REVIEW OF PUBLIC KEY CRYPTOGRAPHY

CHIBUOGWU VICTOR CHUKWUEMEKA

200551029

A PROJECT SUBMITTED TO DEPARTMENT OF MATHEMATICS,
FACULTY OF SCIENCE, LAGOS STATE UNIVERSITY,
IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF
BACHELOR OF SCIENCE DEGREE (B.Sc HONS.) IN MATHEMATICS, AT LAGOS
STATE UNIVERSITY, OJO, LAGOS, NIGERIA

SEPTEMBER, 2024

Certification

I certify that this project work with the title *REVIEW OF PUBLIC KEY CRYPTOGRAPHY* submitted by CHIBUOGWU, VICTOR CHUKWUEMEKA with matriculation number **200551029** was carried out under my supervision in the Department of Mathematics, Faculty of science, Lagos State University, Ojo, Lagos.

....	....
Dr. AbdulKareem A.O.	DATE
Supervisor	
....	....
Dr. Olutimo A.L.	DATE
Head of Mathematics Department	
....	....
External Examiner	DATE

DEDICATION

Dedicated to my family and friends for unwavering support. And to my teachers whose wisdom and guidance made this possible.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to the current Head of the Mathematics Department at Lagos State University, Dr. Olutmimo A.L., and my advisor, Dr. AbdulKareem A.O., for their invaluable guidance and support throughout this research. I also thank my colleagues and friends for their encouragement and assistance. Special thanks to my family for their constant support and understanding.

ABSTRACT

This work presents the use case of Public Key encryption, discussing in detail the models and methods for encryption and decryption. It also illustrates the algorithms of each method, providing examples to solve unconstrained problems.

Contents

Chapter 1

INTRODUCTION

1.1 Introduction to Public Key Cryptography

Public Key Cryptography (PKC), introduced by Whitfield Diffie and Martin Hellman in 1976, revolutionized secure communication by addressing the challenge of key distribution. Unlike traditional symmetric key cryptography, PKC uses a pair of mathematically related keys: a public key for encryption and a private key for decryption. This dual-key system has enabled the development of secure communication protocols and digital signatures, among other applications.

1.2 Background of the Study

Since its inception, PKC has undergone significant development. Major algorithms like RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm), and ECC (Elliptic Curve Cryptography) have been introduced to address various cryptographic challenges. These algorithms are characterized by their computational efficiency, security against attacks, and suitability for different applications. The security of PKC relies on the difficulty of mathematical problems such as factoring large integers or solving discrete logarithm problems in finite fields.

1.3 Statement of Problem

Despite its advantages, PKC faces several challenges:

- **Key Management:** Generating, distributing, and storing keys securely.

- **Computational Efficiency:** Ensuring encryption and decryption operations are fast and scalable.
- **Vulnerabilities:** Addressing vulnerabilities such as side-channel attacks, quantum computing threats, and implementation errors.

1.3.1 Aim and Objectives of the Study

The primary aim of this study is to provide a comprehensive examination of public key cryptography from a mathematical perspective. The specific objectives include:

- **Historical Development:** Tracing the historical development and evolution of public key cryptography.
- **Mathematical Principles:** Explaining the mathematical principles underlying key generation, encryption, and decryption.
- **Real-World Applications:** Evaluating the effectiveness of public key cryptographic systems in various real-world applications.
- **Security Analysis:** Identifying and analyzing major security threats and vulnerabilities associated with public key cryptography.
- **Future Directions:** Suggesting potential improvements and future research directions.

1.4 Significance of the Study

This study advances our understanding of the mathematical foundations of public key cryptography, provides insights into real-world applications and challenges, and guides policymakers in understanding and regulating cryptographic technologies.

1.5 Scope of the Study

The study comprehensively examines public key cryptography, including its theoretical foundations, development from inception to current technologies, real-world applications in secure communication, digital signatures, blockchain, and challenges such as key management and performance optimization. Future directions include post-quantum cryptography. Symmetric key cryptography is covered only for context or comparison.

