
Algerian Republic Democratic and Popular

Ministry of High Education and Scientific Research

**Farhat Abbes Setif-1 University**

**Departement of Computer Science**

# *Master Thesis*

*Field: Computer Science*

*Speciality: Quantum Computing*

Title:

## ”Quantum Key Distribution Inspired by DNA Encoding: A Novel Approach to Secure Communication”

Presented by :

- RIDJALLINE Nabil
- MAYOUF Idriss

Directed by :

- **Dr. ACHIRI Leila**

Jury members :

- Dr. DJEMMAM Safia (MC. Univ. Ferhat Abbas. Sétif 1)
- Dr. KIDAR Halima Saadia (MC. Univ. Ferhat Abbas. Sétif 1)

***Promotion 2024/2025***

## Abstract

This thesis explores the development of a hybrid cryptographic model that integrates quantum cryptography with DNA-based encryption to address the growing demand for secure communication in the face of emerging quantum threats. The study begins with a comprehensive review of classical, modern, and DNA-inspired cryptographic methods, highlighting their theoretical foundations and evaluating their resilience against contemporary security challenges.

Building upon this background, the thesis introduces the fundamental concepts of quantum computing—such as qubits, entanglement, and superposition—which are essential for understanding quantum key distribution (QKD) protocols. Special focus is given to the BB84 protocol, analyzing its operational mechanics, security guarantees, and vulnerability to eavesdropping.

The core contribution lies in the simulation of the BB84 protocol under two scenarios: with and without the presence of an eavesdropper. The generated quantum key is then utilized in a DNA-based encryption and decryption process, wherein classical data is encoded into synthetic DNA sequences using predefined biological mapping rules. This hybrid approach demonstrates enhanced security by combining the physical robustness of quantum mechanics with the structural complexity of DNA encoding.

The study concludes by summarizing key findings, acknowledging current limitations, and proposing future directions for optimizing and implementing the model in practical environments. The results highlight the potential of interdisciplinary cryptographic systems in building resilient security architectures for the post-quantum era.

**Keywords:** Quantum computing, Quantum cryptography, DNA encryption, Quantum key distribution (QKD), BB84 protocol, Secure communication

**الملخص**

يتناول هذا البحث تطوير نموذج تشفير هجين يجمع بين التشفير الكمي **و**التشفير القائم على الحمض النووي(DNA) ، بهدف الاستجابة للحاجة المتزايدة إلى وسائل اتصال آمنة في ظل التهديدات المتصاعدة الناتجة عن تطور الحوسبة الكمومية. يبدأ البحث بمراجعة شاملة لأساليب التشفير الكلاسيكية والحديثة، إضافةً إلى التشفير القائم على ال -DNA، مع تسليط الضوء على الأسس النظرية وآليات العمل ومواطن القوة والضعف لكل منها في سياق التحديات الأمنية المعاصرة.

يقدّم البحث بعد ذلك المبادئ الأساسية للحوسبة الكمومية مثل الكيوبتات، والتشابك الكمومي، والتراكب، والتي تُعد ضرورية لفهم بروتوكولات التشفير الكمي، مع التركيز بشكل خاص على بروتوكول BB84 ، من خلال تحليل آليته التشغيلية، وضماناته الأمنية، وإمكانية تعرضه للتنصت.

أما المساهمة الأساسية للرسالة فتتمثل في محاكاة بروتوكول BB84 في حالتي وجود متنصت (المعروفة بشخصية "إيف") وغيابه، حيث يُستخدم المفتاح الناتج عن التوزيع الكمي في عملية تشفير وفك تشفير البيانات باستخدام ترميز قائم على الـ DNA ، يتم تحويل البيانات إلى تسلسلات DNA صناعية باستخدام قواعد ترميز بيولوجية محددة مسبقًا، مما يضيف طبقة أمنية إضافية من التعقيد البيولوجي.

يختتم البحث بتلخيص النتائج الرئيسية، ومناقشة التحديات والقيود الحالية، واقتراح اتجاهات بحث مستقبلية لتحسين وتطوير هذا النموذج الهجين. وتُظهر النتائج إمكانات الأنظمة التشفيرية متعددة التخصصات في بناء بنى أمنية قوية وموثوقة في عصر ما بعد الكم.

**الكلمات المفتاحية:** الحوسبة الكمومية، التشفير الكمومي، تشفير الحمض النووي(DNA) ، توزيع المفاتيح الكمومية(QKD) ، بروتوكول84BB ، الاتصال الآمن.

**Résumé**
Ce mémoire porte sur le développement d'un modèle cryptographique hybride combinant la cryptographie quantique et le chiffrement basé sur l'ADN, afin de répondre aux besoins croissants en communication sécurisée face aux menaces émergentes de l'informatique quantique. L'étude commence par une revue complète des méthodes de cryptographie classique, moderne et inspirée de l'ADN, en mettant en lumière leurs fondements théoriques, mécanismes de fonctionnement, ainsi que leurs avantages et limites dans le contexte des défis actuels en cybersécurité.
Le mémoire introduit ensuite les principes fondamentaux de l'informatique quantique, tels que les qubits, l'intrication quantique et la superposition, indispensables à la compréhension des protocoles de distribution de clés quantiques (QKD). Une attention particulière est portée au protocole BB84, dont les mécanismes, garanties de sécurité et vulnérabilités face à l'espionnage sont analysés en profondeur.
La contribution principale du travail réside dans la simulation du protocole BB84 dans deux scénarios : en présence et en absence d'un espion (souvent nommé "Eve"). La clé générée est ensuite utilisée dans un processus de chiffrement et déchiffrement fondé sur l'ADN, où les données sont encodées sous forme de séquences d'ADN synthétique selon des règles de correspondance biologique prédéfinies. Cette approche hybride permet de renforcer la sécurité en combinant la solidité physique de la mécanique quantique avec la complexité structurelle du codage ADN.
L'étude se conclut par un résumé des résultats obtenus, une discussion des limitations du modèle proposé et des pistes futures pour son amélioration et sa mise en œuvre concrète. Les résultats mettent en évidence le potentiel des systèmes cryptographiques interdisciplinaires pour bâtir des architectures de sécurité robustes à l'ère post-quantique.

**Mots-clés :** Informatique quantique, Cryptographie quantique, Cryptage ADN, Distribution de clé quantique (QKD), Protocole BB84, Communication sécurisée.

**Table of Contents**

# List of Figures

## General Introduction

The rapid evolution of information technologies, coupled with the rise in cyber threats, has elevated the demand for secure communication frameworks that are resilient against both current and emerging adversarial capabilities. Traditional cryptographic systems, including RSA and Elliptic Curve Cryptography (ECC), rely on the presumed computational intractability of problems such as integer factorization and discrete logarithms. However, the emergence of quantum computing presents a formidable challenge to these systems. Algorithms like Shor's algorithm demonstrate that a sufficiently powerful quantum computer could efficiently solve these problems, rendering many classical encryption schemes obsolete [1].

In response, the field of quantum cryptography has emerged as a physics-based alternative to conventional cryptographic paradigms. Quantum cryptography, and particularly Quantum Key Distribution (QKD), offers unconditional security based on the principles of quantum mechanics, notably the Heisenberg uncertainty principle and the quantum no-cloning theorem. These principles ensure that any attempt at eavesdropping on a quantum channel introduces detectable disturbances, enabling communicating parties to verify the integrity of the key exchange process [2].

The most widely recognized QKD protocol, BB84, utilizes the quantum states of single photons for secure key exchange. Due to the inherent properties of quantum systems, even an adversary with unlimited computational resources cannot intercept the key without altering the quantum state of the transmitted particles and thus revealing their presence. This quantum advantage introduces a new level of security grounded in the fundamental laws of physics rather than computational assumptions.

Parallel to these developments, DNA cryptography has emerged as an unconventional yet promising paradigm in the field of secure information processing. Inspired by the massive parallelism, data density, and inherent biological complexity of DNA molecules, this approach encodes digital information into nucleotide sequences (A, T, C, G). Various mapping schemes are used to convert binary data into DNA sequences, which are then encrypted and transformed using biochemical operations, such as hybridization, polymerase chain reaction (PCR), and restriction enzyme digestion [3].

DNA cryptography provides a highly obscure and physically embedded medium for storing and transmitting information. Its benefits include not only compact storage and computational difficulty but also biological steganography—the ability to conceal information within harmless biological material. This adds a new layer of protection, particularly relevant for long-term archival or covert communication scenarios.

Given the complementary strengths of quantum and DNA cryptography, integrating these two domains offers the potential for multi-layered, interdisciplinary

[1] Bernstein, D. J., Buchmann, J., & Dahmen, E. Post-quantum cryptography. Springer. (2009)

[2] Gisin, N., Ribordy, G., Tittel, W., & Zbinden, H. Quantum cryptography. Reviews of Modern Physics, 74(1), 145–195. (2002).

[3] Leier, A., Richter, C., Banzhaf, W., & Rauhe, H. Cryptography with DNA binary strands. BioSystems, 57(1), 13–22. (2000).

cryptographic systems. While quantum cryptography ensures secure key distribution, DNA-based methods can provide enhanced message-level security, obfuscation, and concealment.

This integration offers resilience across both digital and physical layers, creating a robust security framework that is resistant not only to computational attacks but also to detection and reverse engineering.

The primary motivation of this thesis is to explore the theoretical underpinnings, design principles, and security implications of combining quantum and DNA cryptographic techniques. By analyzing the mechanisms and interactions between quantum key exchange and DNA-based data representation, this research aims to propose a novel hybrid cryptographic model that is practical, scalable, and highly secure against evolving threats.

This thesis is organized into six chapters, each building a foundational and analytical framework that culminates in the proposed hybrid cryptographic model:

- Chapter 1 provides a comprehensive review of classical, modern, and DNA-based cryptographic techniques. It explores the theoretical underpinnings, operational mechanisms, and comparative strengths and weaknesses of each approach in the context of evolving security challenges.
- Chapter 2 presents the foundational principles of quantum computing, including qubit representation, quantum gates, entanglement, and superposition, offering the necessary background for understanding quantum-based cryptographic protocols.
- Chapter 3 delves into the domain of quantum cryptography, with particular emphasis on Quantum Key Distribution (QKD). It includes an in-depth analysis of the BB84 protocol, discussing its operational framework, security guarantees, and susceptibility to eavesdropping.
- Chapter 4 constitutes the core contribution of this thesis. It presents a simulation of the BB84 protocol under two scenarios: with and without an eavesdropper (commonly referred to as "Eve"). Furthermore, it integrates a DNA-based encoding scheme for data encryption and decryption, utilizing the secret key generated via BB84. This hybrid approach is examined for its practical feasibility and enhanced security potential.
- Conclusion: Summarizes the main findings of the thesis, discusses the limitations of the proposed approach, and suggests possible directions for future research to improve and expand the hybrid cryptographic system.

# Chapter 1:

## Introduction

This chapter introduces the foundational principles of cryptography, tracing its evolution from ancient techniques like the Caesar cipher to modern symmetric and asymmetric algorithms. It discusses the structure and security goals of cryptographic systems and highlights the shift from traditional methods to emerging paradigms like DNA-based cryptography.

## Cryptographic Paradigms and Their Evolution

Cryptology is a mathematical science concerned with securing information and analyzing the robustness of methods used for this purpose. It comprises two interdependent branches: cryptography, which focuses on the design of algorithms and protocols for secure communication, and cryptanalysis, which aims to evaluate and potentially break these systems by discovering weaknesses in their structure or implementation. While cryptography seeks to prevent unauthorized access and ensure confidentiality, integrity, and authenticity of data, cryptanalysis plays a crucial role in assessing the resilience of cryptographic schemes by attempting to recover original plaintexts from ciphertexts without access to the corresponding keys.

The discipline of cryptology encompasses both cryptographic construction and cryptanalytic evaluation, forming a unified scientific field dedicated to the protection and analysis of information security systems[4]. Accordingly, the confidentiality of encrypted data ultimately depends on two essential factors: the robustness of the underlying cryptographic algorithm and the secrecy of the cryptographic key.

At the core of cryptography lies the concept of a cryptosystem, which is formally defined by a structured set of elements: the space of all possible plaintexts, the corresponding space of ciphertexts, and the key space, which encompasses all valid cryptographic keys. These components are governed by a pair of mathematically defined functions—encryption and decryption—that transform data in a reversible manner under controlled conditions.

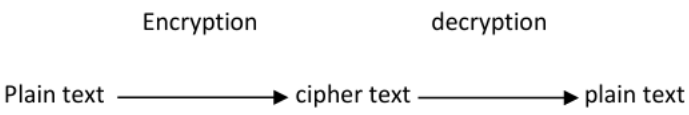



**Figure 1.1. Flow Diagram of Cryptography**

## 1.1. Vocabulary of Cryptography:

[4] G. C. kessler, "An Overview of Cryptography," in Auerbach, 1999.

**Plaintext**: The original, unaltered data intended for transmission or storage prior to any cryptographic processing.

**Encryption**: The process by which plaintext is algorithmically transformed into an encoded format to conceal its content from unauthorized access.

**Ciphertext**: The encrypted, non-readable output produced from the encryption of plaintext, designed to obscure the original information from unintended recipients.

**Decryption:** This is the reverse process of encryption, whereby ciphertext is algorithmically transformed back into its original readable form, known as plaintext, using the appropriate decryption key.

### 1.2. Types of Cryptographic Functions

Cryptographic functions are typically categorized into three main types, according to their method of key utilization[5]:

- Secret Key Functions (Symmetric Cryptography)
- Public Key Functions (Asymmetric Cryptography)

This classification is based on the number and nature of keys involved in each approach.

These distinctions, form the foundation for selecting appropriate cryptographic methods based on specific security requirements.

### 1.3. Cryptanalytic Attacks

Cryptosystems must withstand various adversarial attacks designed to compromise their security properties.

