Project Report

On

USE OF PRIME NUMBERS IN CRYPTOGRAPHY

Submitted

in partial fulfilment of the requirements for the degree of

BACHELOR OF SCIENCE

in

MATHEMATICS

by

AMRUTHA SUNIL (AB23BMAT005)

Under the Supervision of

MRS. DONNA PINHEIRO

DEPARTMENT OF MATHEMATICS

ST. TERESA'S COLLEGE (AUTONOMOUS)

ERNAKULAM, KOCHI - 682011

APRIL 2026

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

CERTIFICATE

This is to certify that the dissertation entitled, **USE OF PRIME NUMBERS IN CRYPTOGRAPHY** is a bonafide record of the work done by Ms. **AMRUTHA SUNIL** under my guidance as partial fulfillment of the award of the degree of **Bachelor of Science in Mathematics** at St. Teresa's College (Autonomous), Ernakulam affiliated to Mahatma Gandhi University, Kottayam. No part of this work has been submitted for any other degree elsewhere.

Date: 14/01/2026
Place: Ernakulam

Mrs.Donna Pinheiro
Assistant Professor,
Department of Mathematics,
St. Teresa's College(Autonomous),
Ernakulam.

ST. TERESA'S COLLEGE (AUTONOMOUS) DEPARTMENT OF MATHEMATICS AND STATISTICS ERNAKULAM

Dr. Elizabeth Reshma M.T
Assistant Professor and Head ,
Department of Mathematics,
St. Teresa's College(Autonomous),
Ernakulam.

External Examiners

1:..... 21/01/26

2: .....

DECLARATION

I hereby declare that the work presented in this project is based on the original work done by me under the guidance of Mrs.Donna Pinheiro, Assistant Professor, Department of Mathematics, St. Teresa's College(Autonomous), Ernakulam and has not been included in any other project submitted previously for the award of any degree.

Place: Ernakulam.

Date: 14/01/2026

AMRUTHA SUNIL (AB23BMAT005)

ACKNOWLEDGEMENT

We would like to express our profound gratitude towards Mrs. Donna Pinheiro, our research guide, for her unwavering encouragement, wise counsel and strong support during the course of this project.We truly appreciate the opportunity to learn from her instructions, expertise and patience that helped us complete this project. Additionally, we would like to express our gratitude to the entire staff of Department of Mathematics and Statistics,St. Teresa's College, Ernakulam for fostering a wonderful academic environment and providing us all the resources required for our study.

Place: Ernakulam.

Date:

AMRUTHA SUNIL (AB23BMAT005)

Contents

INTRODUCTION

During the last several decades, the internet and technology have grown explosively. Data security ensures that our data is only accessible to the sender and the receiver and prevents any modification or alteration of the data. For this, different types of algorithms and methods have been developed. Cryptography is derived from a Greek word. “Crypto” means “hidden or secret” and “graphy” means “writing”. Cryptography can be defined as techniques that cipher data, depending on specific algorithms that make the data unreadable to the human eye unless decrypted by algorithms that are predefined by the sender [1].

Cryptography

The information that we are transforming into secret code is called the “plain text”. The process of transforming is called “Encryption” and the encrypted text is called “ciphertext”. The process of transforming the ciphertext into its original form is called “Decryption”. To encrypt and decrypt, it requires a “key”, which is known only to the sender and the receiver. A “cryptographic algorithm (cipher)” is a mathematical formula or a set of rules used to encrypt or decrypt data.

There are two types of cryptographic algorithms such as Symmetric key cryptography (secret key cryptography) and Asymmetric key cryptography (public key cryptography). In Symmetric key cryptography, only one key is used to encrypt and decrypt the message by the sender and the receiver. It can be easier to make but less secure. Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Blowfish are examples of Symmetric key cryptography.

In Asymmetric key cryptography, two keys are used: a public key and a private

key. For encryption, a public key is used, which is known to the public and for decryption, a private key is used, which is known to the user. Rivest-Shamir-Adleman (RSA), Elliptic curve cryptography (ECC) are examples of Asymmetric key cryptography.

There are few historical algorithms. The Caesar Cipher, created by Julius Caesar, is a simple substitution cipher that shifts each letter in the alphabet by a fixed number, typically three. While easy to use, it is also easy to break. The Simple Substitution Cipher replaces each letter with another letter based on a secret key, with the same key used for both encryption and decryption. Unlike substitution ciphers, Transposition Ciphers rearrange the letters of the message according to a key. One common type, the columnar transposition cipher, writes the plaintext into rows and then reads the columns in an order determined by the key. There are two variants: complete columnar transposition, which fills empty spaces to equalize column length, and incomplete columnar transposition, which leaves columns uneven, making decryption more difficult without the key.

Prime numbers

Prime numbers are often referred to as the "basic building blocks" or atoms of natural numbers due to their fundamental role in number theory. In today's era of digital technology, they hold significant importance for computer scientists and programmers, particularly in solving practical problems involving data security and encryption.

From a historical perspective, the exact origins of prime number discovery are unclear. However, some of the earliest known evidence comes from ancient Egyptian Papyrus, over 3,500 years old. The ancient Greeks were the first to include prime numbers in formal education, and since then, many renowned mathematicians have made significant contributions to the study of primes.

In 1640, Pierre de Fermat proposed (without proof) *Fermat's Little Theorem*, which was later demonstrated by Leibniz and Euler. He also investigated the primality of *Fermat numbers* of the form $2^{2^n} + 1$. Meanwhile, Marin Mersenne

focused on *Mersenne primes*, which are primes of the form $2^p - 1$, where p is itself a prime number.

Goldbach introduced *Goldbach's Conjecture*, which suggests that every even number can be represented as the sum of two primes [4]. Similarly, Euler, Gauss, Chebyshev, and Riemann made substantial contributions, especially concerning the distribution of prime numbers. Nonetheless, the study of prime numbers remains both fascinating and complex, with numerous unresolved questions existing regarding primes.

Prime numbers also play a crucial role in modern cryptography, especially in asymmetric encryption algorithms like RSA (Rivest–Shamir–Adleman). The RSA algorithm relies on the mathematical properties of primes to generate secure public and private keys, making prime numbers essential to digital security systems around the world.

Chapter 1

CRYPTOGRAPHY AND PRIME NUMBERS

1.1 Types of cryptography

As we know, Cryptography technique is used to convert plaintext into ciphertext. This technique is done by a cryptographic key, which is basically a string of characters, which is used to encrypt and decrypt the data. According to the concepts of the keys used in this technique, the cryptography is classified mainly to three types:

- Symmetric key cryptography
- Asymmetric key cryptography
- Cryptographic hash functions

Hybrid encryption is another type, which is a combination of symmetric and asymmetric key cryptography.

1.2 Symmetric key cryptography

It is the most widely used cryptographic technique, also called Private key cryptography. It is used for encryption and decryption of input data using the same secret key. The sender and receiver share a common secret key to maintain the confidentiality of the data. This technique is relatively fast and secure. The plaintexts are transformed into ciphertext using a particular key by the sender and the same key is used to decrypt the ciphertext to plaintext by the receiver. The major drawback of this technique is the sharing of keys.

In encryption, a process is done to the plaintext (say adding key characters to the plaintext) to obtain ciphertext. Then during the decryption, the reverse process is done (subtracting key characters from the ciphertext) to obtain the plaintext back.

The symmetric ciphers are divided into two types:

- Stream cipher - It takes the input as single characters.
- Block cipher - It takes the input as a whole block of text.

Example for symmetric ciphers:

- Caesar cipher
- Playfair cipher
- Vignere cipher etc.

Example for symmetric algorithms:

- AES (Advanced Encryption Standard)
- DES (Data Encryption Standard)
- TLS/SSL Protocol

1.3 Asymmetric key cryptography

It is also known as Public key cryptography, as it requires one pair of keys: Public key and Private key, to encrypt and decrypt data. The public key is publicly distributed, anyone can use that to encrypt messages, but only the recipient who holds the corresponding private key can decrypt those messages. (For understanding it works like a mailbox. Anyone can send a message to the owner of a mailbox but only the owner has the key to open and read it.)

Though there is a mathematical connection between the pair of keys, the public key cannot generate the private key. These asymmetric algorithms are considered complex, as it involves heavy mathematical computations, which is hard and considered one way. These are also called trapdoor functions. Hence it will be slower compared to symmetric cryptography but relatively more secure because of its complexity in the process. This cryptographic system addresses two major challenges faced in traditional (symmetric) cryptography: key distribution and digital signatures. Asymmetric algorithms use one key for encrypting data and another, related key for decrypting it. These algorithms possess an important feature:

- It's impossible to figure out the decryption key just by knowing the encryption key and the cryptographic algorithm.
- Either of the two keys can be used for encryption, while the other is used for decryption.

