PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA

FACULTY : Mathematics And Computer Science

DEPARTEMENT : Computer Science

N° :……..

DOMAIN: Mathematics and Computer Science

BRANCH: Computer Science

OPTION : Networks and Technologies of Information and Communication

A Dissertation in Fulfillment

For the Requirements of the Degree of Master

By: ABDERRAHMEN GHADBANE

SUBJECT

Benchmarking Study of Post-Quantum Schemes in Smartphones

Defended publicly on: 25/06/2018

Board of Examiners:

Dr. ABID. GRAINI	University of M'sila	Chairman
Dr. NOUREDDINE CHIKOUCHE	University of M'sila	Supervisor
Dr. MOHAMED KAMEL	University of M'sila	Examiner

Academic year: 2017 /2018

ACKNOWLEDGEMENT

I would first like to thank my dissertation advisor Dr. NOUREDDINE CHIKOUCHE of the Mathematics and Computer Science faculty at MOHAMED BOUDIAF University - M'sila. The door to Prof. CHIKOUCHE office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge the reader of this dissertation, and I am gratefully indebted to his/her for his/her very valuable comments on this dissertation.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you.

Contents

List of figures

List of tables

GENERAL INTRODUCTION

GENERAL INTRODUCTION

Today, the security of current public key cryptography relies on the hardness of two mathematical problems: discrete logarithm problem and integer factorization problem.
These two mathematical problems can be solved in a short amount of time on a large quantum computer using Peter Shor's algorithm that he proposed in 1994 [1], which means that many of the currently used cryptographic solutions may become useless in the near future.
Research is currently focused on cryptographic solutions that can resist Shor's algorithm, this area of research is called post-quantum cryptography. Cryptography provides an important feature that is secure communication, but applying it on devices that have limited resources like mobile phones can be challenging and the wrong choice of cryptographic solutions can lead to a bad user experience or in low system security level.

Objective of this dissertation:

The goal of the work entitled "**Benchmarking Study of Post-Quantum Schemes in Smartphones"** is to present different post-quantum cryptosystems and signature schemes, such as: NTRU, variants of McEliece, XMSS, Rainbow, etc..., then benchmark them in various Android devices using Java as a programming language for our development environment. We analyze the obtained results taking into consideration the time of execution when performing different cryptographic operations, and occupied resources like memory, and energy consumption to determine which cryptographic schemes are suitable for the mobile environment.

Organization of this dissertation:

Our work is divided into three chapters:

- In the first chapter, we present the fundamentals of cryptography and the threat of quantum computers on classical cryptography, then we present the different categories of post quantum cryptography.
- In the second chapter, we present different post-quantum cryptosystems and signature schemes

- In the third chapter, we benchmark the different post-quantum schemes that we mentioned in chapter two on Android smartphones, then we discuss the obtained results.

CHAPTER 01
PRELIMINARIES

CHAPTER 1

PRELIMINARIES

1.1 Introduction

Public key cryptography is widely used today by businesses, governments and personals to ensure the confidentially and safety of data. Current cryptographic schemes are based on the difficulty of solving mathematical problems, like integer factorization and discrete logarithm. In 1993 Peter Shor published a new algorithm [1] that can be used in a quantum computer to solve the factorization and the discrete logarithm problems in a fair amount of time. This will make most current cryptosystems useless and compromise the safety and integrity of exchanged data.

While secret-key cryptosystems like (AES) [20] aren't based on integer factorization or discrete logarithm problems, which make them resist Shor's quantum attack, it should be noted that in 1996 Lov Grover proposed a quantum algorithm [2] that cuts the time needed to brute force secret-key cryptosystems on a traditional computer in half, this means larger keys are needed to compensate for quantum attacks.

Researchers from the National institute of Standards and Technology (NIST) published a report [3] showing the effects that can be caused by quantum attacks against popular public and secret key cryptographic schemes, **table 1.1** shows their findings.

Cryptographic Algorithm	**Type**	**Purpose**	**Impact from large-scale quantum computer**
AES	Symmetric key	Encryption	Larger key sizes needed
SHA-2, SHA-3		Hash functions	Larger output needed
RSA	Public key	Signatures, key establishment	No longer secure
ECDSA, ECDH (Elliptic Curve Cryptography)	Public key	Signatures, key exchange	No longer secure
DSA (Finite Field Cryptography)	Public key	Signatures, key exchange	No longer secure

Table 1.1 Impact of quantum computing on common cryptographic algorithms [3]

In response to this, the NSA (US National Security Agency) announced in 2016 [4] that it is planning to move to a new cipher suite (Suite B) that is quantum resistant in the near future.
Research is focused today on two main fields: quantum cryptography and post-quantum cryptography. Quantum cryptography uses quantum mechanics for communication and computation to protect transmitted messages, Quantum Key Distribution (QKD) is a well-known example of quantum cryptography. The other field that we are going to study in this dissertation is post-quantum cryptography, which is about the development of cryptographic schemes that can resist quantum and classical attacks and fit in the current IT infrastructure. There are five popular families known to build post-quantum asymmetric cryptography: hash-based cryptography, code-based cryptography, lattice-based cryptography, multivariate-quadratic-equations cryptography and supersingular elliptic curve isogeny cryptography.

1.2 Cryptology

Cryptology has a lot of applications like secure website access, E-mail encryption and smart cards that are used in banking applications or code deciphering, such as the famous attack against the German Enigma encryption machine during the second world war (**Figure1.1**) [51].

Figure1.1 The German Enigma encryption machine [55]

Cryptology can be divided into two categories: cryptography and cryptanalysis.

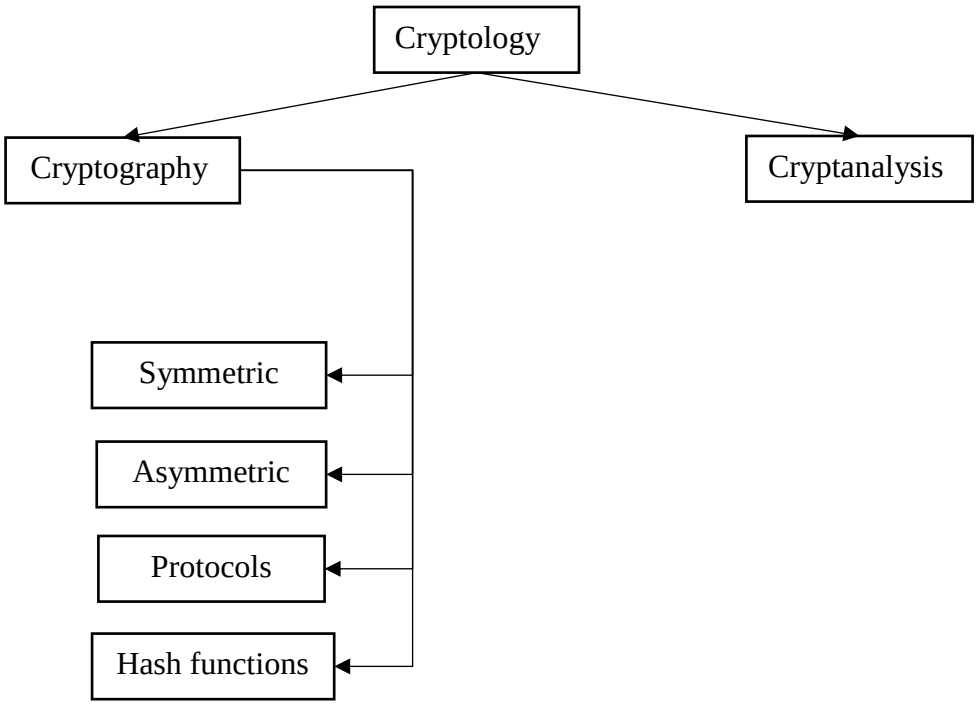

Fig. 1.2 Overview of the fields of cryptology

1.3 Cryptanalysis

Cryptanalysis is the science of breaking cryptosystems by researchers, and finding some weakness or insecurity in them. which is important to make sure that modern cryptosystems are secure and not obsolete.

1.4 Cryptography

Cryptography is the practice and study of secret writing with the goal of hiding the meaning of a message using secure communication in the presence of third parties by ensuring the properties on information security:

- **Confidentiality** ensures that information is not made available or disclosed to unauthorized third parties.
- **Data integrity** ensures the accuracy and completeness of data over its entire lifecycle, which means that information cannot be modified in an undetected or unauthorized manner.

- **Authentication** that allows verifying a claim of identity, there are three types of information that can be used for authentication:
    - Something you know: such as a password.
    - Something you have: a driver license for example.
    - Something you are: such as fingerprints.
- **Non-repudiation** ensures that the receiver cannot deny having received a message, nor can the sender deny having sent the message [51].

1.4.1 Symmetric encryption

Symmetric key encryption uses the same key to encrypt and decrypt ciphertext, this method has the advantage of speed because there is less delay, but it also introduces a drawback, that both the sender and the recipient needs to agree on the same key and any exchange must retain the privacy of the key. One example is the Advanced Encryption Standard (AES) [20].

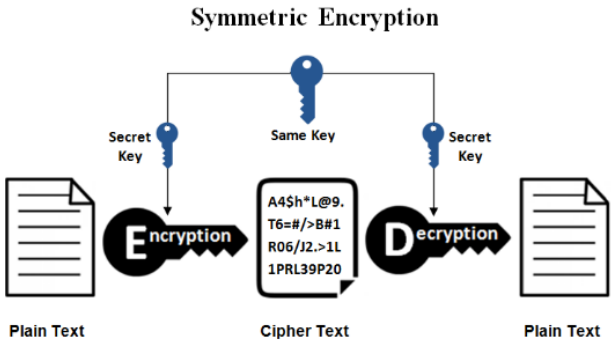

Figure1.3 Symmetric encryption [54]

1.4.2 Asymmetric encryption

Asymmetric encryption (also known as public key encryption) uses a key pair, a public key is used for encryption and it's freely distributed to anyone who wants to send a message, and private key is used for decryption and must be kept a secret. Asymmetric encryption also introduces the notion of non-repudiation that prevents the sender from claiming that the message was never sent and the message from being altered which guarantees data integrity, **Figure1.4** shows the categories of Asymmetric cryptography:

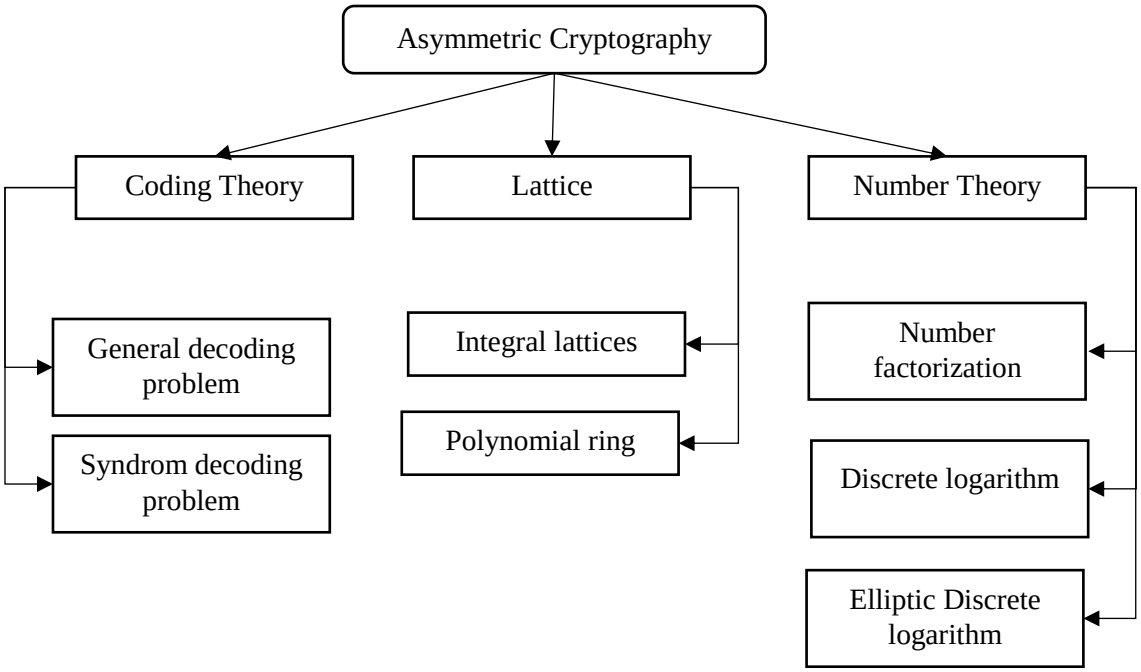

Figure1.4 categories of Asymmetric cryptography

Public key cryptosystems have three functionalities:

- **Key Generator** that generates a public key and secret key with pre-determined security parameters.
- **Encryption** that takes a public key and a plain text and turn them into a cipher text.
- **Decryption** that takes a private key and a cipher text and outputs a plaintext. A notable example is the Rivest-Shamir-Adelman scheme (RSA) [19].

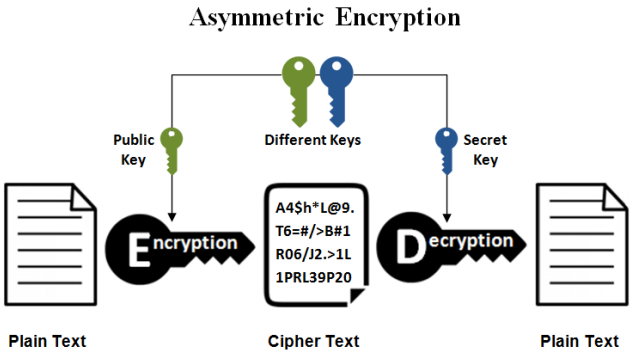

Figure 1.5 Asymmetric encryption [54]

1.4.2.1 Digital signature

Digital signature schemes usually consist of triplet (key generation, signing, verification) where:

- **Key generation:** a public key and a corresponding secret key are randomly generated with a pre-determined security parameter as input.
- **Signing:** a unique signature is generated using the message and the secret key as input.
- **Verification:** takes the signature and the public key and the message as input and outputs a positive or a negative response to the authenticity of the message.