- Passive Attacks: The attacker merely intercepts data without altering it, aiming to gain information covertly.
- Active Attacks: The attacker attempts to modify or disrupt communications, inject fraudulent messages, or impersonate legitimate parties.

Common attack models include:

1. Ciphertext-Only Attack (COA): Attacker has access only to ciphertexts and tries to deduce plaintext or keys.
2. Known-Plaintext Attack (KPA): Attacker possesses some plaintext-ciphertext pairs and exploits this to reveal keys.
3. Chosen-Plaintext Attack (CPA): Attacker can obtain ciphertexts for arbitrary plaintexts of their choosing.

[5] THE CRYPTOGRAPHY GUIDE : TRIPLE DES, [Online]. http://www.cryptographyworld.com/des.htm. [Accessed 05 12 2012].

4. Chosen-Ciphertext Attack (CCA): Attacker can decrypt arbitrary ciphertexts (except the target) to glean key information.

Robust cryptosystems must be designed to resist these attacks under realistic assumptions.

## 1.4. Historical Foundations of Cryptography

### 1.4.1. Ceasar Cipher

Cryptography, strategically utilized since ancient times, began with elementary techniques such as the Caesar cipher—a simple substitution cipher where each letter in the plaintext is shifted by a fixed number of positions in the alphabet. Despite its simplicity, this method laid the groundwork for future cryptographic development. The Caesar cipher, for example, can be formalized mathematically as a permutation over the alphabet, but its historical context reflects an early understanding of secrecy through obfuscation.

| Advantages of Early Cryptography | Disadvantages: |
|---|---|
| • Conceptually simple and easy to implement.<br>• Effective against casual or untrained adversaries | • Vulnerable to frequency analysis and brute-force attacks.<br>• Lack of formal security models and computational rigor.<br>• Keys and encryption methods often static and predictable. |

### 1.4.2. Modern Cryptography

Modern cryptography abandons the security-by-obscurity approach, focusing instead on the protection of secret keys while assuming the encryption and decryption algorithms are publicly known (Kerckhoffs's principle).

The cryptographic key must be:

- Randomly generated with sufficient entropy.
- Long enough to resist exhaustive search attacks.
- Kept strictly confidential to maintain system security.

Modern communication scenarios typically define three parties: Alice (sender), Bob (receiver), and Eve (eavesdropper). The goal is to enable Alice and Bob to communicate securely despite Eve's presence.

Cryptosystems broadly divide into:

- Symmetric-key cryptography: One shared secret key.
- Asymmetric-key cryptography: Two mathematically related keys (public/private).

#### 1.4.2.1. Symmetric-Key Cryptography (Secret Key Cryptography)

Symmetric cryptography uses a single secret key for both encryption and decryption. The key must be securely shared and kept confidential by both communicating parties.

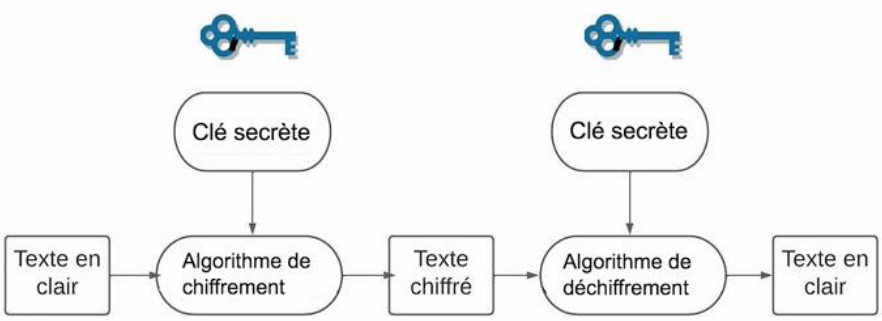



**Fig. 1.2. Principle of Symmetric Cryptography**

A canonical example is the One-Time Pad (OTP), proposed by Gilbert Vernam. OTP encrypts each plaintext bit by XORing it with a random bit from a key of equal length, producing theoretically unbreakable ciphertext if the key is truly random, as long as the key is never reused.

- **One-time pad (OTP)**

One-time pad algorithm was developed towards the end of 19th century from Vernam Cipher. If the key used in OTP is randomly generated and not used more than once, then the algorithm is considered to be completely unbreakable. Previously, the randomly generated keys were shared as a pad of paper, so the sheets could be torn off after the use of the key; thus, the algorithm was termed as one-time pad. The secret key of the algorithm is generally a string of characters or numbers which is at least as long as the longest message to be encrypted. One-time pad is a binary additive stream cipher, where one bit of plain text is encrypted at time by an 'exclusive OR' (XOR) addition with the corresponding bit in the secret key. The keys used for encryption and decryption are the same. This algorithm is explained using an example in **Figure 3**. For cryptographic applications Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) are used and it is more protected than PRNGs. CSPRNG requires passing the next- bit test i.e. if the first $k$ bits of a random sequence are given, there is no algorithm for prediction of the $(k+1)th$ bit with probability of success non-negligibly better than 50% within polynomial time[6]. The CSPRNGs which can pass next-bit test, is capable of passing all other polynomial-time statistical randomness test[7] and can withstand massive hack attacks.

[6] Katz J, Lindell Y. Introduction to modern cryptography. (2ndedn). CRC Press, USA. 2014.

[7] Yao AC. Theory and applications of trapdoor functions. Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Washington DC, USA. 1982.

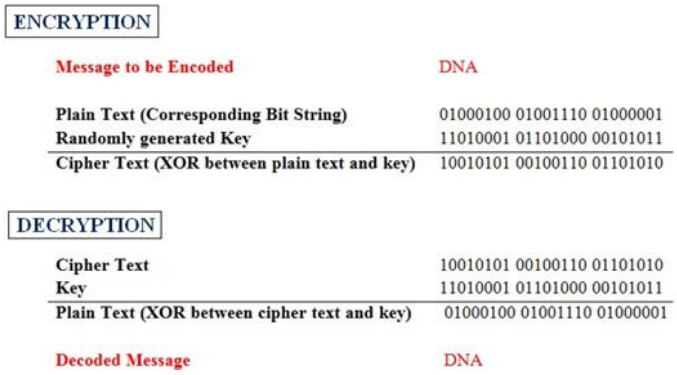



**Figure 1.3. Example of one-time pad method.**

In this system, the sender (Alice) adds a randomly generated key to the plaintext to produce the ciphertext. This encrypted message is then transmitted to the receiver (Bob), who decrypts it by subtracting the same key.

**Example:** Alice sends the message "FERHAT" to Bob and obtains during the encryption process the ciphertext:

| ALICE | ENCRYPTION | | | | | |
|---|---|---|---|---|---|---|
| Message | F | **E** | **R** | **H** | **A** | **T** |
| Plain text | 06 | 05 | 18 | 08 | 01 | 20 |
| Key | 26 | 07 | 13 | 05 | 02 | 11 |
| Cipher text (mod 26) | 06 | 12 | 05 | 13 | 03 | 05 |

Note that the ciphertext is as random as the key. Consequently, they both do not contain any information.

The ciphertext is now sent to Bob, who decrypts the message by subtracting the same key:

| BOB | DECRYPTION | | | | | |
|---|---|---|---|---|---|---|
| Ciphertext | 06 | 12 | 05 | 13 | 03 | 05 |
| Key | 26 | 07 | 13 | 05 | 02 | 11 |
| Plaintext (mod 26) | 06 | 05 | 18 | 08 | 01 | 20 |
| Message | **F** | **E** | **R** | H | A | **T** |

Symmetric algorithms like DES, AES, and Blowfish are widely used for encrypting large volumes of data efficiently.

- **Data Encryption Standard (DES)**

The Data Encryption Standard (DES) is a symmetric-key algorithm for encrypting data. DES was developed in the early 1970s by IBM and was adopted by the National Institute of Standards and Technology (NIST) as a federal standard in 1977. It uses a 56-bit key for encryption and operates on 64-bit blocks of data. DES was widely used for decades, but it is now considered insecure due to advances in computing power, which make it vulnerable to brute-force attacks.[8]

- **AES Encryption**

The increasing power of computers led to the end of DES. It is no longer used when high security is required (such as for military use, classified documents, etc.). For such tasks, the algorithm now preferred is known by its generic name AES (Advanced Encryption Standard), which emerged from a competition organized due to the well-known weaknesses of DES.

Symmetric cryptography is generally faster and more efficient than asymmetric cryptography (subject of next section), making it well-suited for encrypting large amounts of data. However, it requires a secure method of sharing the secret key between parties, which can be a significant challenge, especially in open or unsecured communication environments. While this classical cryptosystem offers unbreakable security, there are notable limitations. The primary challenge is key distribution—Alice and Bob must have access to a highly secure and reliable channel to exchange the key. Furthermore, because the key must be renewed for each individual message, repeating this process for every key becomes impractical and costly. If the key were reused, an eavesdropper (Eve) could gain significant information over time, compromising security.[9]

This type of cryptography is often used in combination with asymmetric cryptography in secure communication protocols, where the asymmetric component helps exchange the secret key securely.

### 1.4.2.2. Asymmetric Cryptography (or Public-Key Cryptography)

Asymmetric cryptography addresses the key distribution problem inherent in symmetric systems by using two distinct keys: a public key for encryption, and a private key for decryption. The public key can be freely distributed, while the private key remains secret.

[8] Schneier, B.. Applied Cryptography: Protocols, Algorithms, and Source Code in C. (1996). Wiley.

[9] Micka P. Letter frequency (English). Available from: http://en.algoritmy.net/article/40379/Letter-frequency-English

In 1976, the seminal work by Diffie and Hellman introduced this paradigm, leading to widespread adoption of public-key cryptosystems such as RSA, Elliptic Curve Cryptography (ECC), and ElGamal. Their reasoning was as follows: for an encryption scheme, the only property required to ensure confidentiality is the difficulty of decryption. There is no inherent reason that the encryption operation itself needs to be difficult. They proposed that each individual have two keys with asymmetric functions: [10]

- A public encryption key, which can be shared openly with anyone who wishes to encrypt a message, and
- A private decryption key, which must remain secret.

Naturally, the two keys are strongly related, since any message encrypted with the public key can only be decrypted using the private key. This concept of asymmetric cryptography is highly appealing, as it is perfectly suited for a large, open network. To send someone a message, you simply retrieve their public key and use it for encryption. Only the holder of the corresponding private key will be able to recover the original plaintext. [11]

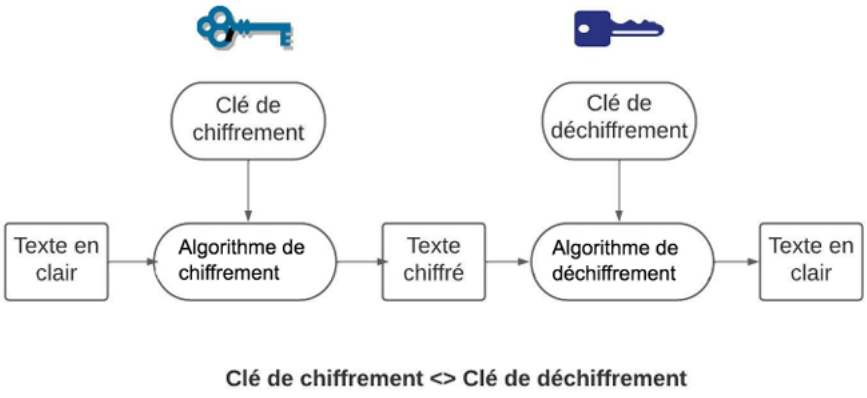



**Figure. 1.4. Principle of Asymmetric Cryptography**

## RSA Protocol:

RSA Protocol is encryption system developed in 1978 by Ronald Rivest, Adi Shamir, and Leonard Adleman is a public-key cryptosystem used for secure data transmission. It is based on the mathematical properties of large prime numbers and modular arithmetic.

## RSA Process

1. Key Generation:

- Select two prime numbers $p$ and $q$.
- Compute $n = p \times q$ and $\varphi(n)$.
- Choose $e$ (the public exponent), ensuring it's coprime with $\varphi(n)$.
- Calculate $d$ (the private exponent) using the modular inverse of $e$ modulo $\varphi(n)$.

[10] Rackoff, Charles et Daniel R Simon. "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack". In: Advances in Cryptology—CRYPTO'91: Proceedings. Springer, p. 433-444. (1992).
[11] Pointcheval , David. "Le chiffrement asymétrique et la sécurité prouvée". In : Habilitation à diriger des recherches, Université Paris VII. (2002).

**2.** Encryption:

- Use the recipient's public key $(e, n)$ to encrypt the plaintext message M using the formula $C = M^e \ mod \ n$.

**3.** Decryption:

- The recipient uses their private key $(d, n)$ to decrypt the ciphertext C using the formula $M = C^d \ mod \ n$, retrieving the original message.

| Advantages: | Disadvantages: |
|---|---|
| • Simplifies key distribution; no need for a secure initial exchange.<br>• Supports encryption, digital signatures, and key exchange protocols.<br>• Suitable for open and large-scale networks such as the Internet. | • Significantly slower and more computationally intensive than symmetric cryptography.<br>• Key sizes are larger, increasing storage and bandwidth requirements.<br>• Security depends on hard mathematical problems; advances in algorithms or quantum computing could undermine some schemes.<br>• Requires robust public key infrastructure (PKI) to manage and verify keys. |

### 1.4.3. DNA Based Cryptography

The continual evolution of cryptographic attacks constitutes a critical challenge in the domain of information security, as cryptographic schemes that are secure today may become susceptible to future cryptanalytic techniques and increasingly sophisticated adversarial capabilities.