Example for asymmetric algorithms:

- RSA (Rivest Shamir Adleman)
- Diffie Hellman key exchange
- ECC (Elliptical Curve Cryptography)

1.4 Cryptographic Hash Functions

This type of cryptography does not use a key. Hash functions compress a string of arbitrary input (message) to a string of fixed length, called hash, hash value, message digest and the process is called hashing. It is also referred to as a "fingerprint" on the plaintext message. The hashes are unique to each plaintext and the hash functions should satisfy some additional conditions to apply in cryptographic applications.

Collision resistance - it should be difficult to find two different messages having the same hash.

Preimage resistance - given the hash value of some unknown message, it should be difficult to find any message hashing to that value.

Second preimage resistance - given some message, it should be difficult to find different messages having the same hash.

Hash functions help to ensure data integrity between communicating parties. If the hash produces the same output, it indicates that the information has not been altered, compromised or damaged.

Example for hash algorithms:

- MAC
- MD5
- SHA-1
- SHA-2

1.5 Hybrid Encryption

In addition to three major cryptography types, hybrid encryption is another type used to secure data and communication. The actual data (plaintext) is encrypted

using the symmetric key. This process is fast and suitable for encrypting large amounts of data.

The symmetric key, used to encrypt the data is then encrypted using the recipient's public key. This ensures that only the recipient, who possesses the corresponding private key, can decrypt the symmetric key.

The encrypted data and encrypted symmetric key are sent to the recipient. The recipient uses their private key to decrypt the symmetric key and then uses the decrypted symmetric key to decrypt the actual data. This approach leverages the speed of symmetric encryption and the security of asymmetric encryption.

1.6 Prime Numbers - The Basic Building Blocks

The exploration of prime numbers and their characteristics has constantly captivated mathematicians. Primes can be viewed as the "fundamental components," the atoms, of the natural numbers. They hold a crucial position in the field of number theory. In addition, prime numbers have significant importance in today's computing and digital technology, aiding computer programmers and scientists in solving real-world issues. In cryptographic encryption systems, prime numbers are essential for security measures where prime factorization is critical.

A prime number, or simply a prime, is a natural number greater than 1 that has no positive divisors other than 1 and itself. Symbolically, a number p is said to be prime if:

(i) $p > 1$

(ii) p has no positive divisors except 1 and p

A prime number is an integer greater than 1, having only two factors: 1 and itself.

The **Fundamental Theorem of Arithmetic**, also called the *Unique Factorization Theorem* or *Prime Factorization Theorem*, states that every integer greater

than 1 can be represented uniquely as a product of prime numbers, up to the order of the factors [4].

The characteristic of a number being prime is referred to as **primality**. Additionally, two numbers, a and b, are considered **relatively prime** if they do not share any prime factors. The succession of prime numbers is unpredictable, and their sequence is still not fully understood.

1.7 Types of Prime Numbers

There are said to be 76 types of prime numbers. Some of them include:

1. **Mersenne Prime**: A prime number derived by subtracting one from a power of two. The simplest method to verify this is to add one to the prime number, and the result should take the form of 2^n In algebraic terms, a Mersenne prime is expressed as

   $$M_p = 2^p - 1$$

   [5].

2. **Twin Prime**: A twin prime is a prime that is either 2 less or 2 more than another prime number. Every twin pair except (3,5) is of the form $(6n - 1, 6n + 1)$, where n is an integer [5].

3. **Semi-Prime**: A natural number that has factors consisting of only one or two identical or distinct prime numbers is referred to as a Semi-Prime. It can also be described as the product of two prime numbers, and if both numbers are identical, then the Semi-Prime number is the square of a prime number [5].

4. **Co-Prime**: Two numbers are called co-prime if their greatest common factor is 1.
   Example: (3, 7)

5. **Fibonacci Primes**: Fibonacci numbers which are prime numbers are called Fibonacci primes.
   Examples: 2, 3, 5, 13

Applications and Importance

Prime numbers play a significant role in numerous applications in mathematics and beyond. Nowadays, we frequently encounter prime numbers, often without realizing it.

For mathematicians, primes are crucial because they serve as the fundamental building blocks of multiplication. Adequate knowledge of primes enables the resolution of many complex multiplication-related problems.

In the broader context, the primary applications of prime numbers pertain to computers. They are utilized in security systems such as online transactions and communications. Consequently, the investigation into prime generation has been conducted, along with the factorization of composite numbers through various methods, including Traditional, Fermat's Theorem, Pollard Rho, and others.

1.8 Prime Numbers In Cryptography

Prime numbers have been recognized as important building blocks in mathematics, but their crucial role in modern cryptography only became clear with the rise of public-key systems in the late twentieth century. Their unique arithmetic properties provide a base for secure digital infrastructures. In public-key cryptography, the security of various protocols relies on computational asymmetry created by large prime numbers. Simple operations, such as multiplication, are easy to calculate, while reversing these operations, like factoring the product of very large primes, is extremely challenging. Reference materials like the *Handbook of Applied Cryptography* explain that prime-based systems such as RSA, Diffie-Hellman, and ElGamal depend on this asymmetry to ensure confidentiality and authenticity in digital communications. Hoffstein, Pipher, and Silverman highlight that the behavior of primes in modular arithmetic supports cryptographic methods that resist attacks based on proven theorems and assumed difficulties. Therefore, primes are not just mathematical oddities; they are essential parts that enable secure data exchange, authentication, and digital trust today.

1.8.1 Mathematical Foundations

The usefulness of prime numbers in cryptography comes from deep number-theory principles that explain how integers act in modular arithmetic. The Fundamental Theorem of Arithmetic asserts that every integer greater than one factors uniquely into prime products. This theorem is more than a classic result; it lays the groundwork for algorithms that manipulate integers in computational contexts. Prime numbers allow modular arithmetic to show predictable patterns, especially within finite fields like $\mathbb{Z}_p$, where p is prime. As outlined in *Introduction to Mathematical Cryptography*, arithmetic in these fields is straightforward because every non-zero element has a multiplicative inverse. This feature allows cryptographic algorithms to use modular division and exponentiation securely. Furthermore, primes enable the formation of cyclic groups, which are vital for discrete logarithm-based methods such as Diffie-Hellman. Cryptographic operations heavily depend on these groups' properties because their mathematical simplicity meets computational difficulty when group orders are large. This blend of structure and unpredictability explains why prime numbers are crucial in modern cryptographic systems.

Integer Factorization

The integer factorization problem is central to several prime-based cryptosystems, most notably RSA. In computational number theory, multiplying two large primes—often hundreds or thousands of bits long—is easy for modern computers, as noted in Crandall and Pomerance's *Prime Numbers: A Computational Perspective*. However, reversing this operation by finding the prime factors of a large composite number $N = pq$ is thought to require more than polynomial time for regular machines. This difficulty supports the essential belief behind RSA's security. While there is a general-purpose algorithm for integer factorization, such as the General Number Field Sieve, executing it remains impractical for well-designed RSA moduli. Stallings emphasizes that the difference between the ease of multiplication and the challenge of factorization allows RSA to secure sensitive communications over public networks. Therefore, cryptographic systems select primes with certain properties—such as being large, random, and co-prime

to certain parameters—to ensure that no known attack can exploit weaknesses in their structure.

Prime Factorization

Prime factorization is a specific type of integer factorization that further underscores the significance of the Fundamental Theorem of Arithmetic. In cryptography, composite numbers used as moduli, like those in RSA, must be carefully designed so that their prime factors stay hidden from potential attackers. The *Handbook of Applied Cryptography* explains that cryptographic primes are generated using probabilistic methods that test for primality efficiently, ensuring the values chosen meet strict security standards. Not all primes are appropriate; for instance, primes used for Diffie-Hellman are often required to be "safe primes," where $p = 2q + 1$ and q is also prime. Such primes help guard against small-subgroup attacks and enhance the security of discrete logarithm calculations. As cryptographers create systems that rely on the challenges of reversing prime multiplication, they must also consider possible side-channel attacks and algebraic vulnerabilities, leading to carefully developed prime generation processes detailed in cryptographic literature.