1.6 Definition of Terms

- **Public Key Cryptography (PKC):** A cryptographic system that uses two keys—a public key for encryption and a private key for decryption.
- **RSA (Rivest-Shamir-Adleman):** A widely used PKC algorithm based on the difficulty of factoring large integers.
- **Digital Signature:** A mathematical scheme for verifying the authenticity and integrity of digital messages or documents.
- **Elliptic Curve Cryptography (ECC):** A PKC algorithm based on the properties of elliptic curves over finite fields.
- **Key Exchange:** The process of securely exchanging cryptographic keys between parties.

Chapter 2

LITERATURE REVIEW

2.1 Mathematical Principles of Public Key Cryptography

Public Key Cryptography is built upon several fundamental mathematical principles:

- **Number Theory:** PKC algorithms rely on prime factorization and discrete logarithms.
- **Modular Arithmetic:** Modular arithmetic is used in PKC algorithms to perform operations within finite sets of integers.
- **Group Theory:** Group theory concepts like cyclic groups define cryptographic primitives.
- **Complexity Theory:** The security of PKC relies on computationally hard problems such as factoring large integers or solving discrete logarithm problems.

2.2 RSA Algorithm: Example of Public Key Cryptography

2.2.1 Key Generation

In RSA, each user generates a pair of keys: a public key and a private key. The public key consists of two components: the modulus n and the public exponent e. The private key consists of n and the private exponent d. The security of RSA is based on the difficulty of factoring the modulus n into its prime factors.

2.2.2 Encryption and Decryption

- To encrypt a message m, the sender uses the recipient's public key (n, e) to compute $c = m^e \mod n$.
- To decrypt the ciphertext c, the recipient uses their private key (n, d) to compute $m = c^d \mod n$.

2.2.3 Example

Encrypting $M = 123$:

$$C = 123^{17} \mod 3233 = 855$$

Decrypting $C = 855$:

$$M = 855^{2753} \mod 3233 = 123$$

2.3 Elliptic Curve Cryptography (ECC)

2.3.1 Elliptic Curves

ECC uses elliptic curves defined by the equation $y^2 = x^3 + ax + b$ over finite fields. It offers equivalent security to RSA but with smaller key sizes, making it suitable for constrained environments such as mobile devices and IoT.

2.3.2 Key Generation

- Select an elliptic curve E defined over a finite field F_p.
- Choose a base point G on the curve E with a large prime order n.
- Select a random integer d as the private key such that $1 < d < n - 1$.
- Compute the public key $Q = dG$, where G is the base point.

2.3.3 Encryption and Decryption

- **Encryption:** To encrypt a message M (represented as a point on the curve), the sender computes the ciphertext pair (C_1, C_2), where $C_1 = kG$ and $C_2 = M + kQ$.

- **Decryption:** The recipient recovers the message M by computing $M = C_2 - dC_1$ using their private key d.

2.4 Pairing-Based Cryptography

Definition

Pairing is a bilinear map defined over elliptic curve subgroups. Let G_1, G_2, and G_T be groups of the same prime order q. A bilinear map $e : G_1 \times G_2 \to G_T$ satisfies the following properties:

- **Bilinearity:** For all $(S, T) \in G_1 \times G_2$ and for all $a, b \in \mathbb{Z}$, we have $e(aS, bT) = e(S, T)^{ab}$.
- **Non-degeneracy:** $e(S, T) = 1$ for all $T \in G_2$ if and only if $S = 1$.
- **Computability:** For all $(S, T) \in G_1 \times G_2$, $e(S, T)$ is efficiently computable.

Consequences of Pairings

Pairings have important consequences on the hardness of certain variants of the Diffie-Hellman problem. For instance, symmetric pairings lead to a strict separation between the intractability of the computational Diffie-Hellman problem and the hardness of the corresponding decision problem.

Chapter 3

Method of Study

3.1 RSA Cryptosystem

The RSA cryptosystem is based on the difficulty of factoring large prime numbers. The steps for key generation, encryption, and decryption are described as follows:

- Choose two distinct prime numbers p and q.
- Compute $N = pq$ and the totient $\phi(N) = (p-1)(q-1)$.
- Choose an integer e such that $1 < e < \phi(N)$ and e is coprime with $\phi(N)$.
- Compute d as the modular multiplicative inverse of $e \mod \phi(N)$.

The public key is (e, N), and the private key is (d, N).