By integrating the speed of molecular computation with robust security mechanisms, DNA-based encryption techniques offer a compelling foundation for the development of cryptographic protocols that meet the escalating demands for both security and performance. As such, DNA computing represents a transformative frontier in cryptography, offering a novel response to the dynamic landscape of cyber threats and the future of secure information processing.[12]

The intrinsic ability of DNA molecules to store, process, and transmit information forms the foundation of DNA cryptography. This approach integrates the biochemical properties of nucleic acid sequences with principles of classical cryptography to achieve highly secure data transmission. Rooted in the framework of DNA computing, DNA cryptographic techniques leverage molecular-level operations rather than conventional mathematical encoding, rendering them exceptionally resistant to conventional

[12] Pramanik, Sabari et Sanjit Kumar Setua (2012). "DNA cryptography". In: 2012 7th international conference on electrical and computer engineering. IEEE, p. 551-554.

cryptanalytic attacks. As a result, these methods present a promising frontier for developing cryptosystems with enhanced security and complexity. [13]

### 1.4.3.1. Biological Background

Deoxyribonucleic acid (DNA) is the molecular carrier of genetic information in all living organisms. It is a biological macromolecule composed of repeating structural units known as nucleotides. Each nucleotide consists of a phosphate group, a sugar molecule (deoxyribose), and one of four distinct nitrogenous bases: adenine (A), thymine (T), guanine (G), or cytosine (C).

DNA strands possess inherent directionality, with one end designated as the 5′ and the opposite end as the 3′ (three-prime) [14].

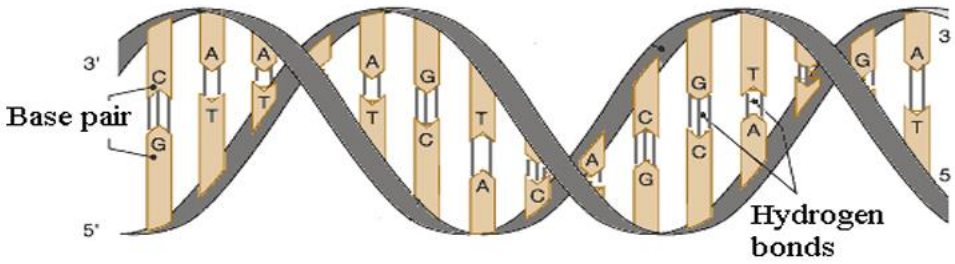



**Figure 1.5. DNA structure**

This structure arises from two complementary DNA strands bound together through hydrogen bonds between specific base pairs: adenine pairs with thymine (A–T), and cytosine pairs with guanine (C–G) as illustrated in Figure 1.5. The orderly pairing of these bases and the antiparallel alignment of the strands result in the stable and compact double-helix configuration.

### 1.4.3.2. Computational Role of DNA

Until 1994, DNA was primarily recognized for its role in encoding biological information. However, a paradigm shift occurred when Leonard Adleman demonstrated that DNA could be harnessed to solve computational problems—specifically, the Hamiltonian Path Problem involving seven vertices. This landmark discovery revealed that DNA not only stores information but can also perform complex computations, giving rise to the field of DNA computing. [15]

[13] Mondal, Mandrita et Kumar S Ray (2019). "Review on DNA cryptography". In: arXiv preprint arXiv: 1904.05528.

[14] G. C. Kessler, "An Overview of Cryptography," 2012.

[15] Vikram, A, S Kalaivani et G Gopinath (2019). "A Novel Encryption Algorithm based on DNA Cryptography". In: 2019 International Conference on Communication and Electronics Systems (ICCES). IEEE, p. 1004-1009.

Following this breakthrough, researchers began exploring the use of DNA sequences as a medium for computation. This computational potential was soon extended into the field of cryptography, leading to the emergence of DNA cryptography.

In addition to DNA, RNA (ribonucleic acid) also plays a critical role in genetic expression. Unlike DNA, RNA contains uracil (U) in place of thymine. Structurally similar to single-stranded DNA (ssDNA), RNA functions in the transmission of genetic information. During transcription, DNA sequences are copied into messenger RNA (mRNA), which is then translated into proteins through the process of translation. Together, these processes constitute the central dogma of molecular biology, as illustrated in Figure 1.6.

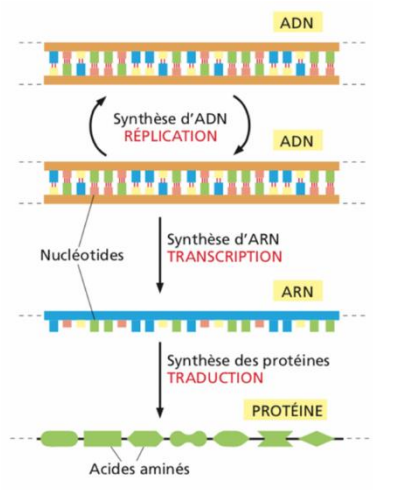



**Figure. 1.6. From DNA to Proteins (Central dogma of molecular biology)[16]**

### 1.4.3.3. DNA Cryptography

DNA cryptography presents a challenge for conventional cryptographic algorithms both theoretically and technologically. The components of DNA cryptography used in the proposed cryptosystem include the following:

- DNA Sequencing: The process of determining the order of bases in DNA is known as DNA sequencing.

- DNA Encoding: The process of converting human-readable plaintext into a DNA base sequence is known as DNA encoding.

In addition to their vast parallelism, DNA molecules also offer enormous storage capacity. One gram of DNA molecules contains around $10^{21}$ DNA bases—equivalent to roughly $10^{8}$ terabytes. It can therefore be concluded that a few grams of DNA could store all the data

[16] Carroll, Sean B. et al. "Introduction à l'analyse génétique ". 6e édition. Bruxelles Paris: DE BOECK SUP. p. 830. 28 mai 2013.

in the world. These advantages of DNA computing have strongly motivated the development of DNA cryptography.

- **Conversion Between Binary Data and DNA Format**

Since DNA strands are represented using the alphabet Σ = {A, C, T, G}, as opposed to binary strings which use Σ = {0,1}, DNA sequences are capable of storing much more information than binary strings. In this work, the binary strings 00, 01, 10, 11 are used to represent A, C, G, and T, respectively. In addition to storing large quantities of data, DNA molecules also allow for compact information storage.[17]

Binary-to-DNA and DNA-to-binary conversions are performed using fixed mappings[18]:

| DNA Code | Binary Form |
|---|---|
| A | 00 |
| C | 01 |
| G | 10 |
| T | 11 |

**Table 1.1. DNA Code to Binary Translation**

Each nucleotide symbol in the DNA sequence corresponds to a 2-bit binary code. This mapping enables seamless encoding and decoding between digital and DNA data formats.

- **OTP Key Selection – DNA Chip**

The study in[19] explores the initial use of DNA for data confidentiality, presenting two primary approaches: DNA-based one-time pad (OTP) encryption and DNA steganography. The OTP method employs both XOR and substitution operations, offering robust and theoretically unbreakable security. While traditional DNA steganography provides limited privacy and is vulnerable to analysis through logical inference, the authors suggest that an enhanced steganographic method significantly improves data concealment. In this modified approach, adversaries—lacking prior assumptions—would remain unaware of the message's existence, thereby achieving a higher level of security.

- **ssDNA as One-Time Pad (OTP) Encryption Key**

In this DNA cryptographic scheme, encryption relies on a one-time pad (OTP) approach using single-stranded DNA (ssDNA) sequences as keys.

[17] Basu, Sayantani et al. "Bio-inspired cryptosystem with DNA cryptography and neural networks". In: Journal of Systems Architecture 94, p. 24-31. (2019).

[18] "Arizona Board of Regents and Center for Image Processing in Education," in Gel Electrophoresis Notes What is it and how does it work, 1999.

[19] T. L. a. J. R. A. Gehani, DNA-Based Cryptography, American Mathematical Society, 2000.

In DNA-based cryptography, the one-time pad (OTP) key is generated based on the size of the plaintext. Specifically, the key is designed to be ten times larger than the binary representation of the data, as each bit is encoded into a 10-nucleotide-long DNA segment. Consequently, for a given message, a corresponding group of single-stranded DNA (ssDNA) sequences is created to match this expanded requirement. This deliberate increase in key length enhances security by ensuring randomness and resistance to cryptanalysis, thereby strengthening the cryptographic robustness of the system.

- **DNA Hybridization Technique: Explanation with Example**

Like conventional cryptographic methods, the DNA hybridization technique [20] involves two main stages: encryption and decryption. In this approach, the plaintext is first converted into binary form. A randomly generated one-time pad (OTP) key is used to encrypt the data. The resulting ciphertext is expressed in DNA format. Decryption involves the reverse process, reconstructing the original binary data and ultimately recovering the plaintext.

**Encryption Process :**

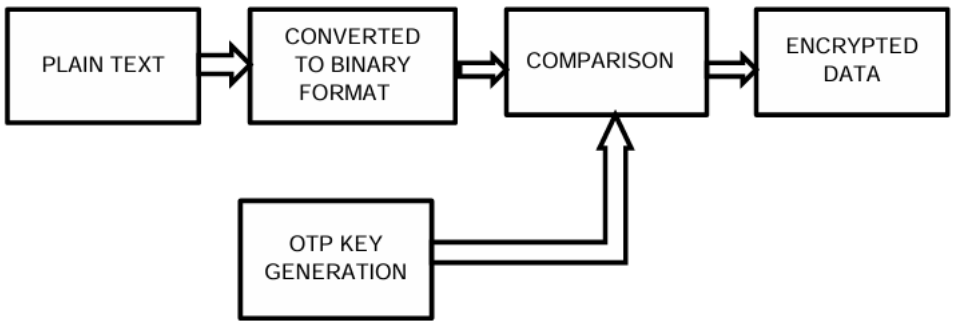



**Figure 1.7. Block diagram for encryption process using DNA hybridization method**

1. Convert Plaintext to Binary For example, the character "A" (ASCII: 65) becomes: A → 01000001
2. Generate OTP Key

   A random DNA-based key is created. Since each binary bit is encoded using a 10-nucleotide segment, an 8-bit message requires an 80-nucleotide key.

**Example:**
Binary key (simplified): 10101100

DNA OTP (representing the key): AGTCAGTCAG... (80 nucleotides)

3. Apply DNA Hybridization Rule

- For every bit '1' in the binary message, the corresponding DNA segment from the OTP is selected and included in the encrypted DNA message.

[20] O. Monica, "DNA Secret Writing Techniques," Romania, 2010.

- For every bit '0', the segment is ignored (no operation).

So, for binary 01000001, only the segments corresponding to the 2nd and 8th positions are included.

4. Resulting Encrypted DNA

   The cipher is a partial DNA sequence constructed from the selected OTP segments.

**Decryption Process**

1. The receiver, who also holds the same OTP key, identifies the positions where encryption was applied (based on the ciphertext structure or predefined rules).
2. The DNA segments are mapped back to their binary positions.
3. The binary string is reconstructed and then converted back to plaintext.

**Example Outcome:**

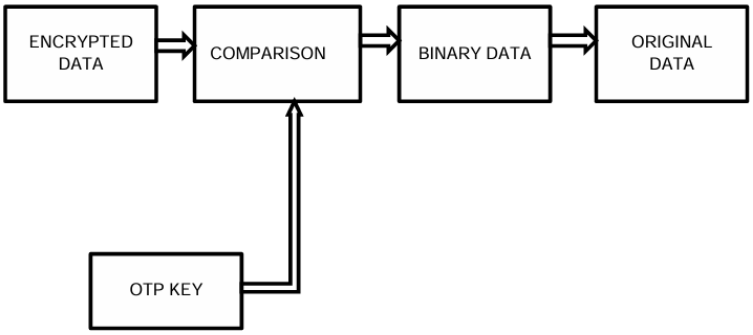



**Figure 1.8. Block diagram for decryption using DNA hybridization**

Encrypted DNA might look like a sequence such as TCGAGTCCGA, composed only of DNA segments aligned with binary '1's. The decryption process would reverse this, recovering the original binary and thus the message "A".

**<u>Algorithm</u>: DNA Hybridization-Based Encryption and Decryption**

**Inputs :**
- Plaintext message P of length $N$ characters
- One-time pad (OTP) DNA key of length $n \times 10$ nucleotides, where $n = 8 \times N$

**Outputs :**
- Encrypted DNA message (ciphertext)
- Decrypted plaintext (during decryption)

**Step 1 :** Binary Conversion
1. Convert each character in the plaintext P to its ASCII representation.
2. Convert the ASCII values to binary.
   - For $N$ characters, the resulting binary string B will have length $n = 8 \times N$ bits.

**Step 2 :** OTP Key Generation

3. Generate a random single-stranded DNA (ssDNA) OTP key.

    - For each binary bit, assign a unique DNA sequence of 10 nucleotides.
    - Total OTP key length = $n \times 10$ nucleotides.
    - The key must be used only once and destroyed after use.

**Step 3 :** Encryption Process

4. Initialize an empty list E to store the encrypted DNA segments.
5. For each bit $b_i$ in the binary message B:

- If $b_i = 1$:
    - Select the corresponding 10-nucleotide segment from the OTP.
    - Generate and store its complementary DNA sequence into E.
- If $b_i = 0$:
    - Do nothing (no sequence added).

6. Concatenate all sequences in E to form the ciphertext DNA message.

**Step 4 :** Decryption Process

7. Receiver must possess an exact copy of the OTP key.
8. Perform DNA hybridization between the encrypted DNA message and the full OTP key:
    - If hybridization (binding) occurs at a segment: interpret as bit 1.
    - If no hybridization: interpret as bit 0.
9. Reconstruct the binary string from hybridization results.
10. Convert the binary string back to ASCII, then to plaintext.

**Step 5 :** OTP Key Destruction

11. Immediately destroy the OTP key after encryption and decryption to ensure perfect secrecy and prevent reuse.

## Conclusion

The chapter emphasizes the limitations of classical cryptographic techniques in the face of evolving threats and sets the stage for newer, interdisciplinary approaches. DNA-based encryption emerges as a promising alternative that combines biological complexity with algorithmic security, offering a potential defense against future cryptanalytic advances.