Digital signature schemes are secure as long as it is computationally infeasible to generate a valid signature without knowing the corresponding secret key. Digital Signature Algorithm (DSA) [21] is an example of a digital signature scheme developed by NIST.

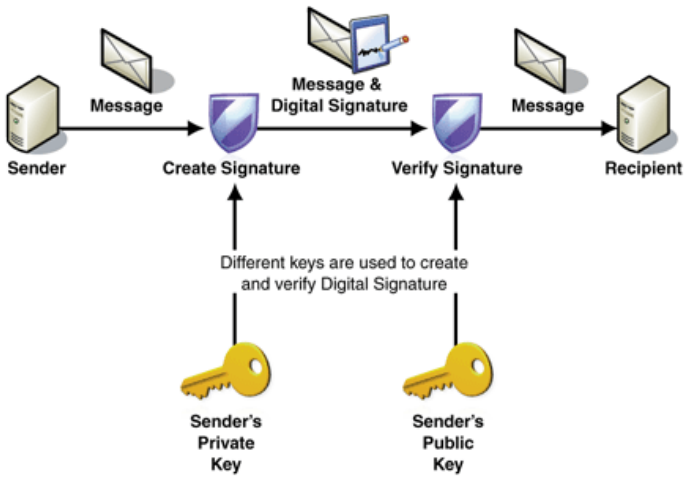

Figure1.6 Digital signature generation [54]

1.4.3 Cryptographic Protocols

cryptographic protocols are a set of exchanged messages between the parties of a network. Symmetric and asymmetric algorithms can be used as building blocks to achieve a secure Internet communication. An example of a cryptographic protocol is the Transport Layer Security (TLS) scheme, which is a key exchange protocol used on Web browsers. The different classes of protocols are:

- **Authentication protocols** ensure the property of authenticity.
- **Key exchange protocols** ensure the property of confidentiality.
- **Signature protocol**s ensure the property of non-repudiation.

1.4.4 Hash functions

Hash functions are widely used in signature schemes and protocols. They compute a short fixed-length message called a digest, which can be used as a digital fingerprint to ensure data integrity. The main difference between hash functions and other cryptographic schemes is that hash functions do not have a key.

1.4.4.1 Strong collision resistance

A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$, has the property of collision resistance if it is computationally infeasible to find two inputs a and b that hash to the same output such that $H(a) = H(b), a \neq b$.

1.4.4.2 Weak collision-resistance

A hash function has the property of weak collision-resistance if it is difficult to find $x' \neq x$ such that $H(x') = H(x)$.

1.4.4.3 One-way function

A hash function is one-way if it is easy to compute $H(x)$ from x, but it is difficult to find x from y such as $y = H(x)$.

1.5 Mathematical problems used in classical cryptography

These mathematical problems are used by classical cryptographic schemes and they can be solved by quantum computers in the near future.

1.5.1 Factorization problem

In number theory, prime factorization is the decomposition of a composite number into a product of smaller prime numbers. Let $x \in \mathbb{N}$ and $x = pq : p, q \in \mathbb{N}$ .if p and q are sufficiently large this problem can be time and resource consuming on traditional computers, but large-scale quantum computers can solve this problem in a matter of minutes which can compromise the security of algorithms that use this problem, like the RSA scheme [19].

1.5.2 Discrete logarithm problem

The discrete logarithm problem is based on the fact that calculating exponentiation is easy but finding the logarithm is hard to calculate, and it’s defined as follows:

Given two numbers a, b such as $b = a^x$, find x in polynomial time. If a, b are quite large, the problem will be computationally hard to solve on classical computers.

1.5.3 Elliptic curved discrete logarithm problem

Given an elliptic curve E. We consider two primitive elements P and T. The ECDLP problem is solved by finding the integer d, where $1 \leq d \leq E$, such that:

$$\underbrace{P + P + \cdots + P}_{\text{d times}} = dP = T. \quad (1)$$

1.6 Quantum computing

When large quantum computers become a reality in the future, the following quantum algorithms can be used to effectively tackle the factorization problem and the discrete logarithm problem.

1.6.1 Shor's algorithm

Peter Shor proposed an algorithm in 1994 [1], it's a quantum algorithm, which means that it's designed to work on a quantum computer, it uses Quantum Fourier Transforms to solve the integer factorization problem by translating it into finding the period of a function using superposition of quantum states [1], then it uses classical computation to translate it back to the original problem. Shor's algorithm can run in polynomial time, it can factor an integer N to its primes in $O((\log N)^2 (log\ log\ N)(log\ log\ log\ N))$ using fast multiplication [13], which is significantly faster than any classical factoring algorithm known today, Shor also says that his algorithm can be used to solve the discrete logarithm problem efficiently.

1.6.2 Grover's algorithm

The quantum algorithm entitled "A Fast Quantum Mechanical Algorithm for Database Search" was published by Lov Grover in 1996 [2], it provides a quadratic speedup when searching an unstructured database, with a worst-case complexity of $\mathcal{O}(N^{1/2})$ where N is the number of entities, compared to the classical computer approach that is $\mathcal{O}(N)$.

By achieving a quadratic speedup when searching for cryptographic keys using brute force, symmetric cryptosystems will be potentially threatened, and the security parameters (key size) should be doubled for a good countermeasure.

1.7 Post quantum cryptographic paradigms

Bernstein in his book (post quantum cryptography) [5] explained why cryptography will not be dead after the development of a large practical quantum computer in the near future. Certain aspects of current public key schemes that are based on integer factorization, discrete logarithm and elliptic-curve discrete logarithm problems will be threatened, but secret key schemes will only need to augment their security parameters. Post-quantum is a term given to cryptosystems that can resist both classical and quantum attacks, and can fit in the current IT infrastructure. The growing interest in post quantum is focused on five areas of research: code based cryptography, Lattice-based cryptography ,Hash-based signatures, Multivariate cryptography, Supersingular elliptic curve isogeny cryptography. Each one of these categories has its own advantages and weaknesses, and some of them have been around for 30 years while others are relatively new.

1.7.1 Lattice-based Cryptography

A lattice is a set of points in n-dimensional space with a periodic structure, more formally, given *n*-linearly independent vectors $b_1, \dots, b_n \in \mathbb{R}^n$, the lattice generated by them is the set of vectors:

$$\mathcal{L}(b_1, \dots, b_n) = \{\sum_{i=1}^{n} x_i b_i \ : x_i \in \mathbb{Z}\} \quad (2)$$

The vectors $b_1, \dots, b_n$ are known as a basis of the lattice.

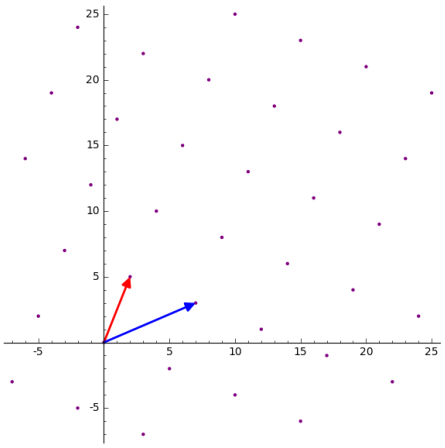

Figure1.7 A two-dimensional lattice and two possible bases b_1 = [2; 5]; b_2 = [7; 3] [6]

Lattice-based cryptographic constructions are based on the presumed hardness of lattice problems, the most basic of which is the shortest vector problem (SVP) and its variant the closest vector problem (CVP) and both of them are NP-complete [5].

1.7.1.1 Shortest vector problem (SVP)

The goal of this problem is to find a shortest non-zero vector s in a lattice $\mathcal{L}(B)$, with a given arbitrary basis B.

1.7.1.2 Closest vector problem (CVP)

In this problem when given a lattice $\mathcal{L}$, vector v in a vector space V and a metric M, the goal is to find a vector in the lattice $\mathcal{L}$ that is the closest to v as measured by the metric M.

1.7.2 Hash-based cryptography

Digital signatures provide integrity, authenticity, and non-repudiation of data, but some of them that relay on the difficulty of factoring large composite integers like RSA and DSA are not quantum resistant, while hash functions like SHA-2 and SHA-3 will have to use higher security parameters, the alternative is hash-based digital signature schemes like Lamport-Diffie One-Time Signature that depend on collision resistance, which mean that using the same private key to sign multiple documents will not yield a similar signature [6].

hash-based signature schemes use cryptographic hash functions [15]. And they rely on the security of the used hash function instead of the hardness of solving a mathematical problem. They also have the advantage of not depending on a specific hash function [16], generally it is possible to change the hash function if it grants an increased level of security.

1.7.3 Code-based cryptography

Code based cryptography schemes were the first to offer quantum resistance capability, based on coding theory and goppa codes, first introduced by McEliece in 1978 [7] they use error correcting codes to generate public keys from private matrices with intentionally added errors, they are quite fast when it comes to encoding and decoding due to low complexity but they suffer from huge key sizes, numerous attempts were made to reduce their size notably Niederreiter [8] and Quasi Cyclic Medium Density Parity Check Codes (QC-MDPC) cryptosystems [9]. Code-based cryptography is a good candidate for post-quantum encryption, with the McEliece cryptosystem as

the most promising one, which has been recently recommended by Post-Quantum Crypto Project of Europe because it "has been studied since 1978 and has withstood attacks very well" [10].

1.7.3.1 Coding theory

Claude Shannon published his paper: "A Mathematical Theory of Communication" in 1948 [14] where he explained the principals of information theory and coding theory. The main idea behind coding theory is to find a reliable and an efficient data transmission method by attaching extra information to the message which allows it to be received without any errors that can occur when transmitting through a noisy channel. A simple coding theory method to find the errors when sending a message is to send it three times and the receiver can apply a majority vote method which enables the recovery of the original message, It's an effective way but it's not the most efficient. The goal of coding theory is discovering optimized methods that allows the message to be sent error-free and reducing the redundancy at the same time.

1.7.3.2 Goppa Codes

The McEiece scheme and its variants use a type of code called goppa codes, with their length and dimension fixed they have some interesting properties, they can be generated by using an irreducible polynomial of a t degree, where t is the error correcting potentials of the goppa code. Goppa codes give the benefit of a quick polynomial time decoding algorithm and their generator matrices are roughly random, which make them elusive. Also, their number increments exponentially with the degree of polynomial t and the length of the code n [47].

1.5.3.3 Binary Syndrome Decoding problem

Given a parity check matrix $r \times n$ denoted H over the field F_2 . s is a target binary vector and t is an integer where $t \geq 0$, the goal is finding a binary x where $s = H \times T$. This problem is NP-complete. There are many NP-complete types of this problem that are defined over the field F_q, but they have little significance to code based schemes [47].

1.5.3.4 Goppa Code Distinguishing problem

Given a Goppa code C, a random number n and a random number k, the goal is to find a $(n - k) \times n$ binary matrix H, which is the parity check matrix of the goppa code C. that problem is also NP-complete. These two problems are the basis of the McEliece scheme security and the same goes for other code based schemes [47].

1.7.4 Multivariate-based cryptography

Multivariate public key cryptosystems use a set of multivariate polynomial equations that are usually based on the MQ Problem. They have properties to make them easy to solve, these properties are hidden by the public key using affine transformations, and the private key is used to unhide them to solve the polynomial equations. Solving these equations is NP-complete which make the multivariate schemes fast and require little computational resources, they also generate the shortest signatures, and that make them practical among post-quantum algorithms. Currently only signature schemes have been developed and they can be grouped into two categories. Single field schemes like Rainbow [11] that uses oil and vinegar polynomials, and the second one is big field schemes like HFEv [12].

1.5.4.1 MQ Problem

Let $\mathbb{F}_q$ denote the finite field of order q and $\mathbb{F}_q[x_1, \ldots, x_n]$ the polynomial ring over $\mathbb{F}_q$ with n variables. For any $f = (f_1(x1, \ldots, x_n), \ldots, f_m(x1, \ldots, x_n)) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$, the goal is to find a common zero $x_0 \in \mathbb{F}_q^n$ of the polynomials $f_1, \ldots, f_m$ and the degree of those polynomials is equal to 2 [22].

1.7.5 Supersingular Elliptic Curve Isogeny

This field of research is based on supersingular elliptic curve isogenies, and it can often be confused with the popular elliptic curve cryptography that rely on the hardness of elliptic curve discrete logarithm problem, which is broken by Shor's algorithm, instead this field uses supersingular elliptic curves and isogenies [17] that can resist quantum attacks. NIST stated that not enough analysis has been done to prove the claimed security [3]. Applying this concept will yield a fairly small keys and messages that are comparable to those we use today, thus making it an interesting field of research in the future.

1.8 Conclusion

In this chapter, we presented the fundamentals of cryptography and the threat of quantum computing on classical cryptography, and field of post-quantum cryptography. In the next chapter, we will present different post-quantum algorithms that we are going to benchmark their performance later.

CHAPTER 02
POST-QUANTUM ALGORITHMS

CHAPTER 2

POST-QUANTUM ALGORITHMS

In this chapter, we explain the algorithms that we are going to use in our benchmark later in chapter three. The algorithms will be divided into two categories: post-quantum cryptosystems and post-quantum signature schemes.

2.1 Post-quantum cryptosystems

In this category, we will present the following algorithms: McEliece, The Kobara/Imai conversion of McEliece, Niederrreiter and NTRUEncrypt.