Computational Complexity of Factorization

The complexity of factorization is a changing topic influenced by algorithm advancements and hardware improvements. Although factorization fits within NP, it has not been shown to be NP-complete or P, leaving its exact theoretical status uncertain. Works like *Introduction to Modern Cryptography* explain how sub-exponential algorithms, such as the Quadratic Sieve and the General Number Field Sieve, have significantly improved factorization efficiency, yet these methods still become impractical for extremely large integers. The security of prime-based cryptosystems relies not only on current factorization algorithms but also on future advances. The introduction of Shor's quantum algorithm complicates the long-term security picture, as quantum computing could threaten the assumptions of hardness that factorization-based systems depend on. This situation highlights the

careful balance between mathematical complexity and cryptographic practicality, as systems need to keep evolving to ensure their security.

Fermat's Little Theorem

Fermat's Little Theorem is one of the foundational results used to build many number-theoretic cryptosystems. While the theorem itself is straightforward, its applications are broad. Cryptographic texts often refer to this theorem concerning primality testing, modular exponentiation, digital signatures, and the creation of pseudorandom number generators. For example, the theorem supports algorithms such as the Fermat primality test and aids the efficiency of computing modular exponentiation, a key operation in RSA and Diffie-Hellman. The theorem also verifies the correctness of public-key operations; raising a message to a public exponent modulo a prime or composite has predictable behavior due to the cyclic nature enforced by Fermat's result.

Euler's Theorem

Euler's theorem expands the reach of Fermat's result by applying it to any modulus n, with $\phi(n)$ representing its Euler totient. Cryptography relies greatly on this theorem, especially in RSA, where the correctness of encryption and decryption processes hinges on the relationship $M^{ed} \equiv M \pmod{n}$, with $ed \equiv 1 \pmod{\phi(n)}$. Standard cryptographic references highlight the theorem as a mathematical guarantee for invertible modular exponentiation. Without Euler's theorem, it would not be possible to create a public-key cryptosystem where encryption and decryption are inverse actions computed with different keys. Therefore, Euler's theorem serves as a conceptual link between prime numbers, modular arithmetic, and the practical realization of secure digital communication.

1.8.2 Role of Prime Numbers in Cryptographic Systems

Prime numbers are crucial in many cryptographic systems beyond the well-known RSA and Diffie-Hellman. For instance, in the Digital Signature Algorithm (DSA),

primes define the cyclic groups that solve discrete logarithm problems, ensuring signatures cannot be forged without tackling complex issues. Cryptographically secure pseudorandom number generators often use primes in their recurrence relations because they create desirable distribution properties. Crandall and Pomerance describe further applications, such as elliptic curve cryptography (ECC), where, although primes appear indirectly in finite fields, their influence ensures elliptic curve operations remain secure. In ECC, primes govern the field structure and shape curve parameters, making the resulting groups resistant to known algorithms. Prime-based frameworks, therefore, touch every level of modern cryptographic systems, from key generation and exchange to signature verification and randomness generation.

Security Implications

The dependence on prime numbers in cryptography offers both strength and vulnerability. On one side, modern cryptographic security is built on well-understood mathematical assumptions related to prime-based difficulty problems. On the other hand, advancements in computing power, algorithm development, and quantum computing threaten existing systems. The arrival of Shor's algorithm shows that once quantum computers are sufficiently advanced, they could factor large integers and efficiently solve discrete logarithms, making RSA, DSA, and Diffie-Hellman insecure. Standard references stress the need to move towards post-quantum cryptography, avoiding weaknesses linked to prime factorization and discrete logarithms. Lattice- based, hash-based, and code-based cryptographies are alternatives that do not depend on primes, indicating a potential shift in cryptographic design philosophy. Nevertheless, current systems based on primes continue to be dominant due to their maturity, ease of implementation, and extensive analysis over decades.

Prime numbers form the mathematical foundation of classical public-key cryptography and remain vital for secure communications. Their properties enable systems to create cryptographic mechanisms that are easy to compute yet incredibly tough to break. While future changes, especially in quantum computing, may lessen their unique role, primes will continue to hold historical and conceptual

importance in understanding cryptographic design. Their ongoing relevance is supported by extensive literature and years of practical application, ensuring that prime-based methods remain a key component in cryptographic education and research.

Chapter 2

MODERN CRYPTOSYSTEM

Modern cryptosystems form the foundation of secure communication in today's digital world. With the rapid growth of internet technologies, cloud computing, digital payments, and connected devices, the need to protect data against unauthorized access, tampering, and identity theft has become more critical than ever. Cryptography has evolved significantly from simple classical techniques to advanced mathematical systems that can secure information even in highly adversarial environments. These modern cryptographic systems use well-defined mathematical problems, complex algorithms, and standardized protocols to ensure confidentiality, integrity, authentication, and non-repudiation.

Modern cryptosystems mainly include two categories: symmetric-key algorithms, which use the same key for both encryption and decryption, and asymmetric-key algorithms, which use mathematically related public and private keys. Asymmetric cryptosystems such as RSA, Diffie–Hellman, and Elliptic Curve Cryptography (ECC) play a central role in secure key exchange, digital signatures, and identity verification. At the same time, hybrid cryptosystems combine the advantages of both symmetric and asymmetric methods, allowing modern protocols like TLS, HTTPS, and secure messaging applications to deliver strong security with high performance.

Furthermore, the emergence of advanced computational threats and the anticipated impact of quantum computing have prompted continuous evolution in the

design of cryptographic systems. Although post-quantum cryptosystems address future threats, classical modern cryptosystems remain widely deployed and essential in practical security infrastructures. They protect billions of online transactions, secure software updates, enable authenticated communication, and form the backbone of modern cybersecurity systems.

2.1 RSA

Established in 1977 by three cryptographers—Ron Rivest, Adi Shamir, and Len Adleman—RSA is the most prevalent public-key encryption method. The algorithm relies on the complexity of factorizing large numbers and the challenge of obtaining factors from exceedingly large prime numbers. The key lengths for the RSA algorithm can be 1024, 2048, or 4096 bits. The considerable key size renders the algorithm virtually unbreakable. The primary services offered by RSA include:

Encryption: The two keys utilized in RSA (the public key and private key) can facilitate either the encryption or decryption of the data exchanged between the two communicating parties.

Key Exchange: RSA is utilized for the secure exchange of keys. Cryptographic systems generate asymmetric keys for use in DES/AES algorithms. The symmetric key is encrypted using the receiver's public key and then transmitted to the receiver, who recovers the symmetric key with their private key. The symmetric key is subsequently input into a symmetric algorithm.

Digital Signature: RSA is capable of generating digital signatures. The private key of the sender acts as input to the Digital Signature Algorithm (DSA) to transform the data or its hash value. The sender's public key is then employed to verify the message's authenticity.

2.2 Diffie-Hellman Key Exchange

The Diffie-Hellman (DH) protocol is not an encryption method but an algorithm that allows two communicating parties to compute and share a key, which can then be utilized in symmetric algorithms like AES. Diffie-Hellman is based on discrete logarithms. A significant limitation of DH is the susceptibility to man-in-the-middle attacks, where an attacker can position themselves between two legitimate hosts and successfully compute the shared key with each host. This attacker would then have the capability to read, alter, and resend messages without detection.

2.3 Elliptic Curve Cryptography

Without a doubt, RSA stands as a cornerstone of public key cryptography. Most products that have implemented asymmetric key encryption systems utilize RSA. Recently, standards like elliptic curve cryptography (ECC) have been gaining traction because they can deliver equivalent security to RSA with smaller keys. To achieve a high security level with RSA (as with certain other public encryption methods), the keys must be of substantial size. When using keys as large as 4096 bits, the processing overhead for applications utilizing RSA increases significantly, potentially leading to slower transactions—particularly for e-commerce sites that require prompt and secure transactions. ECC can attain the same level of security as RSA with considerably smaller key sizes.

2.4 Secure Hash Algorithm 3 (SHA-3)

SHA-3 is a hashing algorithm that was recently standardized. Some of the hash algorithms that are widely used, such as MD4, MD5, and SHA-0, have been shown to have significant weaknesses in cryptanalysis. SHA-1 has been considered theoretically insecure since 2005. In 2017, it was confirmed to be susceptible to collisions. A collision attack against SHA-1 was successfully conducted by Google and CWI Amsterdam using two distinct PDF files that generated the same SHA-1 hash, which led major web browser vendors to stop accepting SHA-

1 SSL certificates. SHA-2 and SHA-1 were developed around the same period, and their algorithms are interconnected. Following the vulnerabilities found in SHA-1, NIST initiated a competition for a new algorithm, SHA-3. By October 2008, around 64 algorithms had been submitted, and in December 2010, five were selected. The Keccak algorithm was chosen as SHA-3 in October 2012.

NIST's SHA-3 specifications include:

- Algorithms that can handle data of arbitrary length while providing four possible output lengths of 224, 256, 384, and 512 bits.