# Chapter 2:

## Introduction

This chapter lays the groundwork for understanding quantum computing by introducing core concepts such as qubits, superposition, entanglement, and quantum gates. It also explores how quantum operations differ fundamentally from classical logic, providing a conceptual basis for later cryptographic applications.

## Principles of Quantum Computing

Recent progress in quantum computing has led to the identification of numerous practical applications in fields like cryptography, finance, optimization, and machine learning. As this technology rapidly evolves, it becomes increasingly important to understand its fundamental principles. Quantum computing is a field of computer science that takes advantage of quantum phenomena—such as superposition, interference, and entanglement—to solve certain complex problems much faster than classical computers.[21]

While classical computers analyze possible solutions one at a time, quantum systems can represent and evaluate many possibilities simultaneously. This is achieved by creating multi-dimensional quantum states, which can be processed in parallel and then translated into meaningful results.

Among various quantum computing models, the quantum circuit model is the most widely used. It operates through sequences of quantum gates and measurements on qubits, the quantum version of classical bits.[22]

### 2.1. Quantum bit

Inside a classical computer, information is represented in two states, 0 and 1. On the other hand, quantum mechanics allows for the overlapping of two different states. This property is called superposition. Thus, the smallest unit of information in the quantum world, the "quantum" bit called qubit, is represented by a complex vector:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \quad (2.1)$$

[21] M. Santelia, and V. Saling, "What is quantum computing?", IBM, https: //www.ibm.com/quantum-computing/what-is-quantum-computing/, May 2020,

[22] S.P. Jordan, "Quantum Computation Beyond the Circuit Model", Massachusetts Institute of Technology, May 2008.

where $\alpha$ and $\beta$ are called complex probability amplitudes.
The state corresponding to the 0 of the classical bit is

$$\begin{pmatrix}1\\0\end{pmatrix} \tag{2.2}$$

The state corresponding to the 1 of the classical bit is

$$\begin{pmatrix}0\\1\end{pmatrix} \tag{2.3}$$

Since writing column vectors every time is a waste of space, we introduce a simplified notation called Dirac's bra-ket notation. By using Dirac's bra-ket notation, a quantum bit can be written as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \alpha, \beta \in \mathbb{C} \tag{2.4}$$

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \tag{2.5}$$

$$|1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \tag{2.6}$$

We say that $|0\rangle$ and $|1\rangle$ form an orthonormal basis, we can express every quantum state as a linear combination of those two. In fact, $\langle 0|1\rangle = 0$ and we can normalize any quantum state (a normalized quantum state is a state such that $\langle\psi|\psi\rangle = 1$). The $\{|0\rangle, |1\rangle\}$ is not the only basis possible. Two of the most important ones are $\{|+\rangle, |-\rangle\}$ and $\{|+i\rangle, |-i\rangle\}$, which states can be written in the $\{|0\rangle, |1\rangle\}$ basis as:

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle) \text{ and } |\pm i\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm i|1\rangle) \tag{2.7}$$

## 2.2. Projective Measurement

Complex probability amplitude affects the probability distribution of the measurement result. $p_0$, the probability that 0 is measured and $p_1$ the probability that 1 is measured are expressed as the square of the absolute value of the complex probability amplitude.

$$p_0 = |\alpha|^2, \quad p_1 = |\beta|^2 \tag{2.8}$$

$|\alpha|^2 + |\beta|^2$ so that the sum of the probabilities $p_0$ and $p_1$ is 1.
When a measurement is performed, the quantum state changes to the state corresponding to the measurement result.

## 2.3. Bloch Sphere Representation

The qubit state can be rewritten as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\varphi}|1\rangle \tag{2.9}$$

where $\theta$ and $\varphi$ are real numbers that determine the amplitude and relative phase, and the state is normalized.

This form allows us to represent any single-qubit state as a point on the Bloch sphere, a unit sphere where:

- $\theta$ defines the angle from the vertical ($|0\rangle$) axis,
- $\varphi$ defines the rotation around the vertical axis (the phase),

- The sphere visually encodes both the probability amplitudes and phase relationships.

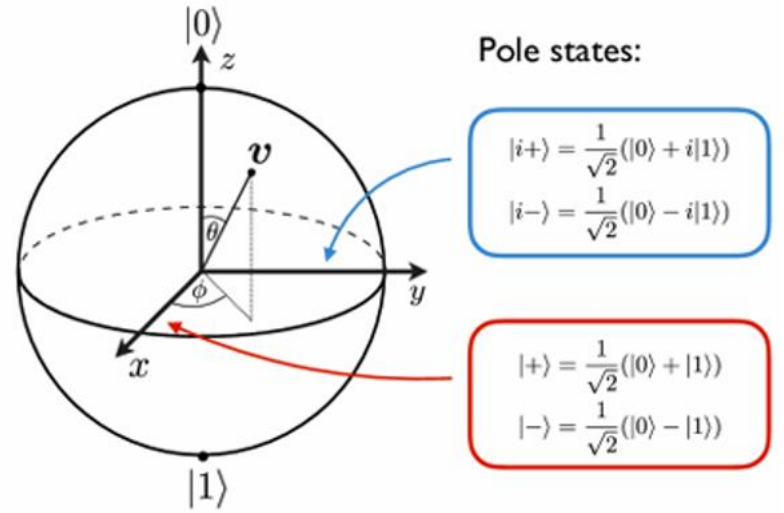



**Figure2.1. Bloch sphere. Source: [23]**

By examining the Bloch sphere, we can better understand key quantum measurement types:

- Z-measurement projects the qubit state onto the z-axis, corresponding to the computational basis $\{|0\rangle, |1\rangle\}$.
- X-measurement projects onto the x-axis, using the basis $\{|+\rangle, |-\rangle\}$.
- Y-measurement projects onto the y-axis, in the basis $\{|+i\rangle, |-i\rangle\}$.

## 2.4. Qubit Properties

Quantum operations are designed to take advantage of the unique behaviors of qubits. While we've already discussed superposition, two other essential properties are interference and entanglement, both of which play a critical role in quantum algorithms.

### 2.4.1. Interference

Superposition allows a qubit to exist in multiple states at once, but that alone doesn't fully distinguish quantum behavior from classical probabilistic systems. Interference is the key difference—it enables quantum states to combine constructively or destructively, affecting the probability of measurement outcomes.

For example, consider the state:

[23] A. Ketterer, "Modular variables in quantum information", October 2016

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) \quad (2.10)$$

By converting this to the $|0\rangle, |1\rangle$ basis, we find: $|\psi\rangle = |1\rangle$

This happens because the $|0\rangle$ terms cancel out due to destructive interference, leaving only $|1\rangle$. This shows how interference can reinforce or eliminate certain results, analogous to how wave amplitudes interact in classical physics. However, in quantum mechanics, these amplitudes correspond to probabilities.

### 2.4.2. Entanglement

When two or more qubits are entangled, their states become inseparably linked—knowing the state of one instantly reveals information about the other, regardless of distance [24]. A classic example is the set of Bell states**:**

$$|\phi_1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (2.11)$$

$$|\phi_2\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \quad (2.12)$$

$$|\phi_3\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \quad (2.13)$$

$$|\phi_4\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \quad (2.14)$$

Even if we can't assign a pure state to an individual qubit in an entangled pair, we can still describe the overall system using a density matrix. Entanglement is foundational for many quantum algorithms, including quantum teleportation.

### 2.4.3. Decoherence

Decoherence is the loss of quantum behavior due to unintended interactions with the environment. When qubits are not perfectly isolated, they may undergo unwanted measurements or become entangled with external systems, causing the loss of superposition and interference effects. [25]

This phenomenon presents a major challenge in building quantum computers. Every qubit is characterized by a decoherence time, which defines how long it can maintain its quantum state before environmental noise causes collapse.

[24] P. Krammer, "Quantum Entanglement: Detection, Classification, and Quantification", Universitat Wien 113 Bibliography. Oc tober 2005.

[25] M. Schlosshauer, "Quantum Decoherence", Physics Reports, Vol. 831, pp. 1-57, DOI 10.1016/j.physrep.2019.10.001. November 2019.

## 2.5. Multiqubit representation

The state of $n$ classical bits is represented by $0, 1$ numbers, and the total number of patterns is $2^n$. Since quantum mechanics allows superposition states of all these patterns, the state of $n$ quantum bits $|\psi\rangle$ is described as a tensor product of the n quantum states so, by $2^n$ complex probability amplitudes like $c_{00\dots0}$, $c_{00\dots1}$, and so on.

$$|\psi\rangle = c_{00\dots0}|00\dots0\rangle + c_{00\dots1}|00\dots1\rangle + \cdots c_{11\dots1}|11\dots1\rangle = \begin{pmatrix} c_{00\dots0} \\ c_{00\dots1} \\ \vdots \\ c_{11\dots1} \end{pmatrix} \quad (2.15)$$

Where: $|q_0 q_1 \dots q_{n-1}\rangle \equiv |q_0\rangle \otimes |q_1\rangle \otimes \dots \otimes |q_{n-1}\rangle$ and $\otimes$ is the tensor product.

note that the Complex probability amplitudes are normalized.

$$\sum_{i_1 i_2 \dots i_n} \left|c_{i_1 i_2 \dots i_n}\right|^2 = 1 \quad (2.16)$$

When the quantum state of $n$ quantum bits are measured, we get bit array $i_1 i_2 \dots i_n$ randomly with the following probability.

$$p_{i_1 i_2 \dots i_n} = \left|c_{i_1 i_2 \dots i_n}\right|^2 \quad (2.17)$$

and the post measure state is $|i_1 i_2 \dots i_n\rangle$.

Thus, the state of an $n$ qubit must be described by a $2^n$ dimensional complex vector that is exponentially large with respect to $n$**,** and here the difference between a classical bit and a qubit is strikingly apparent. The operations on $n$ qubit systems are represented as unitary matrices of $2^n \times 2^n$ dimensions. In other words, a quantum computer is a computer that unitarily transforms a complex vector of exponential size with respect to the number of qubits according to the laws of physics.

## 2.6. Quantum Gates

Quantum gates are the building blocks of quantum circuits, enabling transformations of qubit states. These transformations are governed by Schrödinger's equation **[26]** and are represented by unitary matrices, which are reversible. That is, if a gate is represented by a matrix $U$, its inverse is its conjugate transpose $U^\dagger$, ensuring that quantum operations preserve the normalization of the state:

$$UU^\dagger = \mathbb{1} \quad (2.18)$$

These gates can be visualized not only as matrices but also as rotations on the Bloch sphere.

[26] C.P. Williams, "Explorations in Quantum Computing", Springer, ISBN 978-1-84628 887-6, 1997.

### 2.6.1. Single-Qubit Gates

Some of the most common single-qubit gates include the Pauli gates and the Hadamard gate:

- Pauli-X (bit-flip):

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

- Pauli-Y (bit and phase-flip):

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$$

- Pauli-Z (phase-flip):

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

- Hadamard Gate ($H$): creates superposition

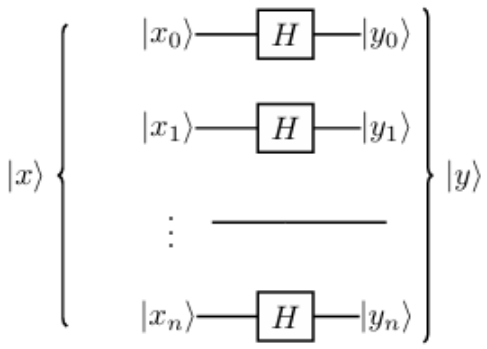
$$Z = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This gate maps:

- $H|0\rangle = |+\rangle \qquad H|+\rangle = |0\rangle$
- $H|1\rangle = |-\rangle \qquad H|-\rangle = |1\rangle$

For n qubits, the Hadamard gate can be extended as $H^{\otimes n}$, producing a uniform superposition of all $2^n$ states, a key step in algorithms like Shor's algorithm:

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




### 2.6.2. Multi-Qubit Gates

Quantum entanglement and interaction between qubits are introduced through multi-qubit gates:

- CNOT (Controlled-NOT): Applies an $X$-gate to a target qubit if the control qubit is in state $|1\rangle$.

Used to create entangled states such as:

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

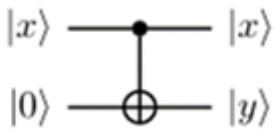



**Figure 2.2. General application of the CNOT gate**

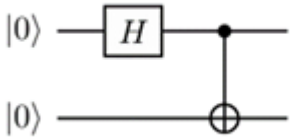



**Figure 2.3. CNOT gate with a qubit in superposition as control qubit**

- Toffoli Gate: Is 3-qubit gate with two control qubits and one target. It performs a NOT on the target only if both controls are $|1\rangle$. It can be decomposed using 2-qubit gates and is essential for universal reversible computing.

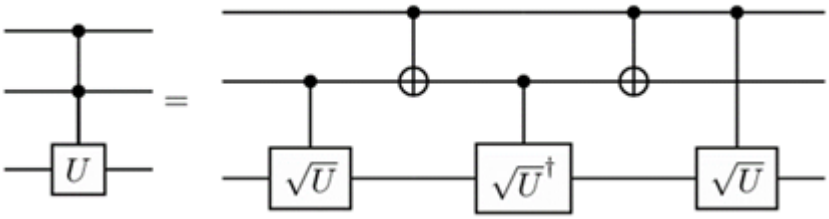



**Figure 2.4. Decomposition of a Toffoli using 2-qubit gates**

- **SWAP Gate:** Exchanges the states of two qubits. Like the Toffoli, it can be built using combinations of simpler 2-qubit gates.