2.1.1 The McEliece cryptosystem

The McEliece cryptosystem was presented by McEliece in 1978 [23], And after 30 years it is still unbroken which makes it a great candidate for post quantum cryptography. The cryptosystem consists of three algorithms: A key generation algorithm that produces a pair of public and private keys, an encryption algorithm and a decryption algorithm. The primary advantage of the McEliece scheme is the proficiency of both operations: encryption and decryption that are faster than the RSA scheme [19] that relies on exponentiation because the McEliece scheme uses matrix multiplication. It should be noted that all code based schemes and especially the McEliece scheme generate very large keys to store the generator matrix, and they can get bigger as the parameters of security level increase, for example a security level of a 128 bits generates a 14762393 bits public key compared to an RSA key which is 3072 bits and has the same security level. Algorithm 1 shows the procedure of key generation:

Algorithm 1: Key Generation Algorithm

Input: Security Parameters m, t

Output: K_{pub}, K_{sec}

$n \leftarrow 2^m, k \leftarrow n - m \cdot t$

$C \leftarrow$ random binary (n, k)-linear code C capable of correcting t errors

$G \leftarrow k \times n$ generator matrix for the code C

$S \leftarrow$ random $k \times k$ binary non-singular matrix

$P \leftarrow$ random $n \times n$ permutation matrix

$\hat{G} \leftarrow k \times n$ matrix $S \times G \times P$

return Public Key $(\hat{G}, t)$; Private Key (S, G, P).

Algorithm 2 shows the procedure of encryption:

Algorithm 2 Encryption Algorithm

Input: $M, K_{pub} = (\hat{G}, t)$

Output: Ciphertext c

Encode the message M as a binary string of length k

$c' \leftarrow M \cdot \hat{G}$

Generate a random n-bit error vector z containing at most t ones

$c \leftarrow c' + z$

return c

Algorithm 3 shows the procedure of decryption:

Algorithm 3 Decryption Algorithm

Input: $c, K_{sec} \leftarrow (P-1, G, S-1)$

Output: Plaintext M

$\hat{c} \leftarrow c \cdot P-1$

Use a decoding algorithm for the code C to decode $\hat{c}$ to $\widehat{M} = M \cdot S$

$M \leftarrow \widehat{M} \cdot S-1$

return M

2.1.2 The Kobara/Imai conversion of the McEliece cryptosystem

The McEliece scheme can be vulnerable to some attacks that can exploit the specifics of the encryption process, instead of the underlying decoding problem. To counter against these attacks, McEliece scheme should be used with a CCA2 conversion (general form of padding), which transforms the clear message into an arbitrary string of bits then it's encrypted with the classical McEliece scheme. Kobara and Imai [29] bring up two generic schemes that can be used with the McEliece scheme to give a CCA2-secure cryptosystem: Fujisaki-Okamoto scheme [30], and Poincheval's scheme [31]. Sadly, these two generic conversions introduce a high data redundancy. Hence, Kobara and Imai [29] propose two unique McEliece conversions that require an added

$(n-k+s)$ bits (where s is relative to the desired level of security). In addition, their third conversion γ (Algorithm below) uses the entropy in the error vector, to further decrease the overhead of data. Algorithm 4 shows the procedure of conversion:

Algorithm 4 Kobara-Imai-γ conversion applied to McEliece cryptosystem

- **Encryption Input:** Binary message M, Constant $Const$

  Output: Ciphertext c

  $\mathrm{r} \leftarrow \mathrm{Rand}$

  $\bar{\mathrm{M}} \leftarrow \mathrm{Prep(M)}$

  $\mathrm{y1} \leftarrow \mathrm{Gen(r)} \oplus (\bar{\mathrm{M}}||\mathrm{Const})$

  $\mathrm{y2} \leftarrow \mathrm{r} \oplus \mathrm{Hash(y1)}$

  $(\mathrm{y5}||\mathrm{y4}||\mathrm{y3} \leftarrow (\mathrm{y2}||\mathrm{y1})$

  $\mathrm{z} \leftarrow \mathrm{Conv(y4)}$

  $\mathrm{c} \leftarrow \mathrm{y5}||\varepsilon^{\mathrm{McEliece}}(\mathrm{y3}, \mathrm{z})$

  return c

- **Decryption Input:** Ciphertext c

  Output: Binary message M

  $\mathrm{y5} \leftarrow \mathcal{M}\mathrm{sb}_{\mathrm{Len(c)-n}}(\mathrm{c})$

  $\mathrm{y3} \leftarrow \mathrm{D}^{\mathrm{McEliece}}(\mathrm{Lsb}_{\mathrm{n}}(\mathrm{c}))$

  $\mathrm{z} \leftarrow \mathrm{y3}\acute{\mathrm{G}} \oplus \mathrm{Lsbn(c)}$

  $\bar{\mathrm{z}} \leftarrow \mathrm{Conv}^{-1}(\mathrm{z})$

  $\mathrm{y4} \leftarrow \mathrm{Lsb}_{\mathrm{log2\ C(n,t)}}(\bar{\mathrm{z}})$

  $(\mathrm{y2}||\mathrm{y1}) \leftarrow (\mathrm{y5}||\mathrm{y4}||\mathrm{y3})$

  $\mathrm{r} \leftarrow \mathrm{y2} \oplus \mathrm{Hash(y1)}$

  $(\bar{\mathrm{M}}||\mathrm{Const}') \leftarrow \mathrm{y1} \oplus \mathrm{Gen(r)}$

  If $\mathrm{Const}' = \mathrm{Const}$

  return $\mathrm{Prep}^{-1}(\bar{\mathrm{M}})$

  Otherwise reject c

2.1.3 Niederrreiter cryptosystem

in 1986 [24] Niederrreiter proposed a cryptosystem that is a variation of the McEliece cryptosystem. McEliece uses generator matrix, so he proposed some changes to decrease the key size by using the parity check matrix instead, which is good since large keys are the main disadvantage of code-based cryptography. But the main advantage of the Niederrreiter scheme is that it can be used to create digital signatures as proposed by Courtois, Finiasz and Sendrier [52]. Algorithm 5 shows the procedure of key generation:

Algorithm 5: Niederreiter Key Generation Algorithm

Select parameters n, k and t to form the length, dimension and error correcting capability of the Goppa code to be used as the key.

Generate a random parity check matrix H of dimension $n - k \times n$, capable of generating an $(n, k, 2n + t)$ linear binary code.

Select a random $n \times n$ permutation matrix P over $\mathbb{F}_2$.

Select a random invertible $n - k \times n - k$ matrix S over $\mathbb{F}_2$.

Compute $H_{pub} = SHP$.

Algorithm 6 shows the procedure of encryption:

Algorithm 6: Niederreiter Encryption Algorithm

Input: message M, public key (H_{pub}, t)

Output: cipher text c

1. Calculate $c = HM^T$.

Algorithm 7 shows the procedure of decryption:

Algorithm 7: Niederreiter Decryption Algorithm

Input: cipher text c, private key (S, G, P)

Output: original plain text M

Compute $S^{-1}c = HPM^T$

Since P is a permutation matrix, the hamming weight of PM^T is equal to the hamming weight of M.

Use the decoding algorithm to decode HPM^T, getting $y = PM^T$.

compute $M^T = P^{-1}y$

2.1.4 NTRU cryptosystem

The NTRU cryptosystem was published by Jeffrey Hoffstein and Jill Pipher and Joseph Silverman in 1998 [25] and it was standardized by IEEE in 2008 [26].

NTRU uses a public parameter N to specify the size of the used polynomials and a large modulus q and a smaller modulus p. The sender creates a pair of public and private keys by generating two polynomials f and g, where f is invertible modulo of both p and q.

Algorithm 8 shows the procedures of key generation, encryption and Decryption:

Algorithm 8: NTRU Algorithm

- **Key generation**

  Paramaters: N, q, p

  Public key: $h \leftarrow pf_q g \bmod q$ where $f_q \leftarrow f^{-1} \bmod q$

  Private key: consists of the polynomials f and $f_q \leftarrow f^{-1} \bmod p$

- **Encryption**

  $e \leftarrow rh + M \bmod q$ where r is a random polynomial

- **Decryption**

  $a \leftarrow fe \leftarrow frh + fM \leftarrow prg + fM \bmod q$

  $b \leftarrow a \bmod p$

  $M \leftarrow f_p b \bmod p$

2.2 Post-quantum signature schemes

In this category, we will present the following algorithms: Rainbow, NTRUSign, Niederreiter-CFS and XMSS.

2.2.1 Rainbow

This signature scheme was proposed in 2005 by Jintai Ding and Dieter Schmidt [28], it is a multivariate signature scheme based on the principle of Oil and Vinegar variables. it uses a multi-layer Oil-Vinegar system. The principle of Oil and Vinegar variables is one way to

easily create an invertible multivariate quadratic system [29]. The following algorithm shows the procedure of signing a document d:

- **Key Generation** The secret key consists of two affine and invertible maps $L_1 : K^d \rightarrow K^d$ and $L_2 : K^n \rightarrow K^n$ and the map $F = (f_{v_1+1}(x), \ldots, f_n(x))$. Here, $d = n - v_1$ is the components number of F. The public key composed of the field K and the map $P(x) = L1 \circ F \circ L2(x) : K^n \rightarrow K^d$.
- **Signature Generation** To sign a document d, we use a cryptographic hash function $h : K^* \rightarrow K^d$ to calculate the value $h = h(d) \in Kd$. Then we calculate recursively $x = L_1^{-1}(h)$, $y = F^{-1}(x)$ and $z = L_2^{-1}(y)$. The signature of the document is $z \in K^n$, while $F^{-1}(x)$ generates one pre-image of x.
- **Signature Verification** To verify the signature, we calculate $h' = P(z)$ and the value of the hash $h = h(d)$ of the document. If $h' = h$ is true, the signature is accepted, or it's rejected otherwise.

2.2.2 NTRUSign

The NTRU Signature Algorithm, also known as NTRUSign was presented in [48], it's based on the GGH signature scheme. It includes mapping a message to a random point in 2N-dimensional space, with N being a parameter, and solving the closest vector problem (CVP) in a lattice which is related to the NTRUEncrypt lattice problem [25]. Algorithm 9 shows the procedure of signing:

Algorithm 9: NTRUSign(pk, sk, μ)

Input: Public and private keys, and $d \in \{0,1\}^*$ the document to sign

Output: s the NTRU signature

repeat

$cpt \leftarrow cpt + 1$

$\mathcal{M} \leftarrow H(d, cpt) \in R_q$

$(x, y) \leftarrow (0, \mathcal{M}) \begin{pmatrix} G & -F \\ -G & f \end{pmatrix} / q$

$s \leftarrow -\{x\} * f - \{y\} * g$

until $\|(s, s * h - \mathcal{M})\| \leq N$

return (s, cpt)

Algorithm 10 shows the procedure of signature verification:

Algorithm 10: Verify($\mathrm{pk} = \mathrm{h}, \mathrm{s}, \mathrm{cpt}, \mathrm{d}$)

Input: Public key pk, the signature s, and the document d

Output: true if and only if s is a valid signature of d

$\mathcal{M} \leftarrow \mathrm{H}(\mathrm{d}, \mathrm{cpt})$

if $\|(\mathrm{s}, \mathrm{s} * \mathrm{h} - \mathcal{M})\| \leq \mathrm{N}$ then

return true;

else

return false;

2.2.3 Niederreiter CFS

A signature scheme based on the Niederreiter cryptosystem (a variant of the McEliece cryptosystem) was introduced by Courtois, Finiasz and Sendrier in [52]. The idea of the CFS algorithm is to frequently hash the message and randomize it by a counter of a bit-length r, until the output is a ciphertext that can be decrypted. To determine the error-vector, the signer uses his corresponding private key, and with the current value of the counter, the error vector will then serve as a signature. Algorithm 11 shows the procedures of key generation, signing and verification:

Algorithm 11 Niederreiter CFS digital signature

• **System parameters:** $\mathrm{m}, \mathrm{t} \in \mathbb{N}$.

• **Key Generation:** Generate a Niederreiter PKC key pair with a code drawn from the class of $[\mathrm{n} \leftarrow 2^{\mathrm{m}}, \mathrm{k} \leftarrow \mathrm{n} - \mathrm{mt}, 2\mathrm{t} + 1]$ binary irreducible Goppa codes.

• **Signing:**

Input: h a public hash function, $\varphi_{\mathrm{n,t}}, \mathrm{D}_{(\mathrm{S},\mathrm{D}_{\mathrm{G}},\mathrm{P})}, \mathrm{r} \in \mathrm{N}_{+}$ and the document d to be signed

Output: A CFS-signature s.

$\mathrm{z} \leftarrow \mathrm{h}(\mathrm{d})$

choose a r-bit Vector i at random

$\mathrm{s} \leftarrow \mathrm{h}(\mathrm{z}||\mathrm{i})$

while s is not decodable do

choose a r-bit Vector i at random

$\mathrm{s} \leftarrow \mathrm{h}(\mathrm{z}||\mathrm{i})$

$e \leftarrow D_{(S,D_G,P)}(s)$

$s \leftarrow (\varphi_{n,t}^{-1}(e)||i)$

• **Verification:**

Input: A signature $s \leftarrow (\varphi_{n,t}^{-1}(e)||i)$, the document d and H^{pub}

Output: accept or **reject**

$e \leftarrow \varphi_{n,t}(\varphi_{n,t}^{-1}(e))$

$s1 \leftarrow H^{pub}(e^{\top})$

$s2 \leftarrow h(h(d)||i)$

if $s1 = s2$ then

accept s

else

reject s

2.2.4 XMSS

The eXtended Merkle Signature Scheme was first introduced by Buchmann in 2011[53], it is a variant of the Merkle tree scheme. XMSS security is based on the properties of cryptographic hash functions and it is forward secure which means that regardless of whether the secret key is compromised, the signatures that were made before the compromise remain valid.