3.2 Elliptic Curve Cryptography

ECC is based on the algebraic structure of elliptic curves over finite fields. The security of ECC relies on the difficulty of the elliptic curve discrete logarithm problem.

3.3 Mathematical Foundation of RSA

The RSA algorithm relies on the difficulty of factoring large composite numbers. The key steps are outlined below.

Key Generation

Given two large prime numbers p and q, we compute:

$$N = p \times q$$

where N is the modulus. Next, calculate Euler's totient function:

$$\phi(N) = (p-1) \times (q-1)$$

Choose an integer e such that:

$$1 < e < \phi(N) \quad \text{and} \quad \gcd(e, \phi(N)) = 1$$

The public key is (N, e). To generate the private key, compute d as the modular inverse of e modulo $\phi(N)$:

$$d \times e \equiv 1 \mod \phi(N)$$

The private key is (N, d).

Encryption

Given a plaintext message M, the ciphertext C is computed using the public key (N, e):

$$C = M^e \mod N$$

Decryption

To decrypt the ciphertext C, the recipient uses their private key (N, d) to recover the original message M:

$$M = C^d \mod N$$

Example

Let $p = 61$ and $q = 53$. First, compute:

$$N = 61 \times 53 = 3233$$

Next, calculate:

$$\phi(N) = (61-1) \times (53-1) = 3120$$

Choose $e = 17$, which is coprime to $\phi(N)$. Now, compute d, the modular inverse of 17 mod 3120, which gives $d = 2753$.

For encryption, given $M = 123$, calculate:

$$C = 123^{17} \mod 3233 = 855$$

To decrypt, calculate:

$$M = 855^{2753} \mod 3233 = 123$$

Thus, the encryption and decryption process works as expected.

3.4 Elliptic Curve Cryptography (ECC)

ECC is based on the algebraic structure of elliptic curves over finite fields. The security of ECC relies on the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP).

Elliptic Curve Definition

An elliptic curve E over a finite field F_p is defined by the equation:

$$y^2 = x^3 + ax + b \mod p$$

where $4a^3 + 27b^2 \neq 0 \mod p$, ensuring the curve has no singularities.

Key Generation

- Select a finite field F_p and an elliptic curve E defined over F_p.
- Choose a base point G on the curve E with large prime order n.
- Select a random integer d (the private key) such that $1 < d < n - 1$.
- Compute the public key $Q = d \times G$, where G is the base point on the curve.

Encryption

To encrypt a message M (represented as a point on the curve), the sender chooses a random integer k and computes the ciphertext pair (C_1, C_2):

$$C_1 = k \times G$$

$$C_2 = M + k \times Q$$

Decryption

To decrypt the ciphertext (C_1, C_2), the recipient uses their private key d to recover the original message:

$$M = C_2 - d \times C_1$$

Example

Consider the elliptic curve $y^2 = x^3 + x + 1$ over F_{89}, and let the base point be $G = (2, 22)$. Suppose the private key is $d = 10$. The public key is:

3.5 Pairing-Based Cryptography

Pairing-based cryptography relies on bilinear maps, enabling applications like identity-based encryption.

Mathematical Definition of Pairing

Let G_1 and G_2 be cyclic groups of prime order q, and let G_T be a multiplicative group of the same order. A pairing is a bilinear map:

$$e : G_1 \times G_2 \to G_T$$

satisfying the following properties:

- **Bilinearity:** For all $P \in G_1$, $Q \in G_2$, and $a, b \in \mathbb{Z}_q$:

$$e(aP, bQ) = e(P, Q)^{ab}$$

- **Non-degeneracy:** $e(P, Q) = 1$ if and only if $P = 1$ or $Q = 1$.

- **Computability:** The pairing $e(P, Q)$ can be computed efficiently.

Example of Pairing

Let $P \in G_1$ and $Q \in G_2$. The pairing $e(P, Q)$ can be computed using Miller's algorithm or the Tate pairing, depending on the elliptic curve used.

Chapter 4

IMPLEMENTATION

4.1 Introduction

This chapter focuses on the practical implementation of the cryptographic algorithms discussed in the previous chapters, particularly RSA and Elliptic Curve Cryptography (ECC). The implementations will be demonstrated using Python, leveraging libraries such as `pycryptodome` for RSA and `ecdsa` for ECC. The source code for key generation, encryption, and decryption will be provided, along with explanations of each step.