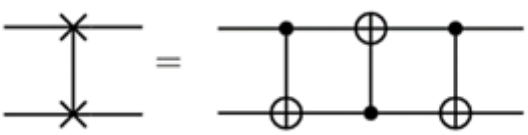



**Figure 2.5. Decomposition of a Swap gate using 2-qubit gates**

## Conclusion

Quantum computing represents a paradigm shift in computation and information processing. With exponentially large state spaces and interference-based computation, quantum systems open new possibilities for secure communication. These properties form the technical backbone of quantum cryptographic protocols discussed in later chapters.

# Chapter 3:

## Introduction

This chapter explores the theory and practice of quantum cryptography, focusing on Quantum Key Distribution (QKD) as a secure communication solution in the quantum era. It explains the principles behind the BB84 protocol, the quantum no-cloning theorem, and Heisenberg's uncertainty principle, which ensure security against eavesdropping.

## Quantum Cryptography and the BB84 Protocol

### 3.1. Quantum cryptography fundamentals

On a wider context, quantum cryptography is a branch of quantum information processing, which includes quantum computing, quantum measurements, and quantum teleportation. Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems. As we known, it is interesting to learn that one decade before people realized that a quantum computer could be used to break public key cryptography, they had already found a solution against this quantum attack – quantum key distribution (QKD). Based on the fundamental principles in quantum physics, QKD provides an unconditionally secure way to distribute random keys through insecure channels. In this chapter, we will introduce the fundamental principles behind various QKD and present the state-of-the art quantum cryptography technologies.

#### 3.1.1. One-time-pad and key distribution problem

In conventional cryptography, an unbreakable code does exist. It is called the one-time-pad and was invented by Gilbert Vernam in 1918 [27]. Therefore, the one-time-pad simply shifts the problem of secure communication to the problem of key distribution. This is the key distribution problem. The one of possible solution to the key distribution problem is public key cryptography.

Quantum mechanics can provide a solution to the key distribution problem. In quantum key distribution, an encryption key is generated randomly between Alice and Bob by using non-orthogonal quantum states. In quantum mechanics there is a quantum no-cloning theorem, which states that it is fundamentally impossible for anyone including an eavesdropper to make an additional copy of an unknown quantum state. Therefore, any attempt by an eavesdropper to learn information about a key in a QKD process will lead to disturbance, which can be detected by Alice and Bob who can, for example, check the bit error rate of a random sample of the raw transmission data.

[27] Vernam, G. S. Cipher Printing Telegraph Systems for Secret Wire and Radio Tele graphic Communications," American Institute of Electrical Engineers, Transactions of the, vol. XLV, 295-301, (1926).

### 3.1.2. Quantum No-Cloning Theorem (Wootters, Zurek, Dieks, 1982):

It is impossible to create an exact copy of an unknown quantum state. This fundamental principle has far-reaching consequences in quantum computing and quantum information science.

**Proof:** Suppose the state of a quantum system $A$, which we wish to copy, is $|\psi\rangle_A$. In order to make a copy, we take a system $B$ with the same state space and initial state $|e\rangle_B$. The composite system is then described by the tensor product, and its state is $|\psi\rangle_A \otimes |e\rangle_B \equiv |\psi\rangle_A|e\rangle_B$.

to manipulate the composite system, we could control the Hamiltonian of the system, and thus the time evolution operator U up to some fixed time interval, which yields a unitary operator. Then $U$ acts as a copier provided that

$$U|\phi\rangle_A \otimes |e\rangle_B = |\phi\rangle_A \otimes |\phi\rangle_B$$

Since $U$ is unitary, it preserves the inner product:

$$\langle e|_B\langle\phi|_A|\psi\rangle_A|e\rangle_B = \langle e|_B\langle\phi|_A U^\dagger U|\psi\rangle_A|e\rangle_B = \langle\phi|_B\langle\phi|_A|\psi\rangle_A|\psi\rangle_B$$

and since quantum mechanical states are assumed to be normalized, it follows that $\langle\phi|\psi\rangle = \langle\phi|\psi\rangle^2$.

This implies that either $|\phi\rangle = |\psi\rangle$ or $|\phi\rangle$ is orthogonal to $|\psi\rangle$.

However, this is not the case for two arbitrary states. While orthogonal states in a specifically chosen basis $\{|0\rangle, |1\rangle\}$, for example, $|\phi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|\psi\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ fit the requirement that $\langle\phi|\psi\rangle = \langle\phi|\psi\rangle^2$, this result does not hold for more general quantum states. Apparently $U$ cannot clone a general quantum state.

The quantum no-cloning theorem is a fundamental consequence of the linearity of quantum mechanics. Closely related is the principle that any measurement which reveals information about a quantum state generally disturbs it, unless the possible states are mutually orthogonal.

At first glance, the impossibility of perfectly copying an unknown quantum state appears to be a limitation. However, this constraint can be turned into an advantage: it forms the basis of unconditionally secure quantum key distribution, where any eavesdropping attempt necessarily disturbs the system, thus revealing the intrusion.

### 3.1.3. Heisenberg's Uncertainty Principle (HUP)

HUP is a cornerstone of quantum mechanics, revealing a fundamental limit to the precision with which certain pairs of physical properties—such as position and momentum—can be simultaneously measured. This principle reflects not a limitation of experimental technique, but a fundamental property of nature arising from the wave-like behavior of quantum particles.

Suppose $A$ and $B$ are two Hermitian operators, and $|\psi\rangle$ is a quantum state. Suppose $\langle\psi|AB|\psi\rangle = x + iy$, where $x$ and $y$ are real. Note that $\langle\psi|[A,B]|\psi\rangle = 2iy$ and $\langle\psi|\{A,B\}|\psi\rangle = 2x$. This implies that

$$|\langle\psi|[A,B]|\psi\rangle|^2 + |\langle\psi|\{A,B\}|\psi\rangle|^2 = 4|\langle\psi|AB|\psi\rangle|^2$$

By the Cauchy-Schwarz inequality $|\langle\psi|AB|\psi\rangle|^2 \leq 4\langle\psi|A^2|\psi\rangle\langle\psi|B^2|\psi\rangle$, and since,

$$|\psi\rangle_{AB} = \sum_{i,j} c_{ij}\,|i\rangle_A \otimes |j\rangle_B$$

and dropping a non-negative term gives

$$|\langle\psi|[A,B]|\psi\rangle|^2 \leq 4\langle\psi|A^2|\psi\rangle\langle\psi|B^2|\psi\rangle$$

Suppose $C$ and $D$ are two observables. Substituting $A = C - \langle C\rangle$ and $B = D - \langle D\rangle$ into the last equation, where the average value of the observable $C$ is often written $\langle C\rangle = \langle\psi|C|\psi\rangle$ and similar to $D$, we obtain Heisenberg's uncertainty principle as it is usually stated

$$\Delta(C).\Delta(D) \geq \frac{|\langle\psi|[C,D]|\psi\rangle|}{2}$$

Quantum communication enables the secure transmission of information using quantum states, typically encoded in photons. Its security relies on the Heisenberg Uncertainty Principle (HUP): any eavesdropping attempt disturbs the quantum system, making detection unavoidable. Unlike classical uncertainty, quantum uncertainty is intrinsic—particles lack definite properties until measured. This principle underlies quantum cryptographic protocols, such as Quantum Key Distribution (QKD), which ensure information-theoretic security.

### 3.2. Quantum key Distribution

In the 1980s, Charles H. Bennett and Gilles Brassard built upon Wiesner's ideas to address the key distribution problem, leading to the introduction of the BB84 protocol in 1984 **[28]**. Quantum Key Distribution (QKD) enables secure key generation over untrusted channels, producing keys independent of any input—a feat unattainable by classical means. While QKD does not replace all cryptographic primitives (e.g., authentication), it offers fundamentally new security guarantees.

In Quantum Key Distribution (QKD), the two parties (Alice and Bob) measure quantum states and then exchange classical information to determine valid key bits, discarding incompatible results during a process called sifting. They perform error correction and estimate the potential information an eavesdropper may have. If this information exceeds a certain threshold, the process is aborted to ensure secrecy. If it is below the threshold, privacy amplification is applied to eliminate any remaining

[28] Bennett, C. H, & Brassard, G. Quantum cryptography: Public key distribution and coin tossing}," in Proceedings of IEEE International Conference on Computers, Sys tems, and Signal Processing, 175. India, (1984).

information, resulting in a shared secret key. Part of the classical communication must be authenticated to prevent man-in-the-middle attacks, with a small probability of protocol failure.

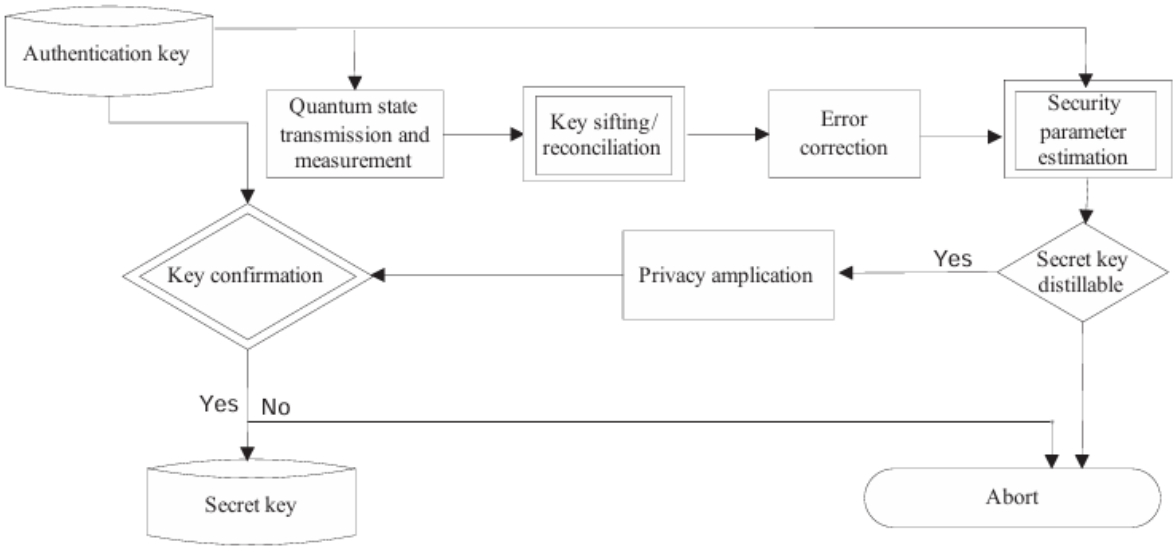



**Figure 3.1. Flow chart of the stages of a quantum key distribution protocol. Stages with double lines require classical authentication.** [29]

### 3.2.1. Types of Quantum key distribution

There are different kinds of quantum key distribution protocols:

**1.** Heisenberg based protocols use the fact that measurement of a quantum state changes the state or disturbs it. The eavesdropper introduces errors in the transmission that can be detected by the communicating parties. Example: BB84, B92 protocols.

**2.** Entanglement based protocols use property of entanglement for secure transmission. Information only comes into existence when entangled particles are measured. The eavesdropper only chance is to introduce its own particles into the channel. These extra particles violate bells' inequalities which can be detected. Entanglement based protocols are further strengthen by monogamy of entanglement. Example: Ekert, Device independent protocols.

The best-known protocol for QKD is the Bennett and Brassard protocol (BB84).

### 3.2.2. Cryptographic Method

[29] Stebila, D, Mosca, M, & Lütkenhaus, N. The Case for Quantum Key Distribution," Quantum Communication and Quantum Networking, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering A. Ser gienko, S. Pascazio and P. Villoresi, eds., Springer Berlin Heidelberg, 283-296. (2010).

Based on two communication channels: A quantum channel, which transmits polarized states to generate the quantum key and a classical channel, which is used for key reconciliation and for sending the encrypted message in classical form. Regarding the intruder Eve: she can listen to the communication transmitted over the quantum channel but cannot alter it. In contrast, the information flow on the classical channel is completely open and can be manipulated, altered, or transformed [30]. In the first part of the cryptographic method, quantum key generation using the BB84 protocol will be employed.

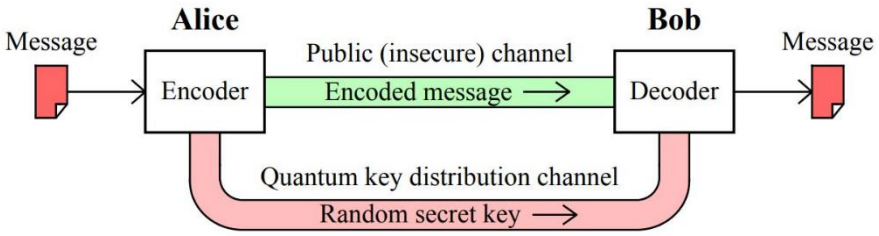



**Figure 3.2. Encryption using Quantum Cryptography**

- A classical channel could be a communication wire like a telephone line where electrical signals represent bits or encoded information we send.
- A quantum communication channel can be a fiber optic cable (as quantum cryptography is implemented using polarization scrambling) through which we can send individual photons (particles of light). The polarization states are used to represent qubits.

### 3.2.3. The BB84 QKD protocol

In this protocol, Alice sends Bob a random sequence of bits encoded using two polarization bases that generate four qubits. These qubits are transmitted through the quantum channel. The four states form two polarization bases with quantum states:

- Horizontal $|\rightarrow\rangle$
- Vertical $|\uparrow\rangle$
- Diagonal right $|\nearrow\rangle$
- Diagonal left $|\nwarrow\rangle$

These states correspond to two polarization bases, $B_{HV}$ (Horizontal-Vertical) and $B_X$ (Diagonal), allowing Alice to send either a 0 or a 1 encoded in any of the states $|\rightarrow\rangle$, $|\uparrow\rangle$, $|\nearrow\rangle$, $|\nwarrow\rangle$. The information is transmitted through a quantum channel, which can be either an optical fiber or free space.