- **Key generation:** XMSS secret key consists of a cryptographic seed with a pseudorandom function (PRNG). This function is used to create the WOTS Key pair and the leaf index i corresponding to the next W-OTS keys to be used. the public key contains bitmasks and the root node value used in the transitional levels of the hash tree.
- **Signature generation:** for signing, a leaf index i is used to resolve the W-OTS key pair that should be used [53]. The signature $(\sigma, i, AUTH)$ consists of the W-OTS signature σ, index i, and the leaf node authentication path $AUTH$. The authentication path contains the hash values of H different nodes in the XMSS tree, one for each layer of the tree, then the leaf index i in the XMSS private key is updated after signing a message has been signed.
- **Signature verification:** for signature verification, first the verifier verifies the W-OTS signature σ using the corresponding public key that is generated by the verifier, then he checks the authentication path by traversing the tree using $AUTH$ to obtain p_H. If p_H is

equal to the root node value in the public key, the signature is accepted or rejected otherwise.

2.3 Conclusion

In this chapter, we presented different post-quantum algorithms and how they generate their keys and perform the encryption and decryption process, and generate digital signatures.
In the next chapter, we will benchmark their performance, then we discuss the obtained results.

CHAPTER 03
BENCHMARKING AND RESULTS

CHAPTER 3

BENCHMARKING AND RESULTS

In this chapter, we benchmark the performance of the different post-quantum schemes that we presented in chapter two on different Android mobile phones, then we discuss the obtained results.

3.1 Development environment

Since we are developing for Android operating system [35], we used Java as our programming language and the following tools and libraries for the benchmarking of our post-quantum schemes:

3.1.1 Key terms

- **Android software development** is the process by which new applications are created for devices running the Android operating system. Apps can be written using Java, C++ or Kotlin using the Android software development kit (SDK).
- **SDK:** The Android software development kit is a comprehensive set of development tools that include a debugger, libraries, a handset emulator based on the free hypervisor QEMU, sample code, documentation, and tutorials.
- **IDE:** An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools, and a debugger.
- **JDK:** The Java Development Kit is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows. The JDK includes a private JVM and a few other resources to finish the development of a Java Application.
- **AVD:** An Android Virtual Device is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, or Android TV for simulation in the Android Emulator. The AVD Manager is an interface that can be launched from Android Studio to help create and manage AVDs.

- **Memory management:** The Android Runtime (ART) and Dalvik virtual machine use paging and memory-mapping (mmapping) to manage memory. This means that any memory an app modifies whether by allocating new objects or touching memory mapped pages remains resident in RAM and cannot be paged out. The only way to release memory from an app is to release object references that the app holds, making the memory available to the garbage collector. That is with one exception: any memory mapped files in without modification, such as code, can be paged out of RAM if the system wants to use that memory elsewhere [35].

3.1.2 Android Studio [36]

Android Studio is the official Integrated Development Environment (IDE) for Android app development [37], based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance productivity when building Android apps, such as:

- A flexible Gradle-based build system.
- A fast and feature-rich emulator.
- A unified environment where developers can develop for all Android devices.
- Instant Run to push changes to your running app without building a new APK.
- Code templates and GitHub integration to help developers build common app features and import sample code.
- Extensive testing tools and frameworks.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- C++ and NDK support.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine [36].

Table 3.1 shows the minimum requirements to run Android Studio on Microsoft® Windows®

Operation System	Microsoft® Windows® 7/8/10 (32- or 64-bit)
RAM	3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
Storage	2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
Resolution	1280 x 800 minimum screen resolution

Table 3.1 Minimum requirements for Android Studio

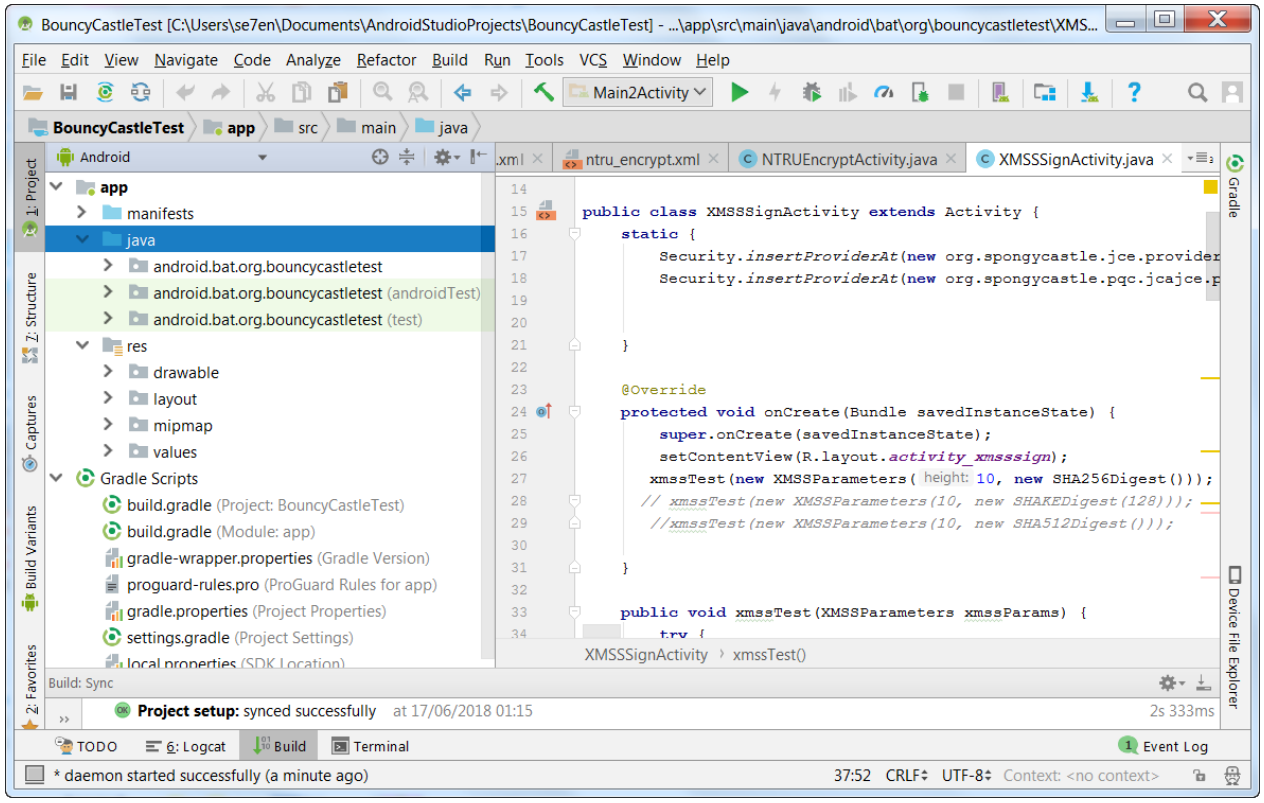

Figure 3.1 Android Studio application window

3.1.3 Android [35]:

Android is an operating system for mobile devices, it was developed by Google and it's based on a modified version of the Linux kernel and other open source software, it's designed primarily for touchscreen mobile devices such as smartphones and tablets. There are other versions of Android like Android TV for televisions, Android Auto for cars, and Wear OS for smart watches, each with a specialized UI. Android also used on digital cameras, game consoles, PCs and other electronics.

Android was initially developed by Android Inc. Google bought the company in 2005, then after two years Android was unveiled, and the first Android device was launched in September 2008, and many versions of the operating system has been launched since, with the current version being 8.1 "Oreo" as shown in **figure 3.2**,

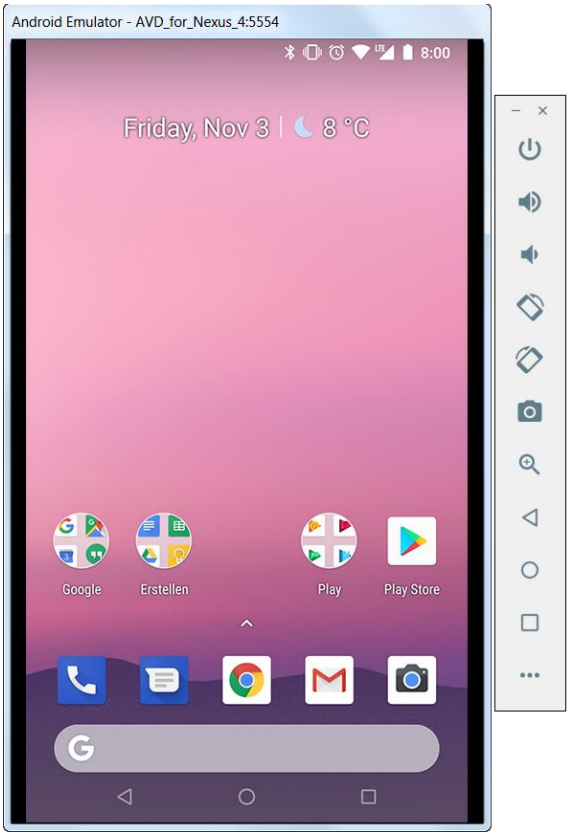

Figure 3.2 current Android 8.1 home screen as seen on the Android Emulator

3.1.4 Bouncy Caslte [38]:

Bouncy Castle is a collection of APIs used in cryptography. It contains APIs for both the Java and the C# programming languages. The APIs are supported by a registered Australian charitable organization: Legion of the Bouncy Castle Inc. The Bouncy Castle Post Quantum provider (BCPQC) contains the implementations of the following schemes:

- Rainbow (multivariate polynomial scheme).
- McEliece (McElieceKobaraImai, McEliecePointcheval, McElieceFujisaki)
- SPHINCS (hash-based signature scheme).
- NH (NewHope: key-exchange scheme).
- XMSS (eXtended Merkle Signature Scheme).
- NTRUEngine (lattice-based scheme).
- GOST3412 (block cipher).
- Blake2s (hash-based signature).

The Android operating system, as of 2014, includes a lightweight version of Bouncy Castle. Due to class name conflicts, this prevents Android applications from including and using the official release of Bouncy Castle. A third-party project called Spongy Castle [49] distributes a renamed version of the library to work around this issue.

3.1.5 FlexiProvider [39]:

The FlexiProvider is a powerful library for the Java Cryptography Architecture (JCA/JCE). It provides cryptographic functions that can be plugged into any application that is built on top of the JCA. The FlexiProvider has been developed by the Theoretical Computer Science Research Group of Prof. Dr. Johannes Buchmann at the Departement of Computer Science at Technische Universität Darmstadt, Germany. The FlexiProvider contains four sub libraries: CoreProvider, ECProvider, PQCProvider, NFProvider.

- **PQCProvider**: The PQCProvider is a provider for cryptographic algorithms which are secure even against quantum attacks. Some of the algorithms are not standardized yet,

but instead are topic of research. The PQCprovider currently contains the implementations of the CMSS signature scheme, the McEliece cryptosystem in four variants, the Niederreiter cryptosystem and the CFS signature scheme.

3.2 Application characteristics

We benchmarked the performance of the following post-quantum schemes in each category:

- Code-based: McEliece, McEliece Kobara-Imai, Niederreiter, Niederreiter-CFS.
- Lattice-based: NTRUEncrypt, NTRUSign.
- Hash-based: XMSS.
- Multivariate polynomial: Rainbow.

3.2.1 Application architecture

Our Android application contains several independent Android Activities that branch out from the main user interface, each one can perform different cryptographic tasks.

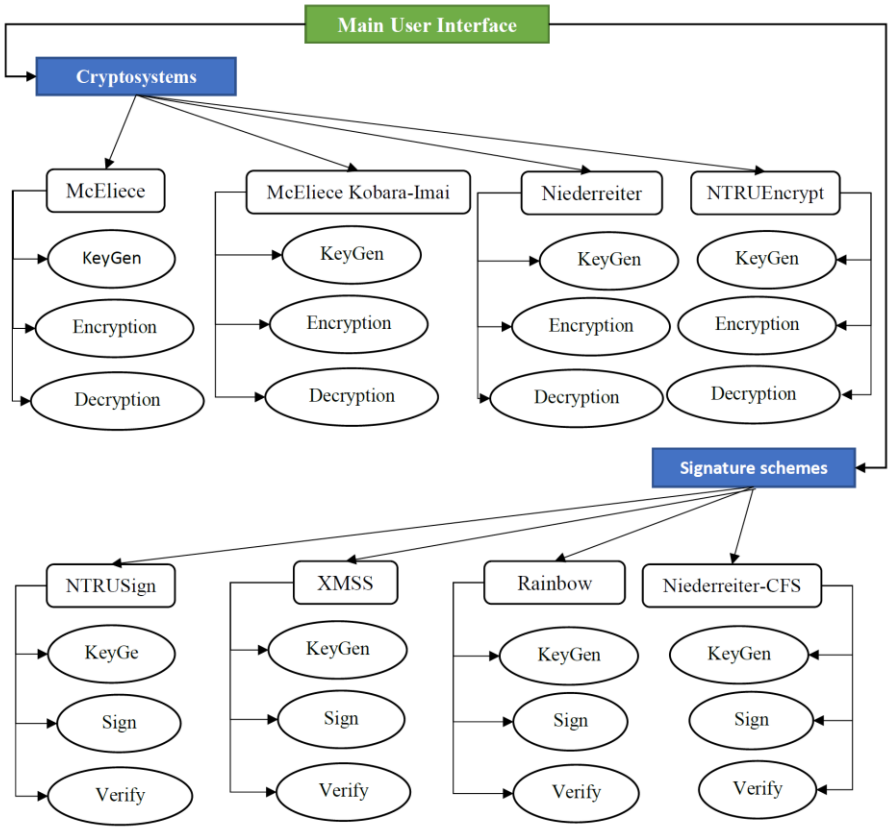

Figure 3.3 Application architecture

3.2.2 Application specifications

Our Android application can just run on android smartphones that can run android version 4.4 or higher, and it contains 11 java classes with their own xml layout.