- SHA3-224 offers attack resistance equivalent to that of 3DES.

- SHA3-256 provides an attack resistance of 2^{112}, comparable to AES-128.

- SHA3-384 has an attack resistance of 2^{128}, akin to AES-192.

- SHA3-512 achieves an attack resistance of 2^{192}, similar to AES-256.

SHA-3 employs a structure known as sponge construction. The sponge function involves an absorption phase that takes an input message Xi and divides it into blocks of fixed size. Each block Yi undergoes processing and is then fed into the next iteration for 24 rounds until the fixed-size output is generated at the conclusion of the process. As with other cryptographic algorithms, every round consists of a series of substitution and permutation operations.

2.5 Post-Quantum Cryptography (PQC) Algorithms And Their Potential Applications

- **Lattice-Based Cryptography**

Lattice-based cryptography is a subdivision of cryptography that depends on the difficulty of lattice problems for its security. Lattices are geometric configurations created by points in multiple dimensions, and lattice-based cryptography takes advantage of the computational complexity associated with specific lattice problems to develop cryptographic techniques that resist attacks, including those from

quantum computers.

In lattice-based cryptography, the security of cryptographic methods is based on the presumption of the challenging nature of lattice problems, like the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). These challenges involve identifying the shortest or nearest lattice vector to a specific point in the lattice and are recognized as computationally intense, even for quantum machines.

Rooted in the computational complexity of lattice challenges, lattice-based cryptography delivers a flexible and quantum-resistant framework for cryptographic methods. Algorithms such as NTRUEncrypt, Kyber, and NewHope utilize lattice structures to provide secure communication, digital signatures, and key exchange techniques. Its durability against quantum attacks, along with applications ranging from secure communication channels to homomorphic encryption, highlights its importance in post-quantum security initiatives and future cryptographic norms. Ongoing research and standardization efforts are further reinforcing the position of lattice-based cryptography in protecting sensitive data and ensuring the reliability of digital transactions in the era of quantum computing.

- **Lithium-based cryptography**

1. Versatility: Lattice-based cryptography presents a broad array of cryptographic primitives, which include encryption methods, digital signatures, and key exchange techniques. This flexibility enables the development of secure cryptographic systems that can be customized for specific application needs.

2. Quantum Resistance: Lattice-based cryptography is fundamentally resistant to assaults from quantum computers. Unlike conventional cryptographic methods grounded in number theory, such as integer factorization and discrete logarithm, lattice-based methods do not depend on assumptions that are susceptible to quantum algorithms, like Shor's algorithm.

3. Security Assumptions: The security of lattice-based cryptography is built upon the difficulty of specific lattice challenges. While these challenges are generally considered computationally hard, ongoing research seeks to evaluate the

security of lattice-based methods and to create efficient algorithms for solving lattice-related issues.

- **Code-based Cryptography**

This method stands as a strong contender in the field of post-quantum security, utilizing the complex characteristics of error-correcting codes to enhance cryptographic techniques against quantum threats. In contrast to conventional cryptographic methods that are susceptible to quantum algorithms, code-based solutions such as the McEliece cryptosystem and the BIKE algorithm draw their strength from the difficulty of error-correcting codes, rendering them robust against quantum attacks. Although these systems often require greater computational resources than classical cryptography, their ability to withstand quantum threats makes them crucial for long-term security in a world increasingly influenced by quantum capabilities. Code-based cryptography is utilized in various fields, including secure communication, digital signing, and key exchange mechanisms, offering a solid defense against evolving quantum threats while maintaining the integrity and confidentiality of sensitive data in the digital era. As research and standardization efforts evolve, code-based cryptography is set to become a fundamental element in the future of secure communication and data protection in the face of quantum computing advancements.

- **Multivariate Polynomial Cryptography (MPC)**

This branch of post-quantum cryptography represents a powerful avenue by using the intricacies of solving systems of multivariate polynomial equations to ensure security in cryptographic applications. Unlike typical cryptographic methods, MPC algorithms such as Rainbow and HFE exploit the significant computational challenges associated with these equations, making them resilient to threats from both classical and quantum attackers. These algorithms provide strong security assurances, even with the progress in quantum computing. One of MPC's primary advantages is its adaptability, enabling its use in various cryptographic functions including digital signatures, encryption methods, and authentication protocols.

The inherent mathematical difficulties associated with MPC algorithms establish a strong security basis while allowing for effective real-world application. Additionally, MPC frameworks offer alternative cryptographic strategies that are suitable for particular scenarios where other post-quantum cryptographic (PQC) solutions may be less effective. In practical terms, Multivariate Polynomial Cryptography is employed to secure communication channels, digital signatures, and encryption protocols, providing a strong defense against quantum threats and ensuring the confidentiality and integrity of sensitive data. As developments and research in MPC move forward, it has the potential to greatly influence the field of post-quantum cryptography, delivering robust and adaptable solutions for securing digital communications and protecting critical infrastructure in the age of quantum computing.

- **Hash-Based Algorithm**

Within the domain of post-quantum cryptography (PQC), hash-based cryptography is a vital area of investigation and innovation, addressing the potential risks that quantum computing poses to conventional cryptographic systems. In contrast to classical computing, quantum computers utilize the principles of quantum mechanics to execute computations at speeds that far surpass those of classical computers. This advancement may leave many existing cryptographic algorithms, such as RSA and ECC, susceptible to attacks, necessitating alternative solutions like hash-based cryptography. In this field, one-way hash functions are fundamental. These functions accept an input and yield a fixed-size output, referred to as a hash value or digest. A key characteristic of one-way hash functions is their irreversibility: it is computationally impractical to ascertain the original input from a hash value. This feature ensures the security of hash-based cryptographic systems by complicating the ability of adversaries to extract sensitive information from hashed data.

One significant use of hash-based cryptography within the PQC framework is in digital signatures. Digital signatures are crucial for verifying the source and integrity of digital messages or documents. In schemes that utilize hash-based signatures, the signer creates a hash value from the message and subsequently

signs this hash value with their private key. Recipients can then confirm the signature using the signer's public key and compare it against the hash value of the message they independently compute.

Another important application of hash-based cryptography involves secure data structures, such as Merkle trees. Merkle trees are binary structures formed with hash values, where every leaf node corresponds to a data block and each non-leaf node represents the hash of its child nodes. These trees enable efficient and secure verification of large datasets, allowing parties to verify the integrity of specific data blocks without needing to inspect the entirety of the dataset. This feature makes Merkle trees essential for ensuring data integrity in distributed systems, blockchains, and peer-to-peer networks.

In terms of PQC, hash-based cryptography offers several benefits. Firstly, hash functions tend to be straightforward and efficient, making them appropriate for environments with limited resources. Additionally, hash-based cryptographic systems typically rely on mathematical challenges that are not currently believed to be susceptible to quantum attacks, thus providing a solid basis for secure communication and data integrity in a world post-quantum computing.

Nevertheless, hash-based cryptography is not without its drawbacks and challenges. A significant issue is the scalability of hash-based methods for particular applications, especially in situations requiring a large number of signatures or verifications. Moreover, like any cryptographic system, the security of hash-based methods hinges on the underlying hash function's integrity. Therefore, continuous research and assessment of hash function designs and implementations are crucial to preserving their security against evolving threats. By harnessing the security attributes of hash functions, hash-based cryptographic schemes deliver strong solutions for digital signatures, data integrity verification, and secure communication in the post-quantum era. As research in this area continues to progress, hash-based cryptography is likely to play a pivotal role in influencing the future of secure digital communications and information exchange.

Chapter 3

MODERN DAY APPLICATIONS OF CRYPTOGRAPHY

Cryptography is crucial for modern digital security. It protects sensitive information in our connected world. Many current cryptographic systems rely on prime numbers. Prime numbers have important properties, like indivisibility, unpredictability, and difficulty in factorization, which make them perfect for secure digital systems. Today, data is frequently shared over untrusted public networks. Cryptographic systems that use large primes keep this information private, verified, and safe from unauthorized changes. Modern cryptography has numerous applications, including digital signatures, secure communication protocols, blockchain systems, password authentication, electronic banking, cloud computing, and large-scale VPNs. Each platform relies on the computational difficulty of prime-based math to ensure security. The following sections examine the role of prime numbers in cryptography, covering theoretical ideas, implementation models, operational processes, and practical applications that shape today's secure digital environments.

