[13] G. Brassad, P. Hoyer, M. Mosca and A. Tapp, Quantum amplitude amplification and estimation. *Contemporary Mathematics* 305 (2002) 53–74. *(Cited on page 52.)*

[14] C.Chaigneau and H. Gilbert, Is AEZ v4.1 Sufficiently Resilient Against KeyRecovery Attacks?. *IACR Transactions on Symmetric Cryptolog* 1(1) (2016) 114—133. *(Cited on page 37.)*

[15] Y. Chen, E. Lambooji and B. Mennink, How to Build Pseudorandom Functions from Public Random Permutations. *Advances in Cryptology – CRYPTO* 266–293 (2019) . *(Cited on page 38.)*

[16] D. Cheung, D. Maslov, J. Mathew and D. K. Pradhan, On the Design and Optimization of a Quantum Polynomial-Time Attack on Eliptic Curve Cryptography. *Theory of Quantum Computation, Communication, and Cryptography* 5106 (2008) 96–104. *(Cited on page 45.)*

[17] A. M. Childs, *Lecture Notes on Quantum Algorithms*, `https://www.cs.umd.edu/ amchilds/qa/qa.pdf`. (Viewed on: 4. 7. 2021.) *(Cited on page 76.)*

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Third Edition.* The MIT Press Cambridge, Massachusetts London, England, 2009. *(Cited on page 15.)*

[19] J. Cui, J. Guo, M. Li and L. Xu, Key-Recovery Attacks on LED-Like Block Ciphers. *TSINGHUA SCIENCE AND TECHNOLOGY* 24(5) (2019) 585-595. *(Cited on pages 32, 33 in 35.)*

[20] D. Deutsch and R. Jozsa, Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London A* 439 (1992) 553–558. *(Cited on page 60.)*

[21] X. Dong, B. Dong and X. Wang, Quantum attacks on some Feistel block ciphers. *Designs, Codes and Cryptography* 88 (2020) . *(Cited on pages VI, 35, 36, 56, 57 in 58.)*

[22] O. Dunkelman, N. Keller and A. Shamir, Minimalism in Cryptography: The Even-Mansour Scheme Revisited. *Advances in Cryptology – EUROCRYPT 2012* (2012) 336-354. *(Cited on page 24.)*

[23] R. P .Feyman., Simulating physics with computers.. *International Journal of Theoretical Physics* 21 (1982) 467–488. *(Cited on pages 2 in 88.)*

[24] M. Grassl, B. Langenberg, M. Roetteler and R. Steinwand, Applying Grover's algorithm to AES: quantum resource estimates. *Post-Quantum Cryptography* 9606 (2016) 29–43. *(Cited on page 44.)*

[25] L. K. Grover, A fast quantum mechanical algorithm for database search.. *Proceedings of the twenty-eight annual ACM symposium on Theory of Computing. STOC '96. Philadelphia, Pensylvania, USA: Associatian for Computing Machinery* (1996) 212–219. *(Cited on pages 2, 3, 40 in 88.)*

[26] J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher. *Cryptographic Hardware and Embedded Systems – CHES 2011* (2011) 326–341. *(Cited on pages 31 in 32.)*

[27] J. Hefferon, *Linear Algebra, Fourth Edition*, 2016. *(Cited on page 4.)*

[28] V. T. Hoang, T. Krovetz and P. Rogaway, Robust Authenticated-Encryption AEZ and the Problem that it Solves. *Advances in Cryptology – EUROCRYPT 2015* (2015) 15–44. *(Cited on page 37.)*

[29] L. Hongwei and L. Yang, A quantum algorithm to approximate the linear structures of Boolean functions. *Mathematical Structures in Computer Science* -1 (2016) 1–13. *(Cited on pages 62, 63, 64, 65, 66 in 67.)*

[30] L. Hongwei and L. Yang, Quantum Differential Cryptanalysis to the Block Ciphers. *International Conference on Applications and Techniques in Information Security* 557 (2015) 44-51. *(Cited on pages 66, 67, 68 in 69.)*