3.2.3 Application screenshots

The main Activity of our Android application contains the buttons that allows us to choose a type of the post-quantum scheme that we are going to run, either a Cryptosystem or a signature scheme.

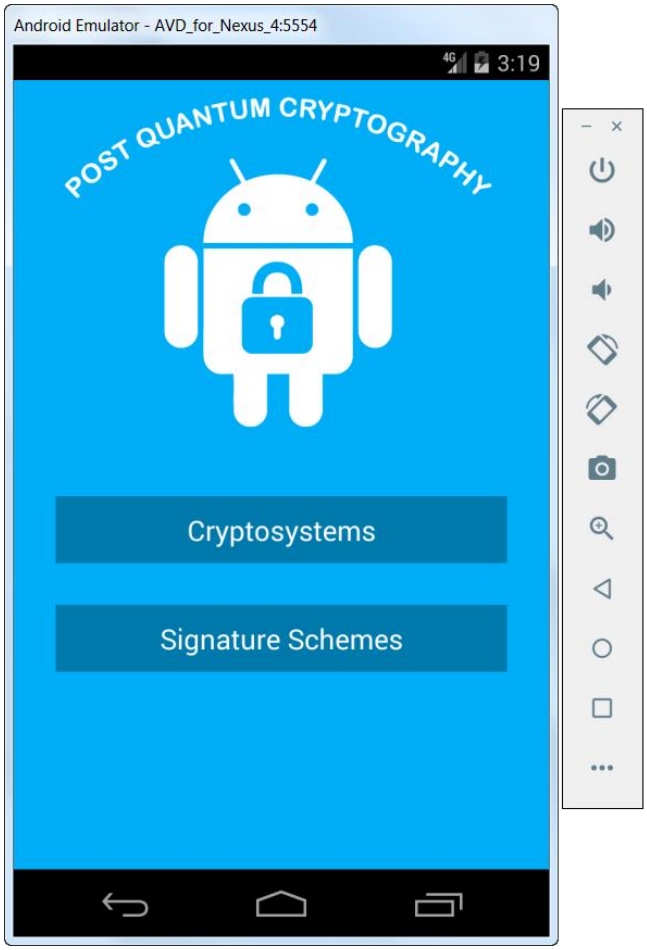

Figure 3.4 Application main user interface

For example, if we click on the Cryptosystems button, the following activity will be shown:

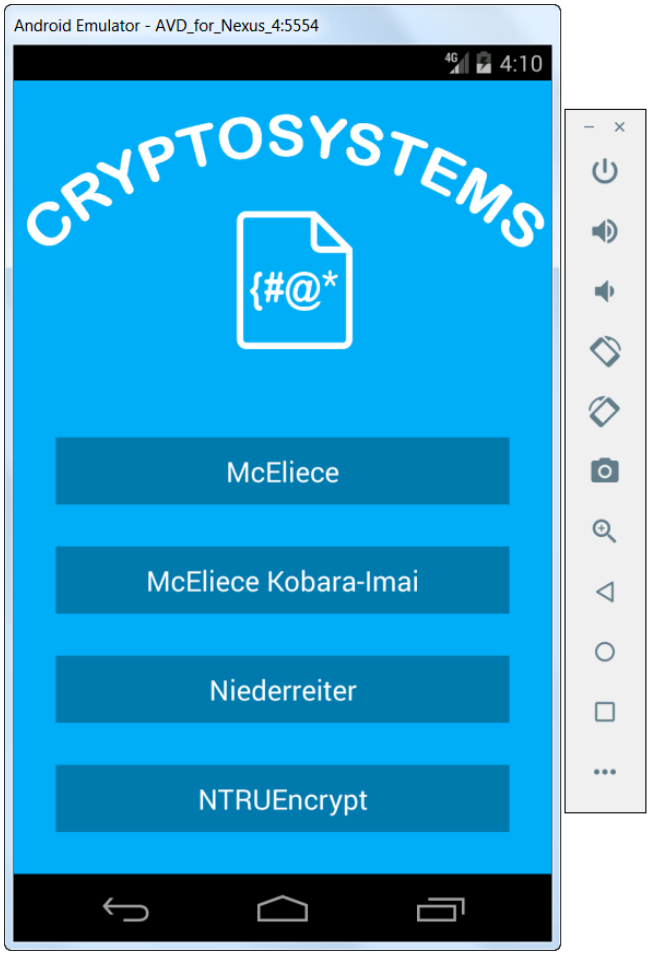

Figure 3.5 Application Cryptosystems Activity

Choosing a cryptosystem will take us to the Activity that allows us to perform the cryptographic tasks for that cryptosystem.

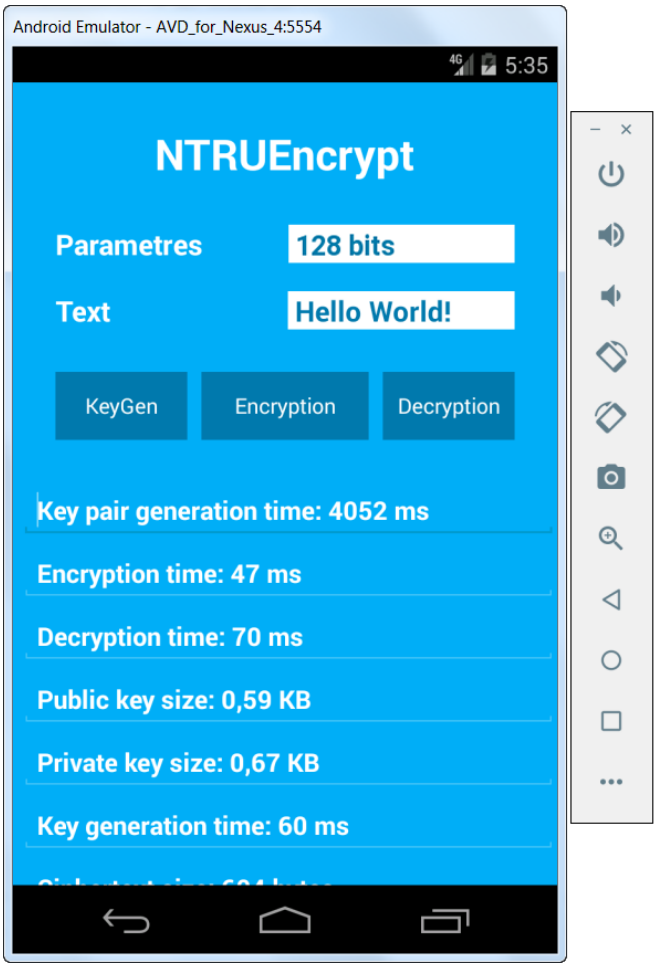

Figure 3.6 Application NTRUEncrypt Activity

In this Activity, first, we choose a security parameter for the cryptosystem, for example in this figure, the selected parameter gives us a security level of 128 bits, then we specify a plaintext in the text field, after that we can generate the key pair according to the selected parameter, then we can encrypt or decrypt the text by clicking on the corresponding buttons, and different statistics will be shown accordingly.

3.2.4 Security and parameters

NIST recommends that all data that has to stay secure for more than 10 years should use a minimum of 128-bit security level [32], the following table represents the different parameters we used for McEliece, McEliece Kobara-Imai, Niederreiter, Niederreiter-CFS, NTRUEncrypt, NTRUSign, XMSS, Rainbow schemes in order to benchmark their performance on

smartphones, and we have chosen the parameters taking into consideration the level of security they offer.

Parameters	**Security level (bits)**	**Algorithm**
m=10, t=50 [7]	60	McEliece, McEliece Kobara-Imai, Niederreiter, Niederreiter-CFS
m=11, t=35 [42]	83	
m=12, t=41 [42]	128	
N=239, p=1024 [43]	85	NTRUEncyprt, NTRUSign
APR2011_439 [44]	128	
APR2011_743 [44]	256	
xmss- shake_10_128 [45]	80	XMSS
xmss-sha2_10_256 [45]	128	
xmss-sha2_10_512 [45]	256	
v1=19, o1=12, o2=13 [46]	80	Rainbow
v1=36, o1=21, o2=22 [46]	128	
v1=86, o1=45, o2=46 [46]	256	

Table 3.2 Chosen parameters for post-quantum schemes

3.3 Smartphones specifications

We used different Android smartphones in our benchmark. The table below shows the model and different specifications for each phone.

Device	**Model**	**CPU**	**Android version**	**Internal memory**	**Internal battery**
LG Leon	LG-H324	Snapdragon 410 Quad-core 1.2 GHz	5.0.1 Lollipop	1024 MB	1900 mAh
Samsung A5	SM-A500H	Exynos 7880 Octa-core 1.9 GHz	6.0.1 Marshmallow	3072 MB	3000 mAh

Table 3.3 Specifications of the chosen android smartphones [33,34]

3.4 Benchmarking methods

Here we describe the methods that we used to obtain the results from each cryptographic scheme that we tested.

3.4.1 computation speed

To measure the computation speed, we used the following method of java.lang.System class that returns the current time in milliseconds:

- *long currentTimeMillis();*

Then we measure the computation speed by using the following code:

```
Long start, finish;
Start = System.currentTimeMillis();
methodCall();
Finish = System.currentTimeMillis();
Long duration = finish – start
```

3.4.2 Memory usage

To measure the memory usage, we used the following two methods of java.lang.Runtime class:

- *long totalMemory();*
- *long freeMemory ();*

The first method returns the total amount of memory in the Java virtual machine, and the second method returns the amount of free memory in the Java Virtual Machine, we subtract the returned values to get the used amount of memory:

```
Runtime runtime = Runtime.getRuntime();
long usedMemoryBefore = runtime.totalMemory() - runtime.freeMemory();
methodCall();
long usedMemoryAfter = runtime.totalMemory() - runtime.freeMemory();
long memoryUsage = usedMemoryAfter-usedMemoryBefore;
```

3.4.3 Energy consumption

To measure the energy consumption, we used “Batterystats” which is a tool included in the Android framework that collects battery data on smartphones. We use the command-line tool to

dump the collected battery data to our development machine and then we create a report we can analyze to extract the amount of battery consumption for each run of our application.

3.5 Results

When benchmarking the performance of the chosen schemes, we used a fixed clear text that is 16 bits (2 bytes) in size.

The following tables shows the results of the different measurements that we obtained using different parameters to obtain the desired security levels.

3.5.1 McEliece

Computation speed, keys size, memory usage and energy consumption of the McEliece scheme on different smartphones with different parameters are shown in the following table:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	60	83	128	60	83	128
key pair generation (ms)	112469	188715	1105292	57104	49276	320313
Encryption (ms)	58	32	27	51	20	11
Decryption (ms)	907	1392	3321	128	150	364
PK size (kBytes)	65,55	415,80	1802,05	65,55	415,80	1802,05
SK size (kBytes)	103,64	443,77	1851,61	103,64	443,77	1851,61
Cipher text size (Bytes)	128	256	512	128	256	512
Memory usage (MB)	0,89	7,99	35,43	1,41	4,74	33,57
Energy consumption (mAh)	1,52	2,28	17,1	3	2,8	18,3

Table 3.4 McEliece cryptosystem benchmarking results

3.5.2 McEliece Kobara-Imai

Computation speed, keys size, memory usage and energy consumption of the McEliece Kobara-Imai CCA2 conversion scheme on different smartphones with different parameters are shown in **table 3.5**:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	60	83	128	60	83	128
key pair generation (ms)	150587	244679	738165	67045	37392	64478
Encryption (ms)	92	191	382	41	83	134
Decryption (ms)	1032	1226	3513	159	207	463
PK size(kBytes)	32,287	79,63	218,264	32,287	79,63	218,264
SK size (kBytes)	67,848	101,96	256,29	67,848	101,96	256,29
Cipher text size (Bytes)	128	256	512	128	256	512
Memory usage (MB)	3,36	3,25	3,77	0,52	3,36	9,20
Energy consumption (mAh)	3,42	4,75	5,32	6,8	3,7	6,3

Table 3.5 McEliece Kobara-Imai cryptosystem benchmarking result

3.5.3 Niederreiter

Computation speed, keys size, memory usage and energy consumption of the Niederreiter scheme on different smartphones with different parameters are shown in the following table:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	60	83	128	60	83	128
key pair generation (ms)	119700	73644	60352	46123	16161	63115
Encryption (ms)	89	169	194	40	79	145
Decryption (ms)	962	1523	3530	155	213	475
PK size (kBytes)	32,27	78,25	216,74	32,27	78,25	216,74
SK size (kBytes)	36	24,12	40,07	36	24,12	40,07
Cipher text size (Bytes)	63	49	62	63	49	62
Memory usage (MB)	1,68	2,66	3,13	0,09	3,44	1,35
Energy consumption (mAh)	2,47	2,09	4,75	*0,9*	*0,3*	*1,5*

Table 3.6 Niederreiter scheme benchmarking result

3.5.4 Niederreiter-CFS

Computation speed, keys size, memory usage and energy consumption of the Niederreiter-CFS scheme on different smartphones with different parameters are shown in the following table:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	60	83	128	60	83	128
key pair generation (ms)	119700	73644	60352	46123	16161	63115
Signature generation (ms)	/	/	/	/	/	/
Signature verification (ms)	/	/	/	/	/	/
PK size (kBytes)	32,27	78,25	216,74	32,27	78,25	216,74
SK size (kBytes)	36	24,12	40,07	36	24,12	40,07
Signature size (kBytes)	/	/	/	/	/	/
Memory usage (MB)	1,68	2,66	3,13	0,09	3,44	1,35
Energy consumption (mAh)	2,47	2,09	4,75	0,9	0,3	1,5