3.1 Digital Signatures

Digital signatures are among the most important uses of prime-number-based cryptography. They offer a way to verify the authenticity, integrity, and source of digital documents, messages, and transactions. A digital signature system uses public-key cryptography, where security depends on math problems related to prime numbers, like the difficulty of factoring large composite integers. RSA, one of the earliest and most commonly used digital signature algorithms, is based on the properties of large primes. In RSA-based digital signatures, two large primes are multiplied to create a modulus whose factorization is too difficult to reverse. This allows for secure identity verification. Hashing methods, like SHA-256, work with RSA to compress variable-length messages into fixed-size digests. This creates unique representations that change dramatically with even minor modifications to the original message. Since the digest is signed instead of the whole message, this process remains efficient while ensuring strong security. Digital signatures thus confirm the sender's identity, guarantee that the message hasn't been changed, and prevent the sender from denying they signed it.

Cryptographic Foundation

Digital signature schemes, such as RSA, are based on number theory involving modular arithmetic, Euler's totient function, and prime factorization. RSA starts with the selection of two extremely large prime numbers that are multiplied to create the modulus for encryption and signature generation. The totient of the modulus, derived from the prime factors, allows for the creation of public and private exponents that are linked mathematically but hard to derive from each other without knowing the primes. These values control the signature creation and verification process. Hashing methods in finite prime fields produce digests that minimize the chances of collisions and ensure that even minor changes to the message yield a completely different hash output. Because the hash is signed using the private exponent, only the legitimate key owner can create a valid signature. The verification process uses the public exponent to confirm the signed digest's authenticity, linking the signer's identity to the message through prime-based

arithmetic.

Digital Signature Process

The digital signature process consists of two main steps: signing and verification. During signing, the message is first processed through a secure hashing algorithm such as SHA-256, creating a condensed digest. This digest is then raised to the power of the private exponent modulo the prime-based modulus, producing a signature that uniquely represents both the message and the signer. In the verification stage, the receiver calculates the hash of the received message independently and raises the signature to the power of the public exponent modulo the same modulus. If the resulting value matches the independently computed digest, the signature is verified as authentic. Because only the private key owner can create a valid signature, this process guarantees authentication. Any changes to the message alter the digest, ensuring integrity. Finally, because the signature mathematically links the sender to the message, it prevents denial.

Industrial Applications

Digital signatures have become widespread in modern infrastructure, enabling secure interactions across many applications. In Public Key Infrastructure (PKI), certificate authorities use digital signatures to issue, verify, and revoke digital certificates that validate identities on the internet. Secure web browsing through HTTPS is possible because web servers and browsers verify SSL/TLS certificates signed using RSA or ECDSA. Software companies sign their code and updates to assure users that the software is authentic and unaltered. Email security protocols like S/MIME use digital signatures to authenticate senders and secure communications. Cryptocurrencies use digital signatures to authorize transactions, and blockchain networks depend heavily on signature verification for secure block validation. Throughout these implementations, prime numbers provide the foundation that ensures digital signatures remain secure and reliable.

Challenges

Despite their strengths, digital signature systems face some challenges. RSA needs large key sizes, which raise computational demands during signature generation and verification. While modern devices can handle these tasks, limited environments like embedded systems or IoT devices may struggle. Another major challenge is the threat from quantum computing. Algorithms like Shor's algorithm can factor large composites and solve discrete logarithms quickly, possibly making prime-based cryptographic schemes vulnerable in the future. Although quantum-resistant alternatives are in development, prime-based digital signatures are essential in current security architectures.

3.2 Secure Communication

Secure communication methods are vital for protecting the privacy and integrity of digital interactions. Whether in personal messaging, enterprise communication, or secure browsing, systems must ensure that unauthorized parties cannot intercept or alter transmitted information. Prime-number-based cryptographic algorithms like RSA and Diffie–Hellman enable secure communication by creating encryption keys and authenticating message sources through rigorous mathematical processes. These systems rely on public-key cryptography, with large primes forming the basis of modular arithmetic operations that stop attackers from decrypting or changing messages. Secure communication systems combine encryption, hashing, digital signatures, and certificate validation to build a layered defense against threats.

Communication Model

A complete communication model includes key generation, message encryption, transmission, decryption, and integrity validation. In RSA-based systems, the sender encrypts messages using the recipient's public key, ensuring that only the private key holder can decrypt them. Hash functions verify message integrity, while digital signatures authenticate the sender and prevent impersonation. Together, these processes create a secure channel where confidentiality, authenticity,

and integrity are maintained through complex mathematical operations involving primes.

Diffie–Hellman and Prime Numbers

Diffie–Hellman is one of the first secure key exchange protocols and is still widely used today. It uses a large prime modulus and a generator to create a shared secret between parties over an insecure channel. Because the discrete logarithm problem is hard to solve in prime fields of adequate size, attackers cannot derive the shared key even if they intercept all transmitted values. This protocol supports key establishment in VPNs, secure messaging platforms like WhatsApp and Signal, and encryption protocols like TLS. Its reliance on prime-based modular arithmetic ensures the strength of the shared secret key.

Implementation Features

High-level secure communication systems include additional protections like Optimal Asymmetric Encryption Padding (OAEP) to reduce vulnerabilities and padding attacks in RSA. Certificate validation ensures that communication endpoints are authenticated through trusted certificate authorities. Hash-based verification techniques guarantee that transmitted messages remain unmodified, while secure key management practices protect private keys from unauthorized access.

Performance Evaluation

Research shows that RSA with 1024-bit keys offers a practical balance of speed and security, while larger keys significantly improve resistance to brute-force attacks at the expense of system performance. Hybrid encryption systems, which use RSA only for key exchange and symmetric encryption algorithms like AES for data encryption, enhance performance by lowering the computational burden. These systems have become standard in secure communication because they combine the strength of prime-based encryption with the speed of symmetric ciphers.

3.3 Blockchain and Cryptocurrency

Ber-resistant architecture. Prime numbers are essential to block creation, mining, wallet authentication, and cryptocurrency systems rely on cryptography to protect their decentralized nature and ensure authentication and transaction signing. Cryptocurrencies like Bitcoin and Ethereum primarily use elliptic curve cryptography, especially the Elliptic Curve Digital Signature Algorithm (ECDSA), which works over finite fields defined by large primes. Hashing algorithms such as SHA-256 maintain block integrity, secure mining operations, and link blocks through cryptographically protected references. This design prevents unauthorized modifications.

Cryptographic Layer in Blockchain

Blockchain architecture uses multiple layers of cryptography at once. RSA can be used for wallet encryption, node identity authentication, or secure communication between clients and servers. ECDSA, functioning over prime fields, signs transactions to authorize cryptocurrency transfers. Hashing methods like SHA-256 secure block headers and serve as the foundation of the Bitcoin proof-of-work system. These cryptographic layers work together to ensure transparency, security, and consensus within a decentralized network.

RSA in Blockchain

Even though elliptic curve algorithms dominate blockchain transaction signing, RSA still plays a key role in supporting security measures. Wallet backups and communications between nodes often use RSA encryption to block unauthorized access. In enterprise or permissioned blockchain networks, RSA certificate systems are used to verify user roles and control participation. Thus, RSA complements ECC by adding layers of secure identity management and communication.

ECDSA and Prime Curve Operations

ECDSA delivers robust cryptographic security with smaller key sizes compared to RSA, making it ideal for large-scale, high- throughput blockchain environments. Bitcoin uses the elliptic curve secp256k1, defined over a large prime field with modulus p = 2^2 $- 2^{32} - 977$. This prime modulus allows for efficient computation and strong resistance to discrete logarithm attacks. Because elliptic curve operations provide security comparable to 3072-bit RSA keys while using only 256-bit keys, they lower storage needs and boost computational efficiency for digital wallets and blockchain nodes.

Hash Functions in Blockchain

Hash functions are crucial for blockchain integrity. SHA-256, based on modular arithmetic involving prime components, secures mining, transaction verification, and block chaining. Each block contains the hash of the previous block, linking the entire chain. Altering a single transaction changes the block's hash, requiring the recalculation of all following blocks—an impossible task without majority control over the network's computational power. This cryptographic structure ensures immutability and resistance to tampering.

Limitations of Prime-Based Algorithms

Despite their strengths, prime-based algorithms face new challenges. RSA requires large keys, which can become inefficient for resource-limited blockchain nodes. ECC, while efficient, could be vulnerable to potential quantum algorithms that can solve discrete logarithms quickly. As quantum computing advances, blockchain networks must look for post-quantum alternatives to maintain long-term security.

3.4 Password Authentication and Data Transmission

Password authentication is a vital part of digital security. However, traditional systems often have weaknesses, such as predictable hashing, susceptibility to brute-

force attacks, and risks from database breaches. A modern hybrid authentication framework that combines RSA, SHA-256 hashing, and Encrypted Negative Passwords (ENP) offers better security by using prime-based cryptographic methods. This hybrid approach reduces vulnerabilities through multiple layers of cryptographic transformation, ensuring that even if stored data is compromised, recovering the original password remains nearly impossible.