4.2 RSA Algorithm Implementation

4.2.0.1 RSA Key Generation

In RSA, key generation is the process of creating a public-private key pair. The public key is used for encryption, while the private key is used for decryption. The security of RSA lies in the difficulty of factoring large composite numbers. Below is the Python implementation for generating RSA keys:

[language=Python] from Crypto.PublicKey import RSA

Generate RSA key pair def generate$_r sa_k eys() : key = RSA.generate(2048) Generate 2048-$ $bitRSAkeyprivate_k ey = key.export_k ey() public_k ey = key.publickey().export_k ey()$

with open('private$_k ey.pem','wb') aspriv_f ile : priv_f ile.write(private_k ey)$

with open('public$_k ey.pem','wb') aspub_f ile : pub_f ile.write(public_k ey)$

print(”RSA Keys Generated and Saved.”)

4.2.1 RSA Encryption and Decryption

The following code demonstrates how to encrypt and decrypt a message using the RSA keys. The message is first encrypted using the recipient's public key and decrypted using the private key.

[language=Python] from Crypto.Cipher import PKCS1$_{O}AEPfromCrypto.PublicKeyimportRSAf$

RSA Encryption def rsa$_{e}ncrypt(message, public_{k}ey_{p}ath =' public_{k}ey.pem') : withopen(public_{k}ey_{p}ath$
$public_{k}ey = RSA.import_{k}ey(pub_{f}ile.read())cipher_{r}sa = PKCS1_{O}AEP.new(public_{k}ey)encrypted_{m}essa$
$cipher_{r}sa.encrypt(message.encode('utf-8'))returnencrypted_{m}essage$

RSA Decryption def rsa$_{d}ecrypt(encrypted_{m}essage, private_{k}ey_{p}ath =' private_{k}ey.pem') :$
$withopen(private_{k}ey_{p}ath,' rb')aspriv_{f}ile : private_{k}ey = RSA.import_{k}ey(priv_{f}ile.read())cipher_{r}sa =$
$PKCS1_{O}AEP.new(private_{k}ey)decrypted_{m}essage = cipher_{r}sa.decrypt(encrypted_{m}essage).decode('utf$
$8')returndecrypted_{m}essage$

Example usage message = "Public Key Cryptography is secure." encrypted$_{m}sg =$
$rsa_{e}ncrypt(message)print(f"Encrypted : encrypted_{m}sg")decrypted_{m}sg = rsa_{d}ecrypt(encrypted_{m}sg)p$
$decrypted_{m}sg")$

4.2.2 Elliptic Curve Cryptography (ECC) Implementation

Elliptic Curve Cryptography provides the same level of security as RSA but with much smaller key sizes, making it efficient for constrained environments. Below is an example implementation of ECC using the `ecdsa` library.

4.2.3 ECC Key Generation

ECC key generation involves choosing a private key (a random integer) and computing the corresponding public key, which is a point on the elliptic curve.

[language=Python] from ecdsa import SigningKey, SECP256k1

Generate ECC key pair def generate$_{e}cc_{k}eys() : private_{k}ey = SigningKey.generate(curve =$
$SECP256k1)public_{k}ey = private_{k}ey.get_{v}erifying_{k}ey()$

with open('ecc$_{p}rivate_{k}ey.pem',' wb')aspriv_{f}ile : priv_{f}ile.write(private_{k}ey.to_{p}em())$

with open('ecc$_{p}ublic_{k}ey.pem',' wb')aspub_{f}ile : pub_{f}ile.write(public_{k}ey.to_{p}em())$

print("ECC Keys Generated and Saved.")

4.2.4 ECC Encryption and Decryption

While ECC is primarily used for digital signatures and key exchange, we can simulate encryption and decryption by signing and verifying a message.

from ecdsa import SigningKey, VerifyingKey, BadSignatureError

4.3 Kannan-Type Fixed Point Theorem in b-Metric Spaces

We present a generalization of the Kannan fixed point theorem to b-metric spaces. The classical Kannan fixed point theorem provides conditions for the existence of a unique fixed point for mappings that satisfy a specific contraction condition. We extend this result to b-metric spaces, which are more general than metric spaces and allow for a relaxed triangle inequality.