Table 3.7 Niederreiter-CFS scheme benchmarking result

3.5.5 NTRUEncrypt

Computation speed, keys size, Signature size, memory usage and energy consumption of the NTRUEncrypt scheme on different smartphones with different parameters are shown in the following table:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	85	128	256	85	128	256
key pair generation (ms)	18789	27671	36352	2532	4052	5741
Encryption (ms)	45	203	472	9	47	112
Decryption (ms)	33	304	829	6	70	164
PK size (kBytes)	0,29	0,59	0,99	0,29	0,59	0,99
SK size (kBytes)	0,34	0,67	1,14	0,34	0,67	1,14
Cipher text size (Bytes)	299	604	1022	299	604	1022
Memory usage (MB)	2,35	3,88	3,73	3,27	0,22	0,69
Energy consumption (mAh)	0,19	0,57	0,95	0,3	0,3	0,9

Table 3.8 NTRUEncrypt scheme benchmarking results

3.5.6 NTRUSign

Computation speed, keys size, Signature size, memory usage and energy consumption of the NTRUSign scheme on different smartphones with different parameters are shown in the following table:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	85	128	256	85	128	256
key pair generation (ms)	18789	27671	36352	2532	4052	5741
Signature generation (ms)	228	1406	2696	41	262	613
Signature verification (ms)	167	469	1024	29	91	254
PK size (kBytes)	0,29	0,59	0,99	0,29	0,59	0,99
SK size (kBytes)	0,34	0,67	1,14	0,34	0,67	1,14
Signature size (kBytes)	0,30	0,59	1,00	0,30	0,59	1,00
Memory usage (MB)	13,22	39,69	29,18	27,90	49,01	34,52
Energy consumption (mAh)	2,09	7,79	42,94	1,7	6,3	28

Table 3.9 NTRUSign scheme benchmarking results

3.5.7 XMSS scheme

Computation speed, keys size, Signature size and memory usage of the XMSS scheme on different smartphones with different parameters are shown in the following table:

Device	**LG-H324**			**SM-A500H**		
Security level (bits)	64	128	256	64	128	256
key pair generation (ms)	/	/	/	487436	352714	1290208
Signature generation (ms)	/	/	/	593	486	1851
Signature verification (ms)	/	/	/	248	199	846
PK size (kBytes)	0,063	0,063	0,125	0,063	0,063	0,125
SK size (kBytes)	2,264	2,264	3,014	2,264	2,264	3,014
Signature size (kBytes)	2,44	2,44	8,88	2,44	2,44	8,88
Memory usage (MB)	/	/	/	18,33	5,61	1,54
Energy consumption (mAh)	/	/	/	24,9	18	65,8

Table 3.10 XMSS scheme benchmarking results

3.5.8 Rainbow

Computation speed, keys size, Signature size and memory usage of the Rainbow scheme on different smartphones with different parameters are shown in the following table:

Device	LG-H324			SM-A500H		
Security level (bits)	80	128	256	80	128	256
key pair generation (ms)	759219	/	/	91001	855157	/
Signature generation (ms)	510	/	/	57	276	/
Signature verification (ms)	112	/	/	15	77	/
PK size (kBytes)	25,464	136,359	/	25,464	136,359	/
SK size (kBytes)	30,04	156,24	/	30,04	156,24	/
Signature size (kBytes)	0,04	0,08	/	0,04	0,08	/
Memory usage (MB)	2,44	/	/	4,48	4,82	/
Energy consumption (mAh)	17,5	/	/	4,5	42	/

Table 3.11 Rainbow scheme benchmarking results

3.6 Comparison

The following figures compare the obtained results from different post-quantum schemes, we chose the 128 bits security parameters that we tested on the SM-A500H smartphone. and we excluded Niederreiter-CFS from signature schemes because it does not fit the comparison results in mobile environment between NTRU, XMSS and Rainbow schemes.

3.6.1 Key generation speed

Figure 3.7 shows a comparison between: McEliece, McEliece KobaraImai, Niederreiter, NTRU, XMSS and Rainbow in terms of key generation speed:

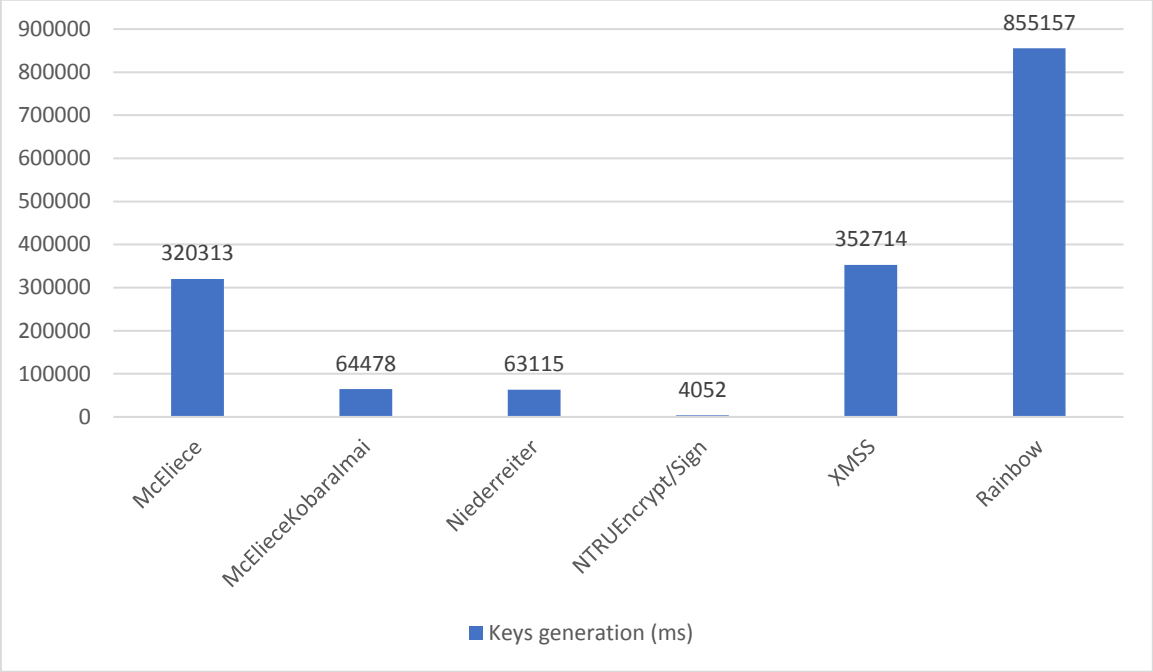

Figure 3.7 Key generation speed comparison between McEliece, McEliece KobaraImai, Niederreiter, NTRU, XMSS and Rainbow in Mobile SM-A500H.

3.6.2 Encryption/Decryption speed

Figure 3.8 shows a comparison between: McEliece, McEliece KobaraImai, Niederreiter, NTRU in terms of encryption and decryption speed:

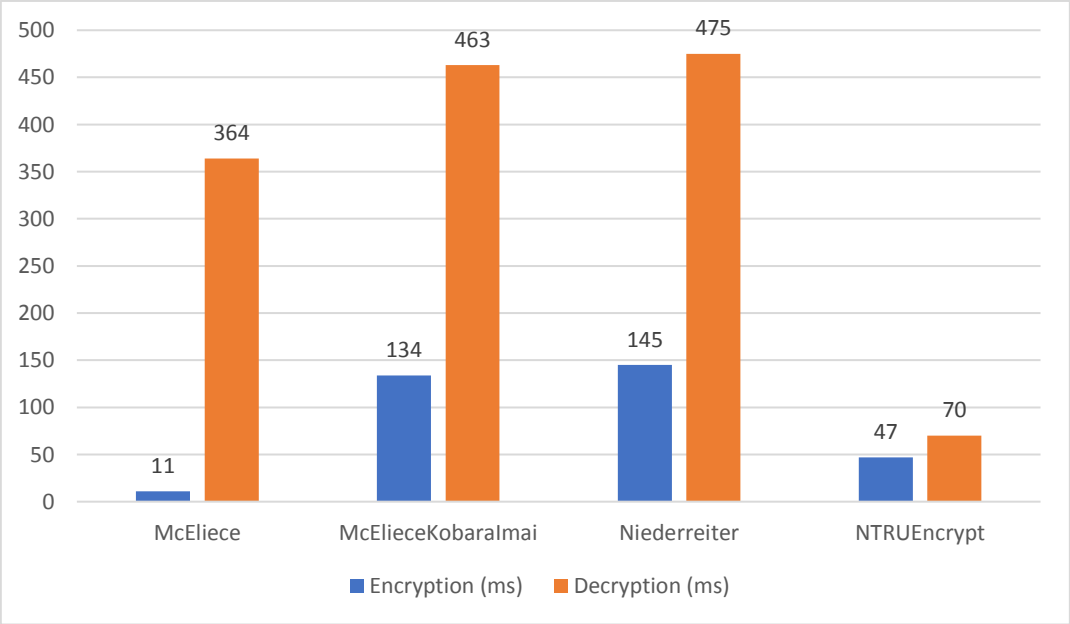

Figure 3.8 Encryption/Decryption speed comparison between McEliece, McEliece KobaraImai, Niederreiter, NTRU in Mobile SM-A500H.

3.6.3 Signing/Verification speed

Figure 3.9 shows a comparison between: NTRUSign, XMSS and Rainbow in terms of signing and verification speed:

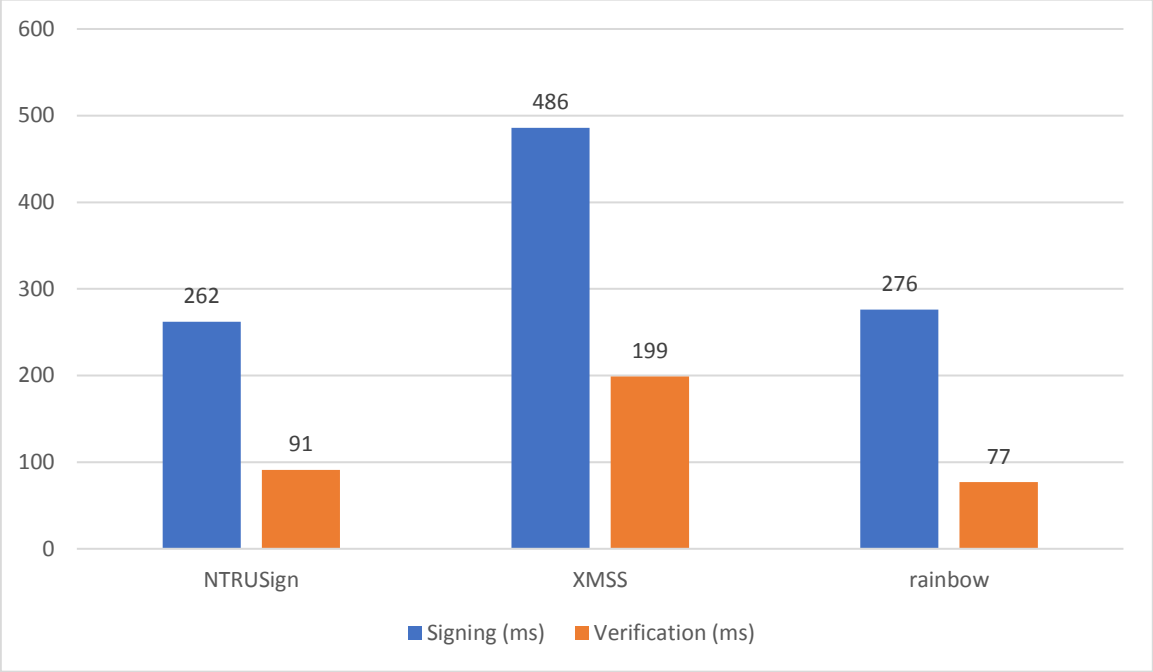

Figure 3.9 Signing/Verification speed comparison between NTRUSign, XMSS and Rainbow in Mobile SM-A500H.

3.6.4 Key size

Figure 3.10 shows a comparison between: McEliece, McEliece KobaraImai, Niederreiter, NTRU, XMSS and Rainbow in terms of key size:

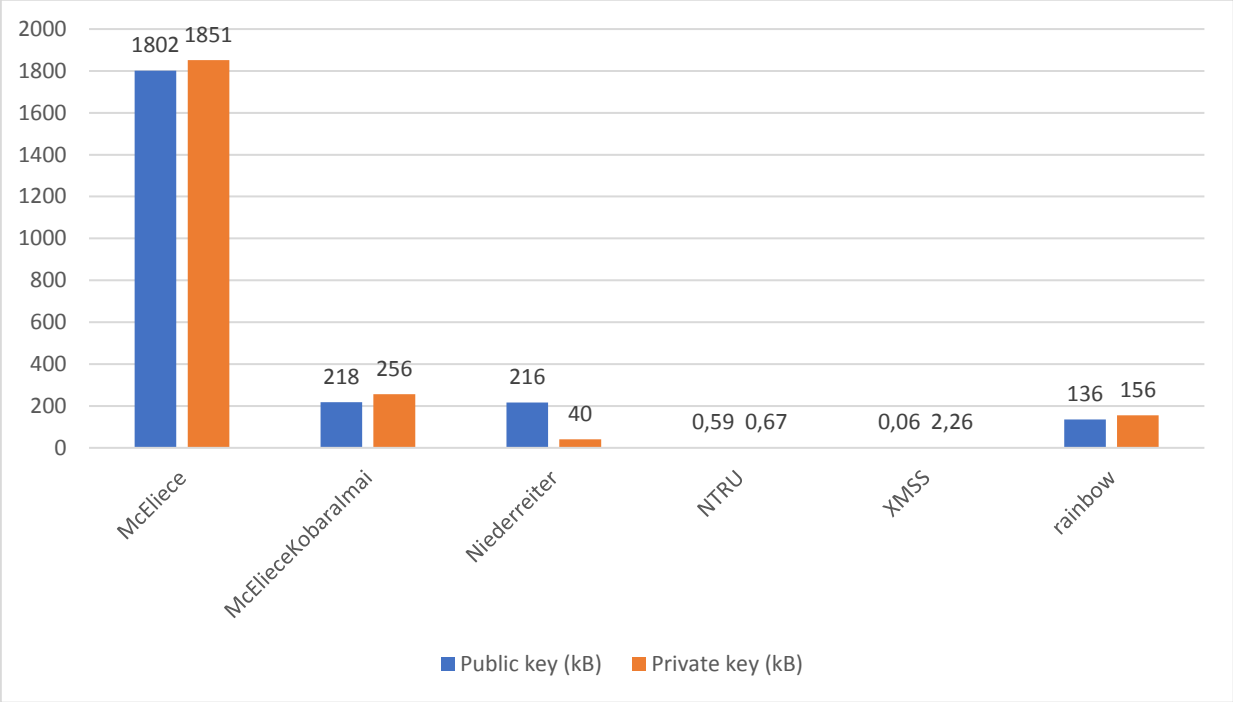

Figure 3.10 Key size comparison between McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt/Sign, XMSS and Rainbow in Mobile SM-A500H.