[31] S. Hodžić and L.R. Knudsen, A quantum distinguisher for 7/8-round SMS4 block cipher.. *Quantum Information Processing* 19, 401 (2020) . *(Cited on pages 53, 54 in 55.)*

[32] M. Kaplan, *Quantum Attacks Against Iterated Block Ciphers (2014)*, `https://arxiv.org/abs/1410.1434`. (Viewed on: 22. 6. 2021.) *(Cited on pages 42 in 43.)*

[33] J. Katz and Y. Lindell, *Introduction to Modern Cryptography (Champan and Hall/Crc Cryptography and Network Security Series)*. Champan & Hall/CRC, 2007. *(Cited on page 26.)*

[34] J. Kilian and P. Rogaway, How to Protect DES Against Exhaustive Key Search. *N. Koblitz, editor, CRYPTO, Lecture Notes in Computer Science* 1109 (1996) 252–267. *(Cited on pages 3 in 50.)*

[35] A. Kitaev, *Quantum Measurements and the Abelian Stabilizer Problem (1995)*, `http://arxiv.org/abs/quant-ph/9511026`. (Viewed on: 15. 7. 2021.) *(Cited on page 78.)*

[36] G. Kuperberg, A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput* 35 (2005) 170–188. *(Cited on pages 3, 74, 75, 76, 78, 79 in 88.)*

[37] H. Kuwakado and M.Morii, Quantum distignuisher between 3-round Feistel chiper and the random permutation. *2010 IEEE International Symposium on Information Theory* 41 (2010) 2682–2685. *(Cited on pages 3, 19, 20, 21, 22, 23, 84 in 89.)*

[38] H. Kuwakado and M. Morii, Security on the quantum-type Even-Mansour cipher. *2012 International Symposium Information Theory and its Application* (2012) 312–316. *(Cited on pages 3, 24, 26 in 89.)*

[39] G. Leander and A. May, Grover Meets Simon- Quantumly Attacking the FX-construction. *Advances in Cryptology ASIACRYPT 2017, International Conference on the Theory and Application of Cryptology and Information Security, LNCS* 10625 (2017) 161–178. *(Cited on pages 3, 36, 50, 52 in 89.)*

[40] M. Liskov, R. L. Rivest and D. Wagner, Tweakable Block Ciphers. *Journal of Cryptology* 24 (2011) 588—613. *(Cited on page 37.)*

[41] M. Loep , *A Course in Quantum Computing*, `https://lapastillaroja.net/wp-content/uploads/2016/09/Intro_to_QC_Vol_1_Loceff.pdf`. (Viewed on: 6. 7. 2021.) *(Cited on page 35.)*

[42] A. J. Menezes and P. C. van Oorschot, *S. A. Vanstone*. Handbook of APPLIED CRYPTOGRAPHY. CRC Press, 1996 *(Cited on pages 20 in 26.)*

[43] M. Mitton, On the Walsh-Fourier analysis of Boolean functions. *Journal of Discrete Mathematical Sciences and Cryptography* 9(3) (2006) 429–439. *(Cited on page 64.)*

[44] G. L. Mullen and D. Panario, *Handbook of Finite Fields, First Edition.* Chapman and Hall/CRC, July, 2013. *(Cited on page 44.)*

[45] M. Musa, E. Schaefe and S. Weding, A simplified AES algorithm and its linear and differential cryptoanalysis. *Cryptologia* 27 (April, 2003) 148–177. *(Cited on pages 44 in 48.)*

[46] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information, 10th Anniversary Edition.* Cambridge University Pres, New York 2010. *(Cited on page 5.)*

[47] K. Orhun, Reflection Cryptanalysis of Some Ciphers. *Conference: Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India* (2008) 294–307. *(Cited on page 58.)*

[48] K. N. Patel, I. L. Markov and J. P. Hayes , Optimal synthesis of linear reversible circuits. *Quantum Information and Computation* 8(3) (2008) 282-294. *(Cited on pages 46 in 47.)*

[49] T. Santoli and C. Schaffner, Using Simon's Algorithm to Attack Symmetric-Key Cryptographic Primitives. *Quantum Information and Computation* 17 (2016) . *(Cited on pages 26, 27, 28 in 29.)*