Vulnerabilities in Traditional Systems

Traditional authentication systems typically store hashed versions of passwords, but attackers can take advantage of flaws in hashing techniques. Rainbow table attacks, dictionary attacks, and brute-force attempts often succeed when passwords are simple or predictable. Many users recycle passwords across platforms, so a breach in one system can endanger multiple accounts. Poorly configured hashing methods, such as unsalted hashes or outdated algorithms like MD5, further increase the risk of compromise. These weaknesses highlight the need for stronger authentication models that use layered cryptographic protections based on prime arithmetic.

ENP + RSA Hybrid Framework

The ENP-RSA hybrid model reinforces password security through a multi-step cryptographic process. First, the user's password is hashed using SHA-256, creating a fixed-length digest. This digest is then changed into an Encrypted Negative Password, a negative representation stored in a unique negative database to block direct extraction. RSA encryption adds further protection by encrypting the negative representation before transmission or storage. Since RSA relies on large prime numbers for key generation, the encrypted negative database values become very hard to reverse. Even if attackers access the database, reconstructing the original password or its hashed value is nearly impossible. This hybrid framework, therefore, improves confidentiality and authentication reliability.

Two-Phase Authentication

The hybrid authentication system works in two main phases. During registration, the user's password goes through hashing, negative transformation, and RSA encryption before being stored. During login, the system decrypts the stored ENP representation using RSA and then recreates the negative version of the password entered by the user. The system compares this generated negative form with the stored negative value. Authentication succeeds only when both match. This two-phase process means that even if attackers intercept communication or access stored data, they cannot reverse-engineer the password due to the combination of hashing, negative encoding, and prime-based encryption.

Secure Multimedia Transmission

Besides authentication, the hybrid ENP-RSA framework supports secure multimedia transmission. Once a user is authenticated, files like images, videos, and documents can be encrypted using RSA before being sent. Since RSA ensures that only authorized recipients with the correct private key can decrypt the multimedia content, the system maintains confidentiality and blocks unauthorized access. This feature is especially valuable for platforms handling sensitive or classified data.

3.5 Electronic Banking and Card Transactions

Electronic banking systems operate in environments where financial transactions face constant threats from cyberattacks, including identity theft, replay attacks, card cloning, and phishing attempts. To protect customers and financial institutions, modern banking infrastructures use cryptographic systems based on prime-number-based algorithms like RSA. When combined with One-Time Passwords (OTPs) generated through prime-seeded pseudo-random number generators, these systems create multi-layered security architectures that greatly lower the chances of unauthorized access.

Need for Multi-Layered Banking Security

The sensitivity and frequency of financial transactions make banks prime targets for cybercriminals. Attackers exploit weaknesses through malware, social engineering, skimming devices, insecure networks, and hacked databases. Without strong cryptographic protections, critical information such as account details, transaction histories, and customer identities can be stolen or manipulated. Multi-layered security models that combine RSA encryption, OTP-based verification, and secure communication protocols offer solid protection. This setup ensures that even if one layer is compromised, subsequent layers maintain system integrity and block unauthorized access.

RSA for Confidential Transaction Processing

RSA is crucial for securing banking transactions. Sensitive information such as ATM PINs, login credentials, card numbers, and transaction messages is encrypted using the bank's public key. Since the private key is known only to the bank's secure server, attackers cannot decrypt intercepted data. OTP systems enhance RSA by generating random one-time codes that change for each login attempt or transaction. These codes, often produced using prime-based pseudo-random algorithms, prevent attackers from reusing credentials or exploiting intercepted information.

System Architecture

A typical secure banking architecture includes various components, such as the user authentication interface, RSA encryption modules, OTP verification services, and backend transaction processors. When a user starts a banking session, RSA ensures the confidentiality of transmitted information, while OTP serves as an extra verification factor. The backend system checks the encrypted data using the private key, authenticates the OTP, and then processes the transaction. This architecture guarantees that even if an attacker acquires the user's password or tries to intercept communication, unauthorized access remains impossible without

the matching OTP and cryptographic keys.

Experimental Outcomes

Studies show that integrating RSA with OTP significantly improves the security of online banking platforms. This combination protects against replay attacks, phishing attempts, brute-force intrusions, and man-in-the-middle attacks. While RSA-based encryption adds some computational overhead, the delay is minimal compared to the security benefits. Modern hardware accelerators and optimized cryptographic libraries enable financial institutions to process encrypted transactions efficiently, ensuring user satisfaction and strong security compliance.

3.6 Cloud Computing, Big Data, and VPN Security

Cloud computing environments handle and store large amounts of sensitive data across distributed systems, making them attractive targets for cyberattacks. To address these risks, cryptographic systems based on RSA encryption, SHA-256 hashing, and VPN tunneling offer solid protection. These systems secure cloud infrastructure by ensuring confidentiality during data transmission, authenticating users and devices, validating certificates, and guarding virtual network traffic from unauthorized access.

Cloud Security Challenges

Cloud systems face serious vulnerabilities, including insecure storage setups, unencrypted APIs, insider threats, unauthorized data access, and session hijacking. Since multiple users and services share the same physical infrastructure, proper isolation and encryption are crucial. Weak encryption methods, poor key management, and inadequate security protocols can lead to large-scale data breaches. Prime-number-based cryptographic approaches like RSA provide strong protection by securing communication channels, encrypting stored data, and enforcing strict authentication methods that prevent unauthorized access to cloud resources.

RSA for Data Encryption

RSA is vital for protecting data sent to and from cloud environments. It encrypts file uploads, API requests, and communication between cloud nodes using the public keys of designated recipients. Since the private keys for decryption remain securely stored within the cloud infrastructure, attackers cannot access sensitive data even if they intercept encrypted traffic. RSA also supports certificate-based authentication, making sure that only legitimate users and devices can access cloud resources.

VPN for Network Protection

Virtual Private Networks (VPNs) act as secure communication tunnels between users and cloud servers. Protocols like IPSec and OpenVPN often rely on RSA-signed certificates for endpoint authentication. Once a connection is established, all transmitted data is encrypted within the VPN tunnel, protecting it from packet sniffing, man-in-the-middle attacks, and unauthorized access. VPN systems using RSA and secure hashing methods ensure that cloud traffic remains encrypted and tamper-proof, even over public networks.

Big Data Integrity and Authentication

In large big data environments, multiple nodes exchange and process large amounts of sensitive information. RSA digital signatures confirm that datasets are authentic and unchanged throughout their transmission and storage lifecycle. Hash functions like SHA-256 validate the integrity of data blocks, ensuring no unauthorized changes occur. Because big data systems manage information from various sources, authentication and integrity validation are crucial for maintaining accuracy and trustworthiness in analytical operations.

The combination of RSA encryption and VPN tunneling significantly boosts cloud security. This dual-layered protection guarantees confidentiality, verifies integrity, and authenticates communication endpoints. RSA-VPN setups perform consistently well in large distributed systems, providing a strong defense

against evolving cyber threats. These models show that prime-number-based cryptographic techniques remain effective in securing modern cloud infrastructures and protecting the privacy of big data systems.

Prime-number-based cryptography is essential in modern digital applications, providing key security mechanisms that protect sensitive information across various environments. Systems like digital signatures, secure communication protocols, blockchain networks, password authentication frameworks, electronic banking infrastructures, cloud computing environments, and VPN architectures rely on the complexity of prime-based mathematical operations to maintain confidentiality, integrity, authenticity, and non-repudiation. While emerging technologies like quantum computing pose new challenges to prime-based systems, these algorithms continue to be central to cybersecurity frameworks used worldwide. The integration of RSA, Diffie–Hellman, ECDSA, SHA-256, OTPs, and hybrid authentication models highlights the lasting importance of prime numbers in enabling secure, reliable digital communication systems.

Chapter 4

COMPARATIVE ANALYSIS OF PRIME-BASED CRYPTOSYSTEMS

Prime numbers form the mathematical basis of modern asymmetric cryptographic systems. They are essential for secure communication across digital infrastructures. Their key properties—indivisibility, difficulty of factorization, and ability to create cyclic groups—make them vital in constructing cryptosystems like RSA, Diffie-Hellman (DH), the Digital Signature Algorithm (DSA), and Elliptic Curve Cryptography (ECC). Over the years, these systems have become fundamental to network security, cloud communication, e-commerce, banking authentication, wireless sensor networks (WSNs), and safe web browsing. Because primes act unpredictably and become less frequent as numbers increase, they create computational challenges. These problems are easy to solve in one direction but extremely hard to reverse, such as integer factorization and discrete logarithms. This difficulty is essential for achieving confidentiality, integrity, authentication, and non-repudiation.