3.6.5 Cipher text size

Figure 3.11 shows a comparison between: McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt in terms of ciphertext size:

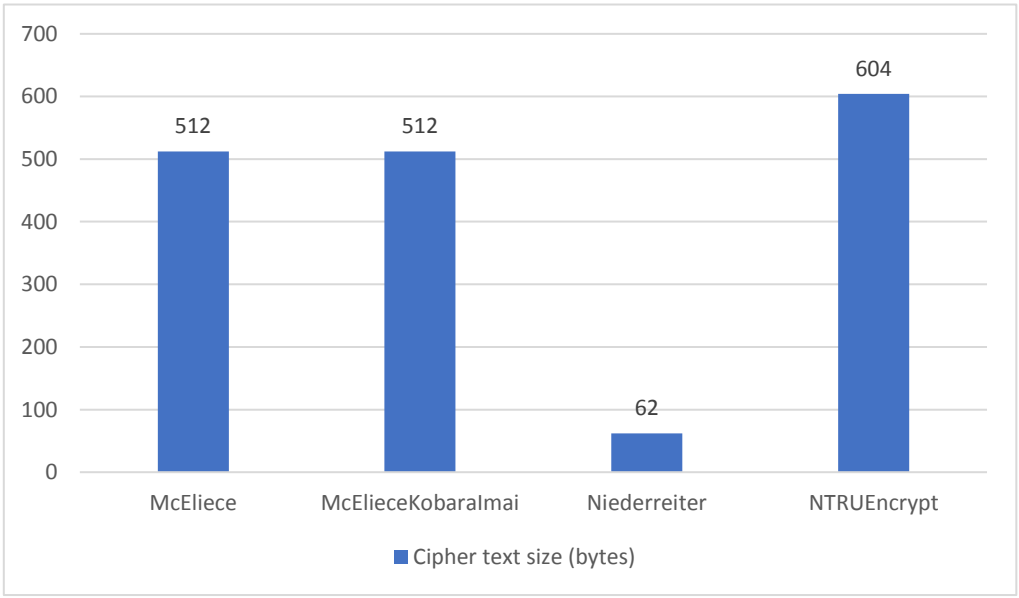

Figure 3.11 Cipher text size comparison between McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt in Mobile SM-A500H.

3.6.6 Signature size

Figure 3.12 shows a comparison between: NTRUSign, XMSS and Rainbow in terms of signature size:

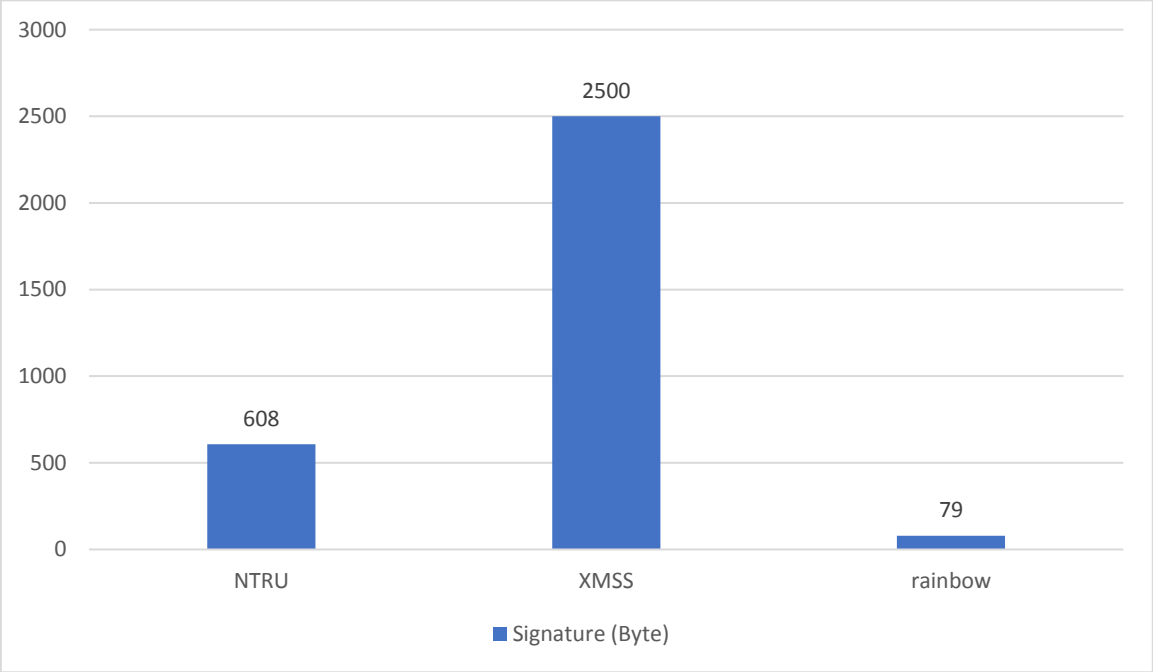

Figure 3.12 Signature size comparison between NTRUSign, XMSS and Rainbow in Mobile SM-A500H.

3.6.7 Memory usage

Figure 3.13 shows a comparison between: McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt, NTRUSign, XMSS and Rainbow in terms of memory usage:

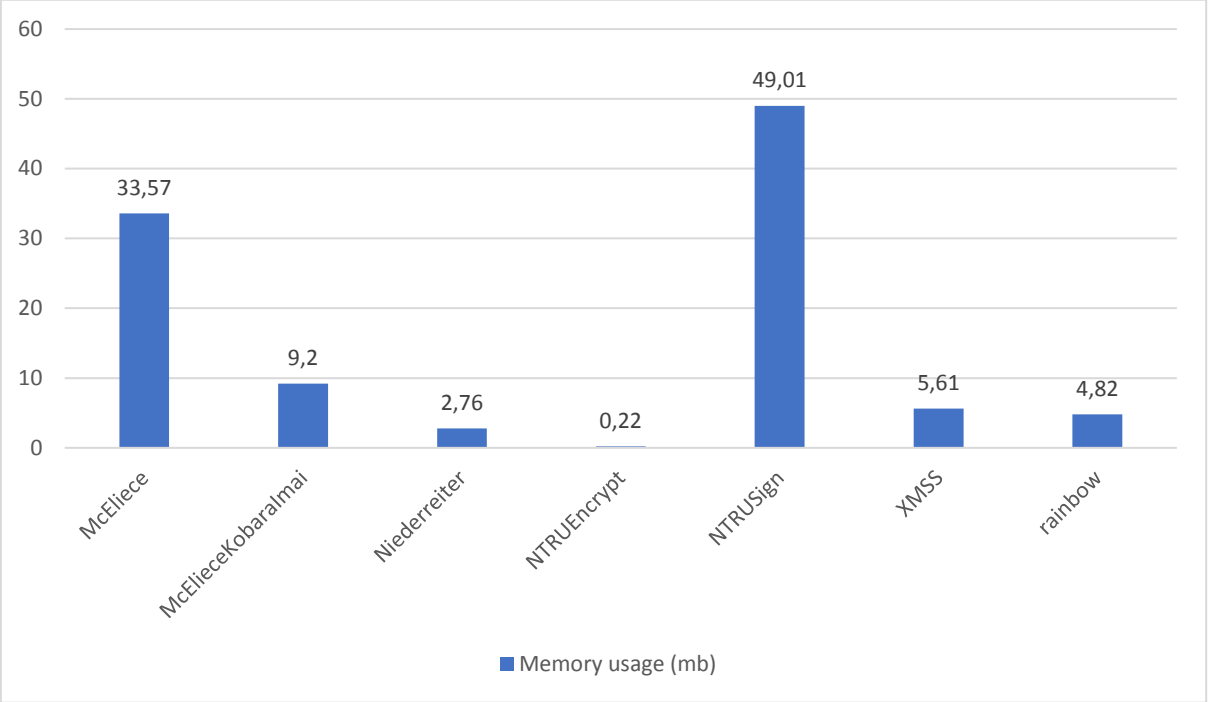

Figure 3.13 Memory usage comparison between McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt, NTRUSign, XMSS and Rainbow in Mobile SM-A500H.

3.6.8 Energy consumption

Figure 3.14 shows a comparison between: McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt, NTRUSign, XMSS and Rainbow in terms of energy consumption:

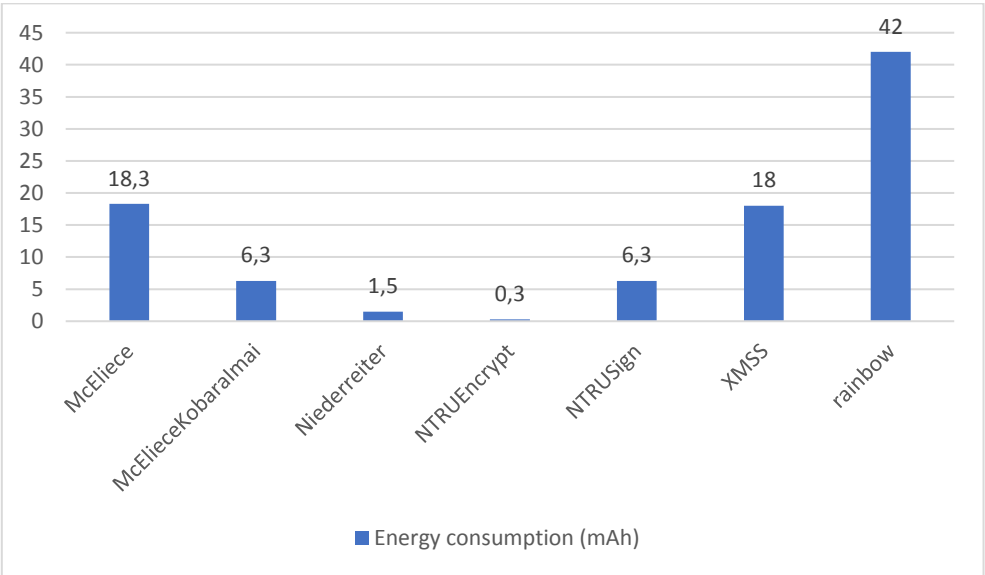

Figure 3.14 Energy consumption between McEliece, McEliece KobaraImai, Niederreiter, NTRUEncrypt, NTRUSign, XMSS and Rainbow in Mobile SM-A500H.

3.7 Discussion

3.7.1 Computation speed

- NTRUEncrypt and NTRUSign are the best schemes when it comes to speed of execution, and except for NTRUSign, the signature schemes perform the worst when generating their keys. but that may not be a problem since they can be reused multiple times for signing and encrypting operations.
- Rainbow performed the worst when it comes to key generation, but the operations of signing and verification were fast and fairly similar to the NTRUSign scheme.
- Code based schemes (McEliece, McEliece-kobaraImai, Niederreiter) performance suffer when generating keys due to the large keys they require, and that make them hard to use in constrained environments, like on mobile phones or on IoT devices.
- Code based schemes performed very well on the encryption process, with McEliece coming on top, the CCA2 secure conversion of the scheme led to slower results because of the added steps to the encryption procedure. When we compared the code based schemes to the NTRUEncrypt scheme, we found that NTRUEncrypt offers a balanced execution speeds for encryption and decryption.
- When performing signing operations, the NTRUSign and the Rainbow schemes performed very close, with the NTRUSign having the edge when it comes to generating signatures, the XMSS scheme needed almost double the time to achieve what the NTRUSign and the Rainbow schemes did, The Niederreiter-CFS signature scheme performed the worst on the signing operation, We could not get past 20 bits of security parameters due to the fact that code based schemes use a lot of computational resources and therefore are not suitable for digital signatures [41]

3.7.2 key size

- Code based schemes require large key sizes especially the McEliece scheme with an astounding 3,8 mb for a key pair of a security level of 128 bits, and that make them hard to use when transmitting those keys, especially when used in a resource constrained environment like on mobile phones or on IoT devices.

- NTRUSign and XMSS have the lowest key sizes (less than 1 kB for NTRUSign) which make them perfect for key exchanging and storing procedures, the Rainbow signature scheme have somewhat large keys (136 kB for a public key with a security level of a128 bits).
-

3.7.3 Signature and cipher text size

- Niederreiter encryption scheme generates the lowest ciphertext sizes (8 times smaller than any other encryption scheme that we tested) which makes it the best scheme when storing or transmitting the ciphertext is a concern. In contrast, the NTRUEncrypt generates the biggest ciphertext sizes, but it's comparable to the McEliece scheme and its CCA2 conversion.
- Rainbow signature scheme generates the lowest signatures (7 times smaller than NTRUSign signatures and 31 times smaller when compared to the XMSS scheme) and that makes it the best scheme for signature transmission. The XMSS scheme generates the biggest signatures with a 2,4-kB signature for a security level of a 128 bits.