[50] T. R. Shi, C. H. Jin, B. Hu, J. Guan, J- Y. Cui and S. P. Wang, Complete analysis of Simon's quantum algorithm with additional collisions. *Quantum Information Processing* 18 (2019) . *(Cited on page 17.)*

[51] K. Shinagawa and I.Tetsu, Quantum attacks on Sum of Even-Mansour pseudorandom functions. *Information Processing Letters* 106172 (2021) . *(Cited on pages 38, 39 in 59.)*

[52] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring. *Proceeding 35th Annual Symposium of Fundations of Computer Science* 26 (1994) 122–134. *(Cited on pages 2, 75 in 88.)*

[53] D. R. Simon, On the power of Quantum Computing. *SIAM Journal on Computing* 26 (1997) 1474–1483. *(Cited on pages 3, 16, 31 in 88.)*

[54] D. R. Stinson, *Cryptography Theory and Practise (Discrete Mathematics and Its Applications), Third Edition*, Champan & Hall/CRC, Taylor and Francis Group, 2006. *(Cited on pages 1 in 87.)*

[55] E. Strubell, *An Introduction to Quantum Algorithms*, `http://mmrc.amss.cas.cn/tlb/201702/W020170224608150507023.pdf`. (Viewed on: 24. 6. 2021.) *(Cited on page 40.)*

[56] H. Xie and L. Yang, Using Bernstein–Vazirani algorithm to attack block ciphers. *Designs, Codes and Cryptography* 87 (2019) 1161—1182. *(Cited on pages 70, 71, 72 in 73.)*

[57] *Appendix A, Linear Algebra for Quantum Computation*, `https://cds.cern.ch/record/1522001/files/978-1-4614-6336-8_BookBackMatter.pdf`. (Viewed on: 25. 5. 2021.) *(Cited on page 5.)*

[58] National Soviet Bureau of Standards, . *Information Processing System - Cryptographic Protection - Cryptographic Algorithm GOST 28147-89* (1989) . *(Cited on page 57.)*

[59] *QuIC Seminar 11, Simon's Algorithm*, `https://www.ryanlarose.com/uploads/1/1/5/8/115879647/simon.pdf`. (Viewed on: 13. 6. 2021.) *(Cited on pages 11 in 17.)*

[60] The Born rule and its interpretation https://www.math.ru.nl/ landsman/Born.pdf 1182021 *(Cited on page 8.)*

Appendices

APPENDIX A The elements of Galois field $GF(2^4)$

Definition A.1. A *field* is a 3-tiple $(F, +, \cdot)$ such that

1. $(F, +)$ is abelian group with identity 0.
2. $(F/\{0\}, \cdot)$ is abelian group with identity 1.
3. distributivity lows hold: $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$.

Let us denote by $GF(q)$ Galois field of order $q = p^h$, where p is prime. Consider $GF(2^4)$. The polynomial $p(x) = x^4 + x + 1$ is irreducible over $F_2 = \{0, 1\}$, therefore $GF(2^4)$ is equivalent to $F_2/(x^4+x^2+1.)$ Using this we are able to determine all elements of $GF(2^4)$.

The elements of Galois field $GF(2^4)$.

Elements in polynomial form	Elements in binary form	Inverse
1	0001	1
x	0010	$x^3 + 1$
$x + 1$	0011	$x^3 + x^2 + x$
x^2	0100	$x^3 + x^2 + 1$
$x^2 + 1$	0101	$x^3 + x + 1$
$x^2 + x$	0110	$x^2 + x + 1$
$x^2 + x + 1$	0111	$x^2 + x$
x^3	1000	$x^3 + x^2 + x + 1$
$x^3 + 1$	1001	x
$x^3 + x$	1010	$x^3 + x^2$
$x^3 + x + 1$	1011	$x^2 + 1$
$x^3 + x^2$	1100	$x^3 + x$
$x^3 + x^2 + 1$	1101	x^2
$x^3 + x^2 + x$	1110	$x + 1$
$x^3 + x^2 + x + 1$	1111	x^3