The BHV and Bx bases satisfy inner product conditions between states. States from different bases are not orthogonal, meaning that $\langle B_{HV}|B_X\rangle \neq 0$. This ensures that a state

[30] Song-Kong Chong and Tzonelih Hwang. Quantum key agreement protocol based on BB84. Optics Communications, 283(6):1192–1195, 2010.

is completely determined when projected onto its correct basis, but if it is projected onto the other (incorrect) basis, the result will be random, as shown in Figure 3.3.

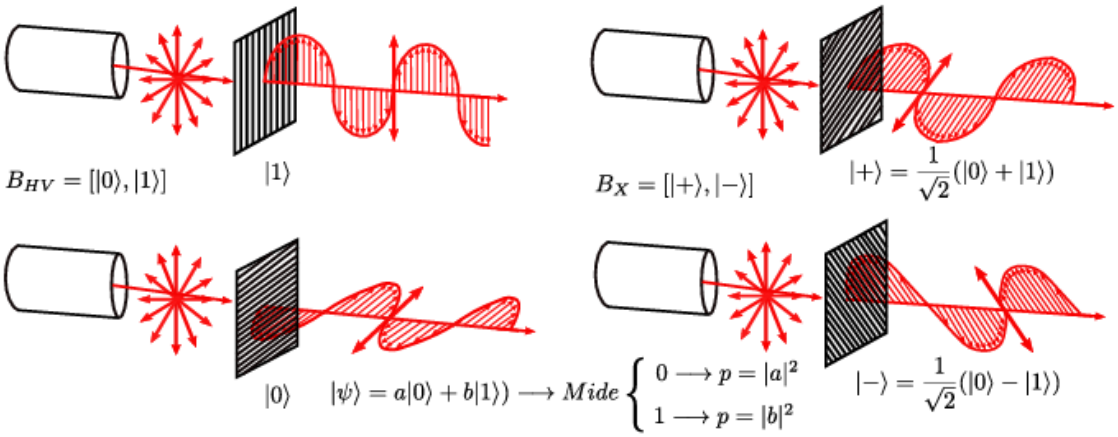



**Figure 3.3. Polarization Bases of the BB84 Protocol: The figure shows the bases and polarization states used in the BB84 protocol to generate the quantum key.**

The generation and distribution of quantum keys using the BB84 protocol is carried out through the following ordered sequence of steps:

1. Alice generates a random string consisting of zeros and ones, such that: $a_1, a_2, \dots, a_n$ with $a_n \in 0, 1$
2. For each bit in the string, Alice randomly selects one of two bases:

- Using the $B_{HV}$ base, a 0 is encoded as $|0\rangle$ (horizontal polarization) and a 1 as $|1\rangle$ (vertical polarization).
- Using the $B_X$ base, a 0 is encoded as $|+\rangle$ (45° polarization), and a 1 as $|-\rangle$ (−45° polarization).

  This process generates a sequence of bases, also represented by $0s$ and $1s$, $\alpha_1, \alpha_2, \dots, \alpha_n$ with $\alpha_n \in 0, 1$.

  In summary, for the model:

- Alice can use the $B_X$ base $\{|+\rangle, |-\rangle\}$, representing 45° and −45° polarizations, to encode the states:
  - $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
  - $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
- Or she can use the $B_{HV}$ base $\{|0\rangle, |1\rangle\}$, representing horizontal and vertical polarization, to encode the states:
  - $|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$
  - $|1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle)$

3. Bob receives the sequence $\alpha_1, \alpha_2, \ldots, \alpha_n$ sent by Alice and randomly chooses one of the two bases ($B_{HV}$ or $B_X$) to measure each of the received quantum states.

- If Bob uses $B_{HV}$, he can measure:
    - $|0\rangle$ with probability $p = 1$
    - $|1\rangle$ with probability $p = 1$
    - But if the state is $|+\rangle$ or $|-\rangle$, he measures $|0\rangle$ or $|1\rangle$ with equal probability $p = 0.5$

- If Bob uses $B_X$, and the state is $|0\rangle$ or $|1\rangle$, he measures $|+\rangle$ or $|-\rangle$ with probability $p = 0.5$
- But if the state is already $|+\rangle$, he measures it with $p = 1$, and the same for $|-\rangle$

4. Alice and Bob discard all events where they use different bases for a signal.
5. To test for tampering, Alice randomly chooses a fraction, $p$, of all remaining events as test events. For those test events, she publicly broadcasts their positions and polarizations.
6. Bob broadcasts the polarizations of the test events.
7. Alice and Bob compute the error rate of the test events (i.e., the fraction of data for which their value disagrees). If the computed error rate is larger than some prescribed threshold value, say 11%, they abort. Otherwise, they proceed to the next step.
8. Alice and Bob each convert the polarization data of all remaining data into a binary string called a raw key (by, for example, mapping a vertical of 45-degrees photon to "0" and a horizontal or 135-degrees photon to "1"). The can perform classical postprocessing such as error correction and privacy amplification to generate a final key.

The basic idea of the BB84 QKD protocol is beautiful and its security can be intuitively understood from the quantum no-cloning theorem. On the other hand, to apply QKD in practice, Alice and Bob need to find the upper bound of Eve's information quantitatively, given the observed quantum bit error rate (abbreviated QBER) and other system parameters. This is the primary goal of various QKD security proofs and it had turned out to be extremely difficult.

### 3.2.4. Eavesdropping Detection and Security Assurance in Quantum Key Distribution

Since this protocol is fundamentally designed to ensure security against potential attacks, it is crucial to consider the actions of an eavesdropper—typically referred to as Eve. Due to the No-Cloning Theorem, Eve cannot copy a quantum state sent by Alice and then forward an identical copy to Bob without being detected. The only option available to Eve is to measure each qubit sent by Alice. However, this measurement collapses the qubit's state. Since Eve does not know the basis Alice used, she must guess randomly, giving her only a 50% chance of choosing the correct basis for each qubit.

After measurement, Eve prepares a new qubit based on her result and sends it to Bob, who remains unaware of any interference. To detect such intrusions, Alice reveals a random subset (k bits) of her original basis choices. Bob compares his corresponding bits with Alice's. Due to Eve's intervention, discrepancies will occur approximately 50% of the time when she measured in the wrong basis.

Alice and Bob then assess how many of these $k$ qubits match. A higher mismatch rate indicates a higher probability of eavesdropping. If the error rate exceeds a predefined threshold, they discard the key. By increasing the number of bits used in the comparison, Alice and Bob can achieve higher confidence in detecting any tampering—offering unconditional security**.**

Statistically, Eve measures in the correct basis 50% of the time. When she chooses the wrong basis and sends a qubit to Bob, there's still a 50% chance Bob will measure the bit Alice originally intended. Thus, the probability that Eve goes undetected for each qubit is:

$$P(\text{Eve not detected}) = \left(\frac{3}{4}\right)^N$$

Where N is the number of qubits Alice and Bob compare. To ensure a specific confidence level in detecting Eve, we can calculate the required number of comparisons using:

$$N = \frac{log\big(P(\text{Eve not detected})\big)}{log\left(\frac{3}{4}\right)}$$

Where $P(\text{Eve not detected})$ is the maximum allowable probability that Eve goes undetected.

## Conclusion

The BB84 protocol is shown to be a robust and practically implementable method for secure key exchange. Through quantum measurement and base selection, it enables detection of intrusions and key validation. This forms the cryptographic core used in the hybrid model developed in subsequent chapters.

## Chapter: 4

### Introduction

This chapter presents the practical implementation of the BB84 protocol using Qiskit, integrating it with a DNA-based encryption system. It outlines a three-phase simulation model involving quantum key generation, encryption/decryption using DNA, and an eavesdropping detection phase to assess security.

### Simulation and Integration

*BB84 Protocol Implementation with DNA Encoding for Enhanced Post-Quantum Security*

In this work, a simulated quantum cryptography model was implemented using the Qiskit programming language. It integrates the generation and distribution of quantum keys through the BB84 protocol, which is the most widely used in quantum cryptography, this will be integrated with the DNA encoding, which provides perfect security under conditions of a truly random key. These two systems together will allow for encoding information with high security.

To generate a secure quantum key, it must have a length equal to or greater than that of the message, since if the key is shorter than the message, it is possible for an intruder to intercept the key and decrypt the information.

To simulate the generation and distribution of quantum keys, the Python programming language and the Qiskit environment are used, along with its various libraries.

In the development of the model, three agents interact:

- Alice, representing the sender
- Bob, representing the receiver
- Eve, representing an eavesdropper on the quantum communication channel

Alice and Bob will be responsible for generating the quantum key, while Eve will attempt to compromise the symmetric key system.

The model is divided into three phases to enable better development of the simulation:

- Phase 1: In this phase, the generation and distribution of the quantum cryptographic key is carried out between Alice and Bob using the BB84 protocol.
- Phase 2: In this phase, the encryption and decryption of information between Alice and Bob is carried out using the DNA ecoding and the quantum cryptographic key generated in Phase 1.

- Phase 3: In this phase, the eavesdropper Eve is introduced into the quantum communication channel to verify whether Alice and Bob can detect the intruder and decide whether the quantum key is secure enough to proceed or whether to abort the process due to security breaches.

Qiskit primarily runs on the Python programming language, though there are also versions available for Swift and JavaScript.

### 4.1. Simulation of Quantum Key Distribution using Qiskit

Imagine two people, Alice wants to communicate a secret message to Bob over an insecure medium such as the Internet. Since the Internet is publicly available for anyone to intercept, a special encryption scheme must be developed such that Bob can only access Alice's message with a special "key". We will focus on the BB84 algorithm, which is a symmetric key distribution algorithm enabling Alice to securely communicate her message to Bob.

**Step 1:** Generate Binary Strings

Alice will generate two strings (consisting of 0s and 1s). One string will encode the basis for each qubit and the other string will encode its state.

0 in the hadamard basis will be represented as state $|+\rangle$ and 1 in the hadamard basis is represented as state $|-\rangle$. The hadamard basis can be seen as the result of applying a hadamard gate on the computational basis $|0\rangle$ or $|1\rangle$. The computational basis will be represented by 0 and hadamard basis by 1. Let us suppose that Alice wants to generate a 32-bit string. The number of qubits needed is 32. We will generate two random strings that represent Alice's state and the basis that Alice uses for encoding the state (the information about which basis Alice uses for encoding the state is very important for Bob to decrypt the message using the key)

Alice's State: [1 1 0 1 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 0 0]
Alice's Bases: [0 0 1 1 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0]
Bob's Bases: [0 1 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0]

**Step 2:** Creating the circuit using BB84 protocol

We will use this algorithm to construct the circuit

1. Whenever Alice wants to encode 1 in a qubit, she applies an X gate to the qubit. To encode 0, no action is needed as the qubits in qiskit are by default initialized to state 0 and applying an X gate is the equivalent of applying NOT.

2. Wherever she wants to encode it in the Hadamard basis, she applies an hadamard gate.

3. She then *sends* the qubits to Bob

4. Bob measures the qubits according to his binary string. To measure a qubit in the Hadamard basis, he applies an hadamard gate to the corresponding qubit and then performs a measurement on the computational basis

## Specific Alice's Bits and Basis

Entrée [1]:

```
from qiskit import QuantumCircuit
from qiskit_aer import Aer

# Use your specific data
alice_bits =  [1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0]
alice_bases = [0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0]

n = len(alice_bits)

# Generate random bases for Bob (you can replace this too if needed)
import random
bob_bases = [random.randint(0, 1) for _ in range(n)]

# Alice's circuit
alice_circuit = QuantumCircuit(n)
for i in range(n):
    if alice_bits[i] == 1:
        alice_circuit.x(i)
    if alice_bases[i] == 1:
        alice_circuit.h(i)

# Bob's circuit
bob_circuit = QuantumCircuit(n, n)
for i in range(n):
    if bob_bases[i] == 1:
        bob_circuit.h(i)
    bob_circuit.measure(i, i)
```

Entrée [5]:

```
full_circuit.draw('mpl')
```

Out[5]:

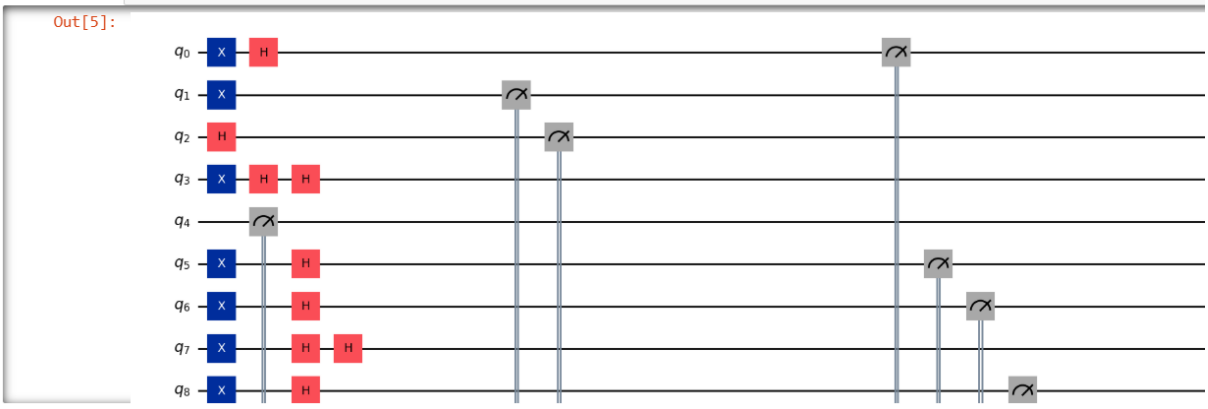



**Figure 4.1. Full circuit (Alice-circuit and Bob-circuit)**

**Step 3:** Creating the key

Alice and Bob only keep the bits in the strings where the bases match. For example, in the example above only if Alice's encoding basis is 0 and Bob's measuring basis is 0 or its 1 and 1 the bits are kept.