3.7.4 Memory usage

- NTRUEncrypt used the lowest amount of memory when performing the operations of encryption and decryption, on the other hand NTRUSign occupied the biggest amount of memory compared to the other signature schemes, which is a problem if the used environment is constrained on memory.
- Niederreiter scheme performed well on the memory consumption test, in contrast, the McEliece scheme and it's kobara-Imai CCA2 variant occupied a lot of memory due to the generation of the large keys they needed.

3.7.5 Energy consumption

- When it comes to energy consumption, we noticed that the more it takes a scheme to perform its various calculations the more it consumes energy, for instance, the Rainbow scheme performed very well when generating signatures, but it performed the worst when generating its key pairs, that's why it also performed the worst on energy consumption. The extensive use of memory also led to a higher consumption of energy, like what we noticed about the NTRUSign when generating signatures. and since the Mceliece scheme performed the worst between the

other code-based schemes when generating its key pairs, it consumed a lot of energy, which can be problematic for environments that are constrained on it.

- The NTRUEncrypt scheme used the least amount of energy on either encryption or decryption operations, and also when generating digital signatures.

3.8 Findings

Table 3.11 shows the advantages and disadvantages of each tested scheme:

Scheme	**Type**	**Advantages**	**Disadvantages**
McEliece	Code-based	Very fast encryption process	• Very large key sizes • High memory usage and energy consumption
McEliece Kobara-Imai	Code-based	More secure McEliece variant	Large key sizes
Niederreiter	Code-based	• Lowest cipher text size • Low memory usage	Large key sizes
Niederreiter-CFS	Code-based		• Resource-intensive signature generation process • Large key sizes
NTRUEncrypt	Lattice-based	• Very fast execution • Lowest key size • Lowest memory usage and energy consumption	Somewhat large cipher text sizes
NTRUSign	Lattice-based	• Very fast execution • Lowest key size	High memory usage and energy consumption (signature generation)
XMSS	Hash-based	small key sizes	• Somewhat large signature sizes • High energy consumption

Rainbow	Multivariate polynomial	• Very fast signature generation and verification • Smallest signature size	• Long key generation process • High energy consumption

Table 3.11 Advantages and disadvantages of each tested scheme

3.9 Conclusion

In this chapter, we benchmarked the performance of the algorithms that we presented in chapter two using different security parameters and different Android smartphones, then we compared the performance of each algorithm in different categories, after that we discussed the obtained results. **Table 3.11** resumes our findings:

GENERAL CONCLUSION

GENERAL CONCLUSION

From our findings, each post-quantum algorithm has some advantages and disadvantages, NTRUEncrypt and NTRUSign schemes performed the best compared to other schemes on both smartphones that we used and they have the best tradeoffs, which make them our recommended post quantum schemes for the smartphone environment, it should be noted that NTRUSign can use some improvements in memory consumption when generating digital signatures. Code based schemes have been around for so long and they are still unbroken, they performed well especially on ciphertext size but their key sizes crippled their performance to some extent, and they can be a great area of research for optimization, one example is the QC-MDPC McEliece variant [9], which claims to offer much lower key sizes. The Rainbow multivariate-based scheme generates the lowest signature sizes and it's very fast, but it takes too long to generate its key pair, which leads to great energy consumptions. The XMSS hash-based scheme has acceptable signature generation speeds, and lower key sizes, but the generated signatures are too big and could use some optimizations in that regard.

BIBLIOGRAPHIE

[1] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, The 35th Annual Symposium on Foundations of Computer Science, Santa Fe USA, 1994.

[2] L. Grover, A fast quantum mechanical algorithm for database search, The Twenty-Eighth Annual ACM Symposium on Theory of Computing, New York, USA, 1996.

[3] L. Chen, S. Jordan and Y. Liu, NIST Internal Report 8105, Report on Post-Quantum Cryptography, 2016.

[4] NSA, www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml, retrieved: 26/10/2017

[5] D. Bernstein, J. Buchmannet and E. Dahmen, Post-Quantum Cryptography, Spinger, p147, Berlin, 2009.

[6] D. Bernstein, J. Buchmannet and E. Dahmen, Post-Quantum Cryptography, Spinger, p35, Berlin, 2009.

[7] R. McEliece, A Public-Key Cryptosystem based on Algebraic Coding Theory, The Deep Space Network Progress Report, DSN PR 42-44, 1978, pp. 114-116

[8] H. Niederreiter, Knapsack-type cryptosystems and algebraic coding theory, Problems of Control and Information Theory, 15, 1986, pp. 159-166.

[9] R. Misoczki, N. Sendrier, Mdpcmceliece: New mceliece variants from moderate density parity-check codes, Project SECRET, INRIA-Rocquencourt, France, 2012.

[10] PQCRYPTO, Initial Recommendations of Long-Term Secure Post-Quantum Systems, Commission of the European Communities, Horizon 2020 program, project no. 645622, 2015.

[11] J. Ding, D. Schmidt, Rainbow: a new multivariable polynomial signature scheme, Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol 3531, 2005, pp. 164–175.

[12] J. Patarin, N. Courtois and L. Goubin, QUARTZ, 128-bit long digital signatures. CT-RSA, San Francisco USA, 2001, pp. 282–297.

[13] B. David, C. Amalavoyal, Efficient Networks for Quantum Factoring, Physical Review A54, 1996, pp. 1034-1063.

[14] C. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27, 1948, pp. 379, 423, 623-656.

[15] D. Bernstein, J. Buchmannet and E. Dahmen, Post-Quantum Cryptography, Spinger, Berlin, 2009, pp. 13-14.

[16] D. Butin, S.Gazdag, and J. Buchmann, Real-world post quantum digital signatures, Cyber Security and Privacy Forum, 530, 2015, pp. 41–52.

[17] D. Jao, L. De Feo, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, PQCrypto'11, Berlin Germany, 2011, pp. 19-34.

[18] L. Chen, S. Jordan and Y. Liu, Report on post-quantum cryptography. National Institute of Standards and Technology Internal Report, 8105, 2016.

[19] R. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, New York USA, 1978

[20] N. James, B. Elaine, Report on the Development of the Advanced Encryption Standard (AES), Journal of Research of the National Institute of Standards and Technology,106, 2001, pp. 511-577.

[21] National Institute of Standards and Technology, Digital Signature Standard (DSS), FIPS Publication, 186-4, 2013.

[22] T. Yasuda, D. Xavier, MQ Challenge: Hardness Evaluation of Solving Multivariate Quadratic Problems, IACR Cryptology ePrint Archive, 2015, 2015, pp. 275-289

[23] R. McEliece, A Public-Key Cryptosystem Based on Algebraic Coding Theory, Deep Space Network Progress Report, 44, 1978, pp. 114-116.

[24] H. Niederreiter, Knapsack-type cryptosystems and algebraic coding theory, Problems of Control and Information Theory, 15, 1986, pp. 159.

[25] J. Hoffstein, J. Pipher and J. H. Silverman, NTRU: A ring-based public key cryptosystem, Springer, Berlin Heidelberg, 1998.

[26] ANSI/IEEE 1363.1-2008: IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices, standards.ieee.org/findstds/standard/1363.1-2008.html, 26/04/2018

[27] J. Ding, D. Schmidt, Rainbow: a new multivariable polynomial signature Scheme, Springer, Berlin Heidelberg, 2005.

[28] Albrecht, B.Stanislav and B. Johannes, Selecting parameters for the rainbow signature scheme-extended version, IACR Cryptology ePrint Archive, 2010, 2010, pp. 437-458

[29] K. Kobara, H. Imai, Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC, Springer, Berlin Heidelberg, 2001, pp. 19-35.

[30] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, CRYPTO '99, Santa Barbara USA, 1999.

[31] D. Pointcheval, Chosen-Ciphertext Security for Any One-Way Cryptosystem, PKC '00, Victoria Australia, 2000.

[32] E. Barker, Recommendation for key management part 1: General, NIST SP, 800-57 Pt1 Rev 4, 2016, pp. 69.

[33] gsmarena, www.gsmarena.com/lg_leon-7053.php, Retrieved: 08/06/2018.

[34] gsmarena, www.gsmarena.com/samsung_galaxy_a5_(2017)-8494.php, Retrieved: 08/06/2018.

[35] Android, www.android.com, Retrieved: 08/06/2018.

[36] Android, developer.android.com/studio/index.html, Retrieved: 08/06/2018.

[37] Android developers blog, android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html, Retrieved: 08/06/2018.

[38] Bouncycastle, www.bouncycastle.org, Retrieved: 08/06/2018.

[39] Flexiprovider, www.flexiprovider.de, Retrieved: 08/06/2018.

[40] A. Canteaut, N. Sendrier, Cryptanalysis of the Original McEliece Cryptosystem, asiacrypt98, Beijing, 1998.

[41] S. Heyse, Post Quantum Cryptography:Implementing Alternative Public Key Schemes On Embedded Devices, Doctorat, Bochum Germany, 2013.

[42] D. Bernstein, T. Chou, and P. Schwabe, McBits: Fast Constant-Time Code-Based Cryptography, CHES 2013, Santa Barbara USA, 2013.

[43] J. Hoffstein, J. Pipher and J. Schanck, Choosing Parameters for NTRUEncrypt, CT-RSA, San Francisco USA, 2017.

[44] K. Jarvis, M. Nevins, NTRU over the Eisenstein Integers, Des. Codes Cryptography, 74, 2015, pp. 219-242.

[45] A. Hülsing, D. Butin and S. Gazdag, XMSS eXtended Merkle Signature Scheme, RFC, 8391, 2018, pp. 1-74.

[46] A. Petzoldt, S. Bulygin, and J. Buchmann, Selecting Parameters for the Rainbow Signature Scheme, PQCrypto'10, Darmstadt Germany, 2010

[47] D. Bernstein, J. Buchmannet and E. Dahmen, Post-Quantum Cryptography, Spinger, Berlin, 2009, pp.104-107

[48] C. Gentry, M. Szydlo, Cryptanalysis of the Revised NTRU Signature Scheme, EUROCRYPT 2002, Berlin Germany, 2002

[49] Github, rtyley.github.io/spongycastle Retrieved: 08/06/2018

[50] IBM, www.ibm.com, Retrieved: 08/06/2018

[51] C. Paar, J. Pelzland, Understanding Cryptography, Spinger, Berlin, 2010, pp.2-11

[52] N. Courtois, M. Finiasz, and N. Sendrier, How to Achieve a McEliece-Based Digital Signature Scheme, ASIACRYPT2001, Gold Coast Australia, 2001.

[53] J. Buchmann, E. Dahmen, and A. Hülsing, XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions, PQCrypto'11, Berlin Germany, 2011

[54] SSL2buy, www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences, Retrieved: 08/06/2018

[55] Wikipedia, en.wikipedia.org/wiki/Cryptography, Retrieved: 08/06/2018

ملخص

تعتمد أنظمة تشفير المفتاح العام الحالية (مثل RSA و ECC و ElGamal) على مشكلة التحليل إلى عوامل أو مشكلة اللوغاريتم المنفصل. التقدم الأخير في الحوسبة الكمومية فتح الباب أمام إمكانية تطوير حواسيب كمومية كبيرة ومتطورة بما يكفي لحل هذه المشاكل وكسر أنظمة التشفير الحالية. لحسن الحظ هناك خوارزميات مقترحة يمكن أن تقاوم هذه الهجمات الكمومية ، ويمكن تجميع هذه الخوارزميات في الفئات التالية: التشفير المبني على التجزئة والتشفير القائم على الشفرة والتشفير المبني على المشبك والتشفير متعدد المعادلات التربيعية والتشفير المتماثل فائق التناسق.

الهدف من هذه الأطروحة هو قياس أداء المعالج والذاكرة واستهلاك الطاقة لمختلف الخوارزميات المقاومة للهجمات الكوانتية على هواتف أندرويد (موارد محدودة) لتحديد أي منها مناسب للاستخدام في تلك البيئة.

الكلمات الرئيسية: الخوارزميات المقاومة للهجمات الكوانتية، خوارزميات التشفير ، خوارزميات التوقيع ، الأمن ، الأداء.

Abstract

Current public key schemes such as (RSA, ECC, ElGamal) are based on the factorization or discrete logarithm problems. Recent advancements in quantum computing open the door to the possibility of developing large quantum computers sophisticated enough to solve these problems and break these cryptosystems. Luckily there are proposed algorithms that can resist these quantum attacks, these algorithms can be grouped in the following categories: hash-based cryptography, code-based cryptography, lattice-based cryptography, multivariate quadratic equations cryptography and supersingular isogeny cryptography.

The aim of this dissertation is to benchmark the performance, memory usage and energy consumption of different post-quantum algorithms on Android devices (limited resources) to determine which ones are suitable for usage in that environment.

Keywords : Post quantum cryptography, Encryption algorithms, Signature algorithms, Security, Performance.

Résumé

Les schémas de clé publique actuels tels que (RSA, ECC, ElGamal) sont basés sur le problème de la factorisation ou du logarithme discret. Progrès récents en informatique quantique ouvrent la porte à la possibilité de développer de grands ordinateurs quantiques assez sophistiqués pour résoudre ces problèmes et casser ces cryptosystèmes. Heureusement, il existe des algorithmes qui peuvent résister ces attaques quantiques, ces algorithmes peuvent être regroupés dans les catégories

suivantes : cryptographie à base de hachage, cryptographie à base de codes, réseaux euclidiens, cryptographie multivariée et cryptographie de l'isogénèse supersingulaire.

Le but de cette thèse est de comparer la performance, l'utilisation de mémoire et la consommation de l'énergie de différents algorithmes post-quantique sur des appareils Android (ressources limitées) pour déterminer ceux qui conviennent à l'utilisation dans cet environnement.

Mots clés : Cryptographie post-quantique, algorithmes de cryptage, algorithmes de signature, Sécurité, performance.