This chapter presents a comparative analysis of prime-based cryptosystems by exploring their theoretical, mathematical, and practical operations. It includes insights from cloud security research, WSN studies, and traditional public-key cryptography literature. It focuses particularly on implementation challenges within

resource-limited environments like WSNs, where memory, energy, and computing power restrictions limit the use of more complex cryptosystems. The chapter also investigates how cloud infrastructures use these algorithms to ensure confidentiality, secure data transmission, virtual machine isolation, access control, and multi-tenancy protection. By evaluating performance, security strength, vulnerability to attacks, parameter selection, and relevance to modern digital applications, this chapter offers a complete overview of how prime-based cryptography operates across various technological settings.

4.1 RSA Cryptosystem

Theoretical Foundation

The RSA cryptosystem is based on the complexity of the integer factorization problem, which involves finding the original prime numbers from their product. While multiplying two large primes is straightforward, reversing the computation is virtually impossible when the primes have several hundred digits. RSA also relies on modular arithmetic, particularly modular exponentiation and Euler's totient function, to create mathematically linked public and private keys. This one-way nature of the computation ensures that while encryption and signature verification are relatively quick, decryption and signature generation require much more processing because of the size of the private exponent.

Use of Prime Numbers

Prime numbers are crucial to RSA. The algorithm starts by selecting two large, random, independent primes, usually generated through probabilistic primality testing methods like Miller-Rabin. These primes are multiplied to create the modulus, which is used for both encryption and decryption. The quality and randomness of the primes determine RSA's overall security. If the primes are too small or poorly chosen, the modulus becomes vulnerable to factorization attacks. Research on WSN cryptography shows that large RSA keys significantly drain energy, memory, and processing power from sensor nodes, making RSA less suitable

for lightweight environments compared to ECC.

Algorithmic Operation

RSA operates through key generation, encryption, decryption, and digital signing. Key generation is intensive and involves producing two large primes, calculating the totient, picking a suitable public exponent, and finding a modular inverse to create the private key. Encryption involves raising plaintext to the public exponent modulo the composite number, which is relatively fast. However, decryption requires using a large private exponent, which demands more processing power. The digital signature process reflects this relationship, with signing being slow and verification fast. This structure has made RSA popular for certificate verification and key exchange, but less commonly used in environments that require frequent signing.

Security Characteristics

RSA's security relies on the size of the modulus and effective key management. Keys smaller than 1024 bits are now considered insecure, and strong security typically starts at 2048 bits. Cloud environments usually use RSA-2048 or RSA-3072 for authentication and certificate management. However, RSA's main vulnerability lies in its exposure to quantum attacks, especially from Shor's algorithm, which can factor large numbers quickly on a powerful quantum computer. Side-channel attacks, like timing analysis and power fault attacks, also pose risks if implementations do not use constant-time operations and secure padding.

Performance Considerations

RSA is computationally demanding, particularly during decryption and signing, making it less ideal for low-energy sensor nodes, mobile IoT devices, and battery-powered environments. Studies in WSN security show that RSA consumes more energy and memory than ECC and even Diffie-Hellman. Its large key sizes lead to more transmission overhead and bandwidth usage, further reducing suitability for

lightweight networks. Still, RSA is vital in cloud infrastructures due to its established trust, compatibility, and integration into protocols like TLS, PKI systems, secure boot mechanisms, digital certificates, and encrypted email communication.

4.2 Digital Signature Algorithm (DSA)

Theoretical Foundation

The Digital Signature Algorithm is built on the discrete logarithm problem in the multiplicative group modulo a prime number. This involves determining the exponent of a generator that produces a given element in the group, a task that becomes difficult when the prime modulus and subgroup order are large. DSA's design includes a public parameter set consisting of a large prime modulus, a smaller prime divisor, and a generator of the related subgroup. These parameters create a mathematically structured environment for securely generating and verifying digital signatures.

Use of Prime Numbers

DSA uses two primes: one large prime (p) that defines the field and a smaller prime (q) that divides (p - 1), ensuring the existence of a prime-order subgroup. The generator is calculated from these parameters, creating a well-structured cryptographic system. The careful selection of primes ensures that discrete logarithm attacks remain computationally infeasible. However, DSA's security strongly depends on the randomness of ephemeral keys; poor randomness can expose the private key, as shown in some real-world breaches.

Algorithmic Processes

The DSA signing process uses a new, random ephemeral key for each signature. The signature is generated by combining this random value with a hash of the message. Verification involves computing two modular exponentiations using the public key and signature components. Signing is quick, while verification requires

more operations and is typically slower. Despite this, DSA is widely adopted in various government frameworks, authenticated software distribution systems, and secure electronic document workflows.

Security Strengths and Weaknesses

DSA provides strong authentication, data integrity, and non-repudiation. Its security relies on the secrecy, uniqueness, and high randomness of the ephemeral keys used for signing. If these values are reused or predictable, the entire system can be compromised. Unlike RSA, which faces factorization threats, DSA is mainly vulnerable to lattice attacks and key recovery attacks due to weak randomness. In cloud systems, DSA is often used for validating digital signatures on virtualized resources, identity verification, and software integrity.

Performance and Application

DSA is efficient at generating signatures, making it suitable for high-load systems that produce frequent authentication tokens or logs. Its performance in WSNs is mixed; while signing is quick, the verification process can be energy-intensive. Nevertheless, DSA remains widely used in secure communication protocols, financial authentication systems, and information integrity systems.

4.3 Diffie-Hellman Key Exchange

Theoretical Basis

Diffie-Hellman was the first algorithm that allowed two parties to establish a shared secret over an insecure communication channel without prior contact. It is based on the discrete logarithm problem modulo a large prime. More specifically, Diffie-Hellman involves exponentiation using a generator of a prime-order multiplicative group. The shared secret produced from the exchanged public values cannot be computed by an attacker who intercepts the communication, assuming the prime modulus is large enough.

Role of Prime Numbers

DH requires a large prime (p) and a primitive root (g) modulo (p). In secure implementations, these primes are typically chosen as "safe primes," of the form ($p = 2q + 1$), where (q) is also a prime, providing defense against subgroup attacks. The choice of primes is essential for maintaining resistance to discrete logarithm attacks.

Algorithmic Operation

Both parties pick private random exponents and compute their public values by exponentiating the generator. After they exchange these values, each party calculates the shared secret by raising the received public value to their own private exponent. The resulting shared key is identical for both parties due to exponentiation properties. However, since the exchange is unauthenticated, DH is vulnerable to man-in-the-middle attacks unless combined with authentication mechanisms like digital signatures or certificates.

Security and Performance

Diffie-Hellman offers strong security when implemented with large primes (2048–4096 bits). It is simpler from a computational perspective than RSA, making it appealing for session key negotiation in secure web communication, VPN tunnels, and cloud service authentication. However, the lack of built-in authentication and vulnerability to MITM attacks necessitate additional security layers. In WSNs, DH is often seen as costly compared to ECC due to its reliance on large primes and repeated modular exponentiations.

4.4 Elliptic Curve Cryptography (ECC)

Mathematical Foundation

ECC builds on the algebraic structure of elliptic curves over finite prime fields. Instead of using large integer composites or classical discrete logarithms, ECC establishes security through the elliptic curve discrete logarithm problem, which is more complex for each bit. The main idea involves scalar multiplication of a point on an elliptic curve, a one-way operation that is easy to execute but nearly impossible to reverse without solving the underlying hard problem.

Use of Prime Fields

ECC works over prime fields (GF(p)), where the prime (p) defines the computational setting for elliptic curve arithmetic. Smaller keys can provide much stronger security because elliptic curves create a higher level of complexity, allowing for compact representations and reduced computational demands. Research in WSN security consistently shows that ECC offers better energy efficiency, faster computations, and less bandwidth usage compared to RSA.

Algorithmic Behavior

ECC supports secure key exchange, encryption, and digital signatures. Scalar multiplication is the foundation of all operations. ECC-based signatures, like ECDSA, provide shorter signature lengths while ensuring robust security properties. Key generation is straightforward and efficient, requiring less processing compared to RSA's prime generation phase.

Security Properties

ECC provides high security levels with smaller key sizes, making it ideal for cloud systems, IoT devices, and WSNs. A 256-bit ECC key is roughly equivalent in security to a 3072-bit RSA key, reducing memory needs and computational load.