Since Bob determines the state that Alice has sent him by randomly choosing, for each of the received states, one of the two possible bases $\langle B_{HV}|\, B_X\rangle \neq 0$ in this process of measurement and storage, it is predictable that Bob will choose the same basis as Alice 50% of the time. Since Alice sends a 0 encoded in the horizontal state of the system $|\leftrightarrow\rangle$, Bob can measure the quantum state $|\leftrightarrow\rangle$ in the $B_{HV}$ basis, or the quantum state $|\nearrow\rangle$ in the $B_X$ basis. Similarly, if Alice sends a 1 encoded in the state $|\updownarrow\rangle$, Bob can measure the state $|\updownarrow\rangle$ or the state $|\nwarrow\rangle$, depending on whether he uses the $B_{HV}$ or $B_X$ basis, as shown in Table 4.1.

| **State** | **Measurement with $B_{HV}$** | **Measurement with $B_X$** |
|---|---|---|
| $\|0\rangle$ | $\|0\rangle$ with $p = 1$ | $\|+\rangle$ with $p = 1/2$, and $\|+\rangle$ with $p = 1/2$ |
| $\|1\rangle$ | $\|1\rangle$ with $p = 1$ | $\|+\rangle$ with $p = 1/2$, and $\|+\rangle$ with $p = 1/2$ |
| $\|+\rangle$ | $\|0\rangle$ with $p = 1/2$, and $\|1\rangle$ with $p = 1/2$ | $\|+\rangle$ with $p = 1$ |
| $\|-\rangle$ | $\|0\rangle$ with $p = 1/2$, and $\|1\rangle$ with $p = 1/2$ | $\|-\rangle$ with $p = 1$ |

**Table 4.1. Probability of measuring a quantum state:**
**The table shows the probability that Bob has of measuring the states encoded by Alice using the bases $B_{HV} = \{|0\rangle, |1\rangle\}$, or $B_X = \{|+\rangle, |-\rangle\}$.**

After Bob measures and stores the entire string of states, he communicates with Alice using a classical channel and tells her which bases he used to measure each of the states. They compare the results and select the matching states to generate the encryption key. As specified in the following sequence:

- Bob communicates with Alice and sends the string of bases he used to measure the states. He uses a public, insecure communication channel for this process.
- Over the same channel, Alice indicates which measurements are correct.
- Alice and Bob compare the results and delete from their strings the bits where different bases were used.
- Alice sends Bob a list of positions along with their values to estimate the error rate.
- If the error rate is below 25%, the key exchange proceeds. If it is higher, communication is aborted, as there is certainty that an intruder is present in the communication channel.
- In the algorithm, after Alice generates her initial random binary string, she chooses a random rotation string to send the states to Bob via a quantum channel. In this case, the channel is simulated, but if the protocol were to be carried out experimentally, the quantum channel would be an optical fiber or free space.
- Similarly, when the states arrive at Bob's end, he also randomly chooses a rotation string, which allows him to measure each state. From this process, Bob generates two strings: one with the rotation of the bases, and the other with the state measurements.
- Afterwards, Alice and Bob communicate over a classical channel and share their rotation strings. From this, Alice informs Bob which states are correct to generate the cryptographic key, as shown in the next part of the algorithm.

**Step 4:** Executing the circuit

We will keep the bits that match so when alice_basis[i] == bob_basis[i]

```
# Full circuit
full_circuit = alice_circuit.compose(bob_circuit)

# Simulation
backend = Aer.get_backend('aer_simulator')
job = backend.run(full_circuit, shots=1, memory=True)
result = job.result()
measurements = result.get_memory()[0][::-1]  # reverse to match order
bob_results = [int(b) for b in measurements]

# Print results
print("Alice bits:  ", alice_bits)
print("Alice bases: ", alice_bases)
print("Bob bases:   ", bob_bases)
print("Bob results: ", bob_results)

# Shared key (where bases match)
shared_key = [alice_bits[i] for i in range(n) if alice_bases[i] == bob_bases[i]]
print("Shared key:  ", shared_key)
```

```
Alice bits:   [1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0]
Alice bases:  [0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0]
Bob bases:    [1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0]
Bob results:  [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0]
Shared key:   [1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0]
```

Entrée [2]:

```
import pickle



# Save the key to a file
with open("bb84_without_eve_key.pkl", "wb") as key_file:
    pickle.dump(shared_key, key_file)
```

## Alice vs Bob Bases Comparison (Matching Positions)

```
import matplotlib.pyplot as plt

basis_match = [1 if alice_bases[i] == bob_bases[i] else 0 for i in range(n)]
plt.bar(range(n), basis_match, color='blue')
plt.ylim(0, 1.5)
plt.xlabel('Qubit Index')
plt.ylabel('Bases Match (1=True)')
plt.title('Matching Bases between Alice and Bob')
plt.show()
```

Python

In the execution of the quantum circuit that Bob uses to measure the states sent by Alice, errors appear that must be considered in the transmission of information. This histogram shows the circuit output using a no-noisy model.

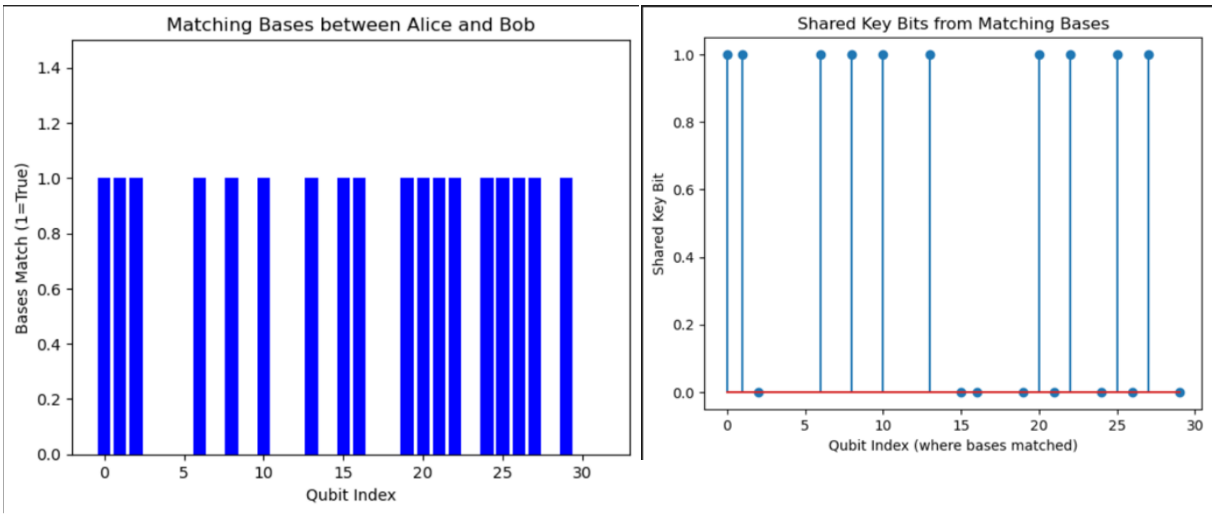



**Figure 4.2. Simulation of the output string from Bob's quantum circuit: The histogram shows the output bit string from the quantum circuit that Bob uses to measure the states. The figure was generated using Qiskit.**

```
print("\nBob's Measurement Histogram:")
plot_histogram(bob_counts, title="Bob's Measurement Outcomes (No Eavesdropper)")
```

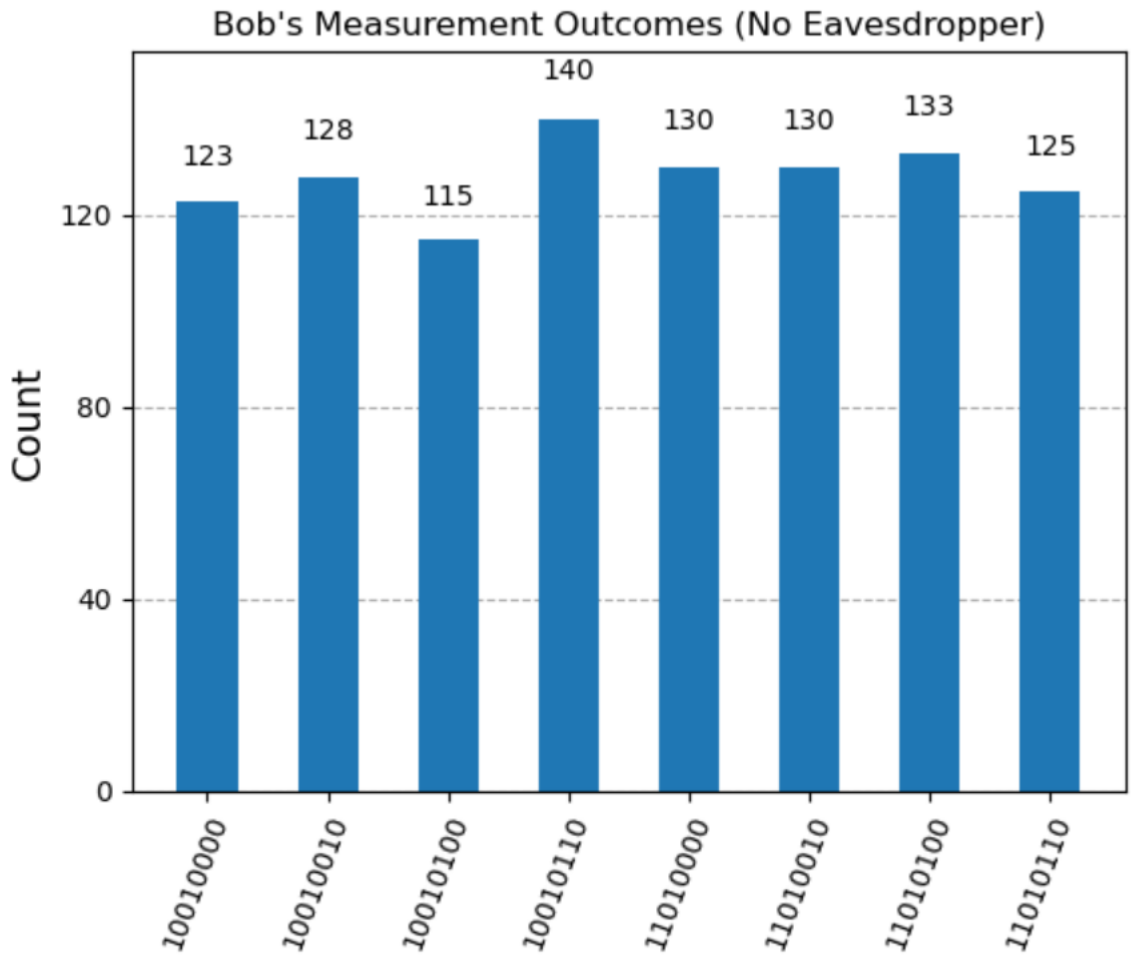



**Figure 4.3. Bob's Measurement Histogram**

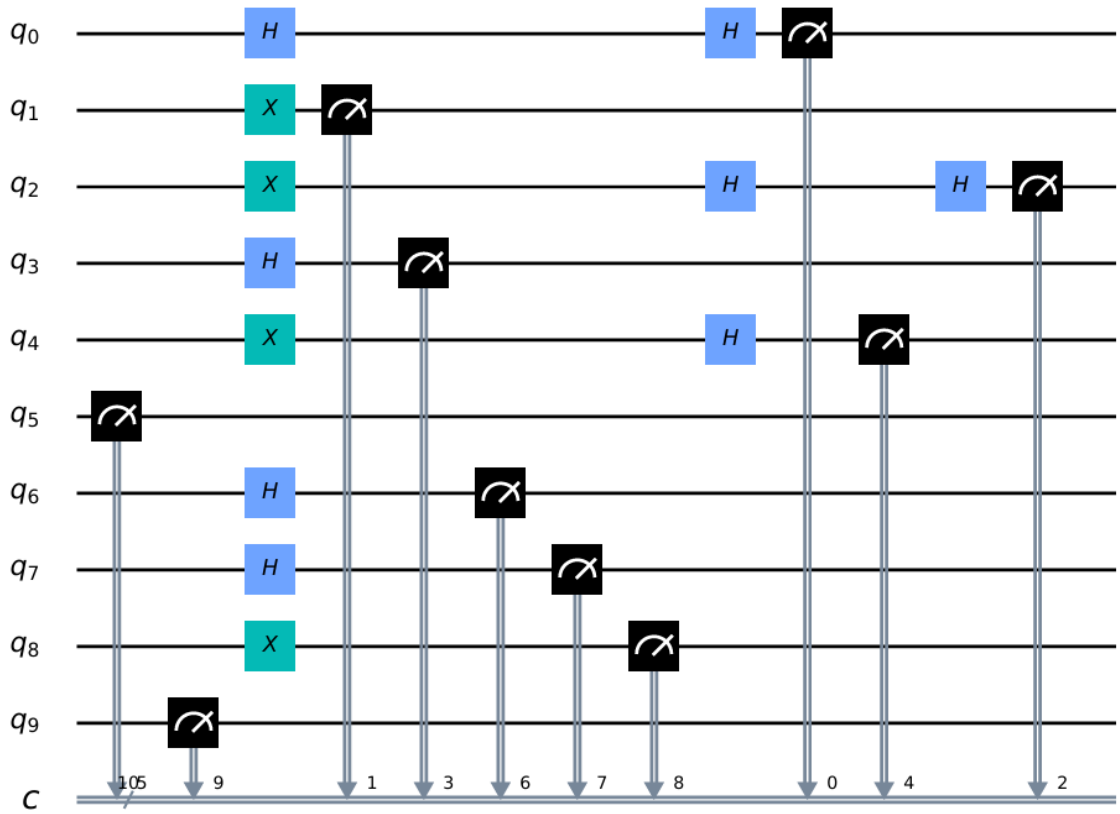



**Figure 4.4. Quantum circuit for measuring the states between Alice and Bob: The quantum circuit is initialized in the state |0⟩ so that Bob can begin measuring the quantum states transmitted by Alice. The figure was generated using Qiskit.**

After Alice and Bob select the positions that match in their binary rotation strings, Alice selects the bits from her initial binary string that will form the quantum key. Similarly, Bob selects the bits from his binary measurement string that will form his version of the quantum key. In the protocol, neither Alice's initial binary string nor Bob's measurement string is shared—they remain secret. Figure 4.5 shows the quantum key reconciliation process.