Kannan-Type Fixed Point Theorem

Let (X, d) be a complete b-metric space with constant $s \geq 1$, and let $T : X \to X$ be a mapping such that for all $x, y \in X$, the following condition holds:

$$d(Tx, Ty) \leq \alpha[d(x, Tx) + d(y, Ty)]$$

for some constant $\alpha \in [0, \frac{1}{2})$. Then T has a unique fixed point, i.e., there exists a unique $x^* \in X$ such that $T(x^*) = x^*$.

Proof

1. **Constructing the Iterative Sequence** Choose an arbitrary point $x_0 \in X$ and define a sequence $\{x_n\}$ by setting $x_{n+1} = T(x_n)$ for all $n \geq 0$. We will show that $\{x_n\}$ converges to a fixed point of T.

2. **Establishing a Distance Estimate** Using the given contraction condition, we have:

$$d(x_{n+1}, x_{n+2}) = d(Tx_n, Tx_{n+1}) \leq \alpha[d(x_n, Tx_n) + d(x_{n+1}, Tx_{n+1})].$$

Since $\alpha < \frac{1}{2}$, we can derive that the sequence $\{x_n\}$ is contractive.

3. **Proving the Sequence is Cauchy** We now show that $\{x_n\}$ is a Cauchy sequence. For any $m > n$, apply the relaxed triangle inequality to obtain:

$$d(x_m, x_n) \leq s[d(x_m, x_{m-1}) + d(x_{m-1}, x_{m-2}) + \cdots + d(x_{n+1}, x_n)].$$

Using the contractive condition and summing the terms, we find that the series converges, implying that $\{x_n\}$ is a Cauchy sequence.

4. **Completeness and Convergence** Since (X, d) is a complete b-metric space, the Cauchy sequence $\{x_n\}$ converges to some $x^* \in X$.

5. **Verifying the Fixed Point** By the continuity of T, taking the limit as $n \to \infty$ in $x_{n+1} = T(x_n)$ gives $x^* = T(x^*)$, so x^* is a fixed point of T.

6. **Uniqueness of the Fixed Point** Suppose $y^* \in X$ is another fixed point such that $T(y^*) = y^*$. Using the contractive condition for x^* and y^*, we get:

$$d(x^*, y^*) = d(Tx^*, Ty^*) \leq \alpha[d(x^*, Tx^*) + d(y^*, Ty^*)] = \alpha[d(x^*, x^*) + d(y^*, y^*)] = 0,$$

which implies $x^* = y^*$. Thus, the fixed point is unique.

Chapter 5

SUMMARY

5.1 Summary

This project explores the applications of cryptographic techniques, specifically RSA and Elliptic Curve Cryptography (ECC), in secure communication. The project examines the theoretical foundations of RSA and ECC and investigates their real-world applications, highlighting RSA's role in securing web traffic through SSL/TLS, authenticating sender identities via digital signatures (PGP), and validating software updates. Additionally, the project discusses ECC's efficiency benefits in mobile devices, IoT devices, smart cards, and cryptocurrency (Bitcoin, blockchain). The project demonstrates the significance of cryptographic techniques in ensuring secure communication and provides insights into their practical applications. Overall, the project showcases the importance of RSA and ECC in maintaining secure communication in various industries and technologies.

5.2 Conclusion

This study has provided an in-depth exploration of RSA, Elliptic Curve Cryptography (ECC), and Pairing-Based Cryptography. Each method was explored mathematically with examples provided to illustrate their applications in cryptography. These methods play critical roles in ensuring the security and privacy of digital communications, and understanding their mathematical foundations is essential for implementing secure cryptographic systems., making it applicable to a broader class of problems.

References

Diffie, W., & Hellman, M. (1976). *New Directions in Cryptography.* IEEE Transactions on Information Theory, 22(6), 644-654.

Kannan, R. (1969). *Some Results on Fixed Points.* Bulletin of the Calcutta Mathematical Society, 61(1), 71-76.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, 21(2), 120-126.