Although ECC, similar to RSA and DH, is vulnerable to quantum computing (via Shor's algorithm), its efficiency makes it the preferred option for modern secure systems.

Implementation and Applications

Due to its low power usage and small key sizes, ECC is commonly used in blockchain systems, secure mobile messaging protocols, cloud-based authentication systems, and lightweight embedded devices. It is particularly valued in systems that demand rapid authentication and minimal data transfer, making it the leading cryptographic choice for next-generation digital infrastructures.

4.5 Comparative Discussion

RSA, DSA, Diffie-Hellman, and ECC are the most influential prime-based cryptosystems, each designed for different computing environments and security needs. RSA is great for compatibility but suffers from large key sizes and high computing requirements. DSA offers efficient signing but relies heavily on randomness. Diffie-Hellman facilitates secure key exchanges but must be combined with authentication to prevent man-in-the-middle attacks. ECC provides the strongest security per bit and the best energy efficiency, establishing itself as the top cryptosystem for modern cloud architectures and resource- constrained wireless sensor networks.

Research in WSNs and cloud security consistently identifies ECC as the best option for limited environments, while RSA is more suitable for certificate validation and legacy systems. DSA and DH play critical roles in authentication and secure key exchange, respectively, but they face issues when randomness or prime parameters are not well managed. Throughout all these systems, prime numbers remain central, delivering the mathematical foundations necessary to secure global communication infrastructures.

Prime numbers are the foundation of modern asymmetric cryptography. They allow secure communication through problems that are easy to solve one way but

difficult to reverse. Each prime-based cryptosystem, RSA, DSA, Diffie-Hellman, and ECC, plays a unique role in digital security. RSA is still the leading choice for certificate-based authentication, even with its performance issues. DSA improves digital signature systems with strong verification methods. Diffie-Hellman enables secure key exchanges, which are crucial for encrypted communication. ECC uses the elegance of elliptic curves over prime fields to offer high efficiency and security, making it the top choice for new technologies.

As cloud computing, IoT, and WSN applications grow, the need for strong and efficient prime-based cryptosystems rises. While quantum computing poses a long-term risk to traditional public-key methods, the principles based on prime number math continue to shape the development of new cryptographic models. It is important to understand the differences, strengths, and limitations of these systems to design secure and future-ready communication systems.

Table 4.1: Comparative Analysis of Prime-Based Cryptosystems

Parameter	**RSA**	**DSA**	**Diffie–Hellman (DH)**	**Elliptic Curve Cryptography (ECC)**
Underlying Hard Problem	Integer factorization problem; reversing the product of two large primes is infeasible.	Discrete logarithm problem in a prime-order subgroup.	Discrete logarithm problem modulo a large prime.	Elliptic curve discrete logarithm problem over prime fields.
Use of Prime Numbers	Uses two large random primes to generate modulus; security depends on prime quality.	Uses large prime p and smaller prime q dividing $p-1$.	Requires large prime modulus (often safe prime $p = 2q+1$) and primitive root g.	Works over finite prime fields GF(p); primes define curve environment.
Key Sizes	Very large keys (2048–3072 bits); heavy for IoT/WSN.	Large primes but smaller than RSA; depends on group order.	Requires large primes (2048–4096 bits).	Much smaller keys; 256-bit ECC $\approx$ 3072-bit RSA.
Performance	Signing slow, verification fast; key generation very slow.	Signing fast; verification slower; randomness dependent.	Efficient shared secret computation but heavy exponentiation.	Very fast; low computation and energy usage.
Suitability for IoT/WSN	Poor — high energy, memory, and processing cost.	Moderate — signing efficient but verification heavy.	Moderate/Low — exponentiation costs high.	Excellent — best for WSN, IoT, mobile devices.
Security Strengths	Strong with large keys; widely trusted; excellent for PKI.	Strong when randomness is perfect.	Excellent for secure key exchange; used in TLS, VPN.	Highest security per bit; efficient and scalable.
Main Weaknesses	Very slow in constrained devices; vulnerable to quantum (Shor).	Randomness failures expose private key; lattice attacks possible.	MITM attacks unless authenticated; needs large primes.	Quantum vulnerable; careful curve selection required.
Main Applications	TLS/SSL, certificates, secure email, cloud authentication.	Government signatures, software integrity.	Key exchange in VPN, TLS, secure messaging.	Blockchain, secure messaging, IoT, WSN, cloud, mobile cryptography.
Quantum Resistance	Not resistant.	Not resistant.	Not resistant.	Not resistant (but more efficient pre-quantum).

Chapter 5

IMPLEMENTATION AND VALIDATION OF RSA-BASED DIGITAL SIGNATURES FOR PDF DOCUMENTS

Digital signatures are an essential component of modern electronic commerce and secure communication. They serve to authenticate the identity of the signer and ensure the integrity of the data. This paper focuses on the application of the widely accepted RSA algorithm for generating a digital signature and embedding it within a PDF document. The PDF format is universally used for sharing fixed-layout documents, making its security a paramount concern in various sectors, including legal, financial, and academic institutions. Our work details the full process, from key generation to final validation, providing a practical blueprint for secure document handling.

5.1 The Foundational Mathematics and History of the RSA Algorithm

The security of modern digital communications relies on public-key cryptography, also known as asymmetric cryptography. The Rivest, Shamir, Adleman (RSA) algorithm was formally introduced by Ronald Rivest, Adi Shamir, and Leonard

Adleman in 1977 and published in 1978 [1]. It remains a key part of this field.

The security of the RSA cryptosystem depends on the difficulty of the Integer Factorization Problem (IFP). It is easy to multiply two large prime numbers (p and q) to get a composite number (N). However, classical computers struggle to reverse this process and find p and q given just N, especially when N is large, such as 2048 bits or more.[2]

The mathematical operation essential to both encryption and digital signatures in RSA is modular exponentiation. Key generation involves selecting two large primes, p and q, computing the modulus $N = p \cdot q$, and determining the Euler totient function $\phi(N) = (p-1)(q-1)$. The public exponent e and private exponent d are chosen to be multiplicative inverses modulo $\phi(N)$, fulfilling the relation $e \cdot d \equiv 1 \pmod{\phi(N)}$. The public key is the pair (N, e), while the private key is (N, d) or (p, q, d).

The basic idea of asymmetric key cryptography was discovered earlier by Clifford Cocks at GCHQ in 1973. However, the broad implementation and standardization of the public-key system are credited to Rivest, Shamir, and Adleman [1]. RSA has become a vital industry standard, commonly used for key exchange, encryption, and digital signatures in secure internet protocols.

5.2 Standards and Protocols for Digital Identity and Document Security

To effectively use RSA in digital signature applications, it is important to follow several globally accepted standards and protocols for digital identity and document formats.

Digital Certificates (X.509)

Digital certificates link a public key to an identifiable entity like a person, organization, or device. The main standard for these certificates is X.509, defined by the International Telecommunication Union (ITU). An X.509 certificate includes the entity's public key, identification details (such as name and email), and a digital signature from a Certificate Authority (CA) that confirms the certificate's authenticity and integrity[2]. This framework creates a Public Key Infrastructure (PKI) that is essential for establishing trust in digital identities used for signing

documents.

Secure Key and Certificate Storage (PKCS#12/PFX)

For a digital signature to work, the private key and its related X.509 certificate need to be stored securely. The PKCS#12 (Public-Key Cryptography Standards #12) format, recognized by the file extensions `.pfx` or `.p12`, provides a standard for storing this sensitive information.[3] PKCS#12 is a password-protected archive file used to safely package the user's private key along with the public key certificate chain. This standard ensures the portability and secure storage of the digital identity, making it the preferred format for issuing and distributing signing credentials in businesses and academic settings.

PDF Digital Signature Specification

The Portable Document Format (PDF), managed by Adobe Systems and later standardized as ISO 32000, has a solid specification for digital signatures [**?**].[4] When a digital signature is added to a PDF, the signing software computes a hash (digest) of the document's content before adding the signature. This hash is then encrypted using the signer's RSA private key to create the digital signature. The resulting signature object is embedded in the PDF file structure, often allowing for incremental saving without invalidating the signature. Validation software, such as Adobe Acrobat Reader, uses the signer's public key (from the embedded X.509 certificate) to decrypt the signature, re-calculate the document's hash, and confirm that both hashes match, which verifies the signer's identity and the document's integrity. The implementation discussed in this paper uses Python libraries, including the popular `cryptography` library, for generating and manipulating these cryptographic components [5].

5.3 Methodology

5.3.1 Key and Certificate Generation (Digital ID Creation)

This establishes the signer's unique cryptographic identity.

1. RSA Key Pair Generation – A Private Key and Public Key have