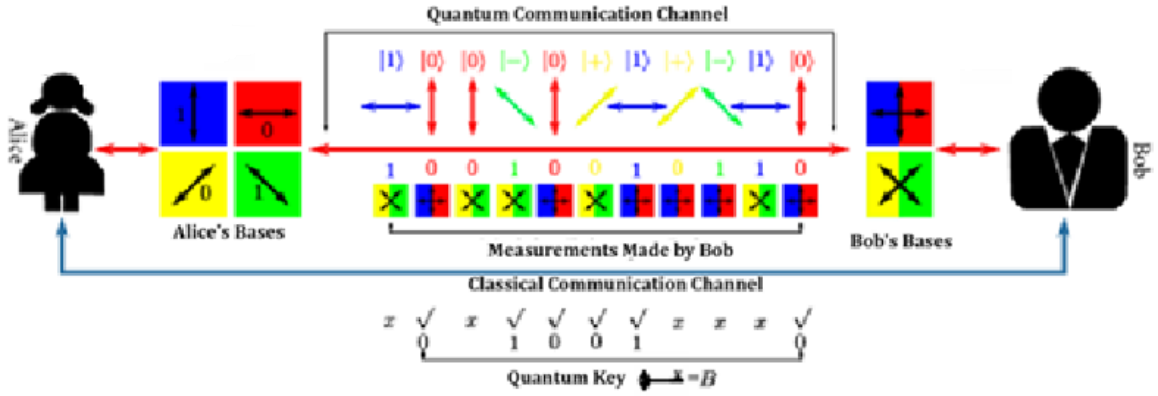



**Figure 4.5. Quantum key reconciliation in the BB84 protocol: The figure outlines the process of transmission, measurement, and comparison of quantum states to generate the cryptographic key.**

```
PROBABILITY_THRESHOLD = 0.25

if error_rate > PROBABILITY_THRESHOLD:
    print(f"Probability found {probability_found:.2%} exceeds threshold. Key discarded.")
else:
    shared_key = alice_key
    print("Key accepted.")
    print("Shared key (Alice & Bob):", shared_key)
```

```
Alice bits:    [1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0]
Alice bases:   [0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0]
Eve bases:     [1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1]
Eve results:   [0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1]
Bob bases:     [1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0]
Bob results:   [0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1]
Matching pos:  [1, 3, 5, 6, 8, 11, 12, 13, 16, 17, 23, 25, 28, 29, 31]
Alice key:     [1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0]
Bob key:       [1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1]
Errors:        [1, 0, 0, 0]
Probability found:    26.67%
Probability found 26.67% exceeds threshold. Key discarded.
```

## 4.2. Our Encryption and Decryption Method

- ✓ Our encryption method begins by dividing the plaintext into blocks, which are then encoded in ASCII. Each block is then converted into a unique DNA sequence using a DNA base encoding method. Next, we apply a technique called “DNA Shift”. Briefly, it is a shifting operation designed to enhance the encryption strength.
- ✓ To generate encryption keys, we use the BB84 protocol.
- ✓ For decryption, the process is simply the reverse of encryption.

### 4.2.1. Encryption Algorithm

- Extraction and Encoding of Blocks in ASCII

The segmentation phase is an important step in the data encryption process. In this step, each character from the plaintext is extracted and converted into its 8-bit binary ASCII equivalent. If the plaintext is not divisible by 16, padding characters are added to complete the block.

- DNA-Based Encoding

The binary-to-DNA conversion phase is a technique used to store digital data in the form of DNA sequences.

This conversion is based on mapping the four nucleotides of DNA (A, C, G, T) to the four binary combinations (00, 01, 10, 11). Each 2-bit binary block is therefore translated into a DNA nucleotide:

$$00 \rightarrow A \ , \ 01 \rightarrow C \ , \ 10 \rightarrow G \ , \ 11 \rightarrow T$$

- Shift-DNA (Shift-ADN)

In this Shift-DNA phase, we apply a shifting operation to each block in order to further strengthen the security of the encryption. This operation is inspired by the ShiftRows operation from AES, which is used in cryptography to mix the bits within a data block.

To apply this shift operation:

1. We divide each 64-base block into 8 groups of 8 bases.
2. For each group of 8 bases, we shift the bases to the left by a specific number of positions. For example: group 1 is shifted by one position, group 2 by two positions, group 3 by three positions, and so on. This shifting is different for each group to add complexity to the operation.
3. After shifting each group of 8 bases, we reorganize the groups so that the shifted bases are grouped together.
4. We then reassemble the 8 groups of 8 bases into a new 64-base block.
5. These steps are repeated for each block in the DNA sequence.

The shifting operation applied to each 8-base block is a simple yet effective method to enhance the security of DNA sequence encryption and add additional complexity to the encrypted sequence.

- Transcription (DNA to RNA)

The resulting RNA strand is a copy of the DNA strand, but with uracil ($U$) replacing thymine ($T$) in the nucleotide sequence. This RNA strand is called messenger RNA (mRNA) because it carries the genetic information from the DNA.

| Base ADN | Base ADN complémentaire | Base ARN complémentaire |
|---|---|---|
| A | T | U |
| T | A | A |
| C | G | G |
| G | C | C |

**Table. 4.2: DNA Base to RNA Base**

- Biological XOR

In DNA-based cryptography, biological XOR is commonly used to encrypt DNA sequences. In this context, a DNA key is combined with the sequence to be encrypted using biological XOR, resulting in an encrypted sequence that is difficult to decrypt without the corresponding key. The process is similar to the one used in computer science to encrypt data using classical XOR, but here nucleotides are compared and combined rather than bits.

For encryption: Message to be encrypted XOR Key = Encrypted Message

For decryption: Encrypted Message XOR Key = Plaintext Message

When the XOR operation is performed a second time with the key, it cancels out the effect of the first XOR operation with the same key. This brings the encrypted sequence back to the original sequence, which is the message to be encrypted.

| XOR | A | C | G | U |
|---|---|---|---|---|
| A | A | C | G | U |
| C | C | A | U | G |
| G | G | U | A | C |
| U | U | G | C | A |

**Table. 4.3: Biological XOR**

- Translation (RNA to Amino Acids)

In this step of translating RNA into amino acids or proteins, the genetic code plays a crucial role. It is a table that associates each nucleotide triplet with a specific amino acid. This table, known as the "genetic code table," allows researchers to predict which RNA sequence will lead to the production of a specific protein.

**Example**

1. Extraction and Encoding of Blocks in ASCII

Let's assume we have the following plaintext: "it works", which contains a total of 8 characters.

| Character | ASCII Decimal | ASCII Binary |
|---|---|---|
| i | 105 | 01101001 |
| t | 116 | 01110100 |
| (space) | 32 | 00100000 |

| Character | ASCII Decimal | ASCII Binary |
|---|---|---|
| w | 119 | 01110111 |
| o | 111 | 01101111 |
| r | 114 | 01110010 |
| k | 107 | 01101011 |
| s | 115 | 01110011 |

Full binary string:

01101001 01110100 00100000 01110111 01101111 01110010 01101011 01110011

**2.** DNA Base Encoding

- Group into 2-bit chunks**:**

01 10 10 01 01 11 01 00 00 10 00 00 01 11 01 11

01 10 11 11 01 11 00 10 01 10 10 11 01 11 00 11

- DNA sequence:

C G G C C T T A A G A A C T T T

C G T T C T A G C G G T C T A T

- Final DNA-encoded string:

CGGCCTTAAGA ACTTTCGT TCTAGCGGTCTAT

**3.** DNA Shift

- **Step 1:** Divide each block into 8 groups of 8 bases.

Divide into 4 groups of 8:

Group 1: C G G C C T T A

Group 2: A G A A C T T T

Group 3: C G T T C T A G

Group 4: C G G T C T A T

Final Output (Group-wise Shifted DNA):

Group 1: G T T G G A A C

Group 2: G A G G T C C C

Group 3: A C A A A A T C

Group 4: C G G T C T A T

- Step 3: Combined Final DNA String:

GTTGGAACGAGGTTCCACA AAATCCGGTC TAT

4. Transcription (DNA to RNA)

We transcribe the DNA to RNA by replacing each base using the following rules:

- A → U
- T → A
- C → G
- G → C

Final RNA Sequence:

GUUGGAACGAGGUUCCACAAAAUCCGGUCUAU

5. Application of Biological XOR
6. For decryption: Encrypted Message XOR Key = Plaintext Message

Original Plaintext Message: “It works”

In this chapter, we proposed a symmetric encryption system based on DNA, following several steps, each relying on different techniques.

We explained each part in detail, providing the necessary information. Our key generation method, based BB84 protocol, was used to enhance the system’s security. Combined with a strategy that uses DNA encryption methods such as transcription, translation, and DNA shifting, we maximize the overall security.

### 4.2.3. Simulation of Quantum Key Generation inspired by DNA encoding

```python
# === Encryption Function ===

def encryption_fn(plaintext, key):
    padded_text = add_padding(plaintext)
    binary_ex = ''.join(format(byte, '08b') for byte in padded_text)
    dna_ex = Dna_transf(binary_ex)
    shifted_Dna = Dna_shifting(dna_ex)
    rna_ex = Rna_transf(shifted_Dna)
    binary_rna = ''.join(Rna_to_bin(rna_ex))

    block_size = len(key)
    blocks = [binary_rna[i:i+block_size] for i in range(0, len(binary_rna), block_size)]
    if len(blocks[-1]) < block_size:
        blocks[-1] = blocks[-1].ljust(block_size, '0')

    ciphertext_blocks = [xor_binary_strings(block, key) for block in blocks]
    return ''.join(ciphertext_blocks)

# === Decryption Function ===

def decryption_fn(ciphertext, key):
    block_size = len(key)
    blocks = [ciphertext[i:i+block_size] for i in range(0, len(ciphertext), block_size)]
    decrypted_blocks = [xor_binary_strings(block, key) for block in blocks]
    binary_rna = ''.join(decrypted_blocks)

    rna_seq = [binary_rna[i:i+2] for i in range(0, len(binary_rna), 2)]
    rna_bases = [{"00": "U", "10": "G", "01": "C", "11": "A"}[pair] for pair in rna_seq]
    dna_bases = [{"U": "A", "A": "T", "G": "C", "C": "G"}[base] for base in rna_bases]
    dna_str = ''.join(dna_bases)

    shifting_dic = {"0": 1, "1": 2, "2": 3, "3": 4}
    mid = len(dna_str) // 2
    block1 = dna_str[:mid]
    block2 = dna_str[mid:]

    def reverse_shift_blocks(block):
        grouped = [block[i:i+8] for i in range(0, len(block), 8)]
        return [right_shift_string(group, shifting_dic.get(str(i % 4), 1)) for i, group in enumerate(grouped)]


    key = load_key_from_file()
    plaintext = "it works"

    ciphertext = encryption_fn(plaintext, key)
    decrypted = decryption_fn(ciphertext, key)

    print("Original:   ", plaintext)
    print("Key:        ", key)
    print("Ciphertext: ", ciphertext)
    print("Decrypted:  ", decrypted)
```

```
Original:    it works
Key:         [1, 0, 1, 1]
Ciphertext:  000111100110101010111100110010010110011100100000110010001101000010011011100110110011101110111001101110011011001110110011
Decrypted:   it works
```

## Conclusion

The successful simulation validates the feasibility of combining quantum key distribution with DNA encoding to form a multi-layered security model. The hybrid approach offers enhanced resistance to classical and quantum attacks, with strong key integrity and biological data obfuscation, paving the way for next-generation secure systems.

## Conclusion and Future Directions

In this thesis, we explored the convergence of two advanced cryptographic paradigms—quantum cryptography and DNA-based encryption—to address the growing demand for secure communication in the face of evolving computational threats. With the advent of quantum computing, traditional cryptographic systems based on mathematical hardness assumptions are becoming increasingly vulnerable. Quantum Key Distribution (QKD), particularly the BB84 protocol, offers a promising alternative by leveraging the fundamental principles of quantum mechanics to achieve provable security in key exchange.

To complement this quantum approach, we introduced DNA cryptography as a novel layer of post-processing that enhances data confidentiality and obfuscation. DNA encoding, with its enormous data density and biological complexity, provides a physical and computational shield that is resistant to both classical and quantum adversaries.

The practical contribution of this thesis included the simulation of the BB84 protocol under two scenarios -both in the presence and absence of an eavesdropper- followed by the application of a DNA encoding scheme for secure message encryption and decryption using the key generated by QKD. The results demonstrated the feasibility of integrating quantum and DNA cryptographic methods to construct a multi-layered security model.

This hybrid framework, although still at a conceptual stage, lays the foundation for future secure communication systems that combine the power of quantum mechanics and molecular biology. Challenges such as implementation complexity, scalability, and cost still need improvement. However, the proposed model is a step toward strong cryptographic solutions in a post-quantum world.

Future work may focus on improving DNA encoding methods and integrating error correction for quantum channels. As cryptography moves beyond mathematics into the fields of physics and biology, these interdisciplinary approaches will become increasingly important for protecting information in the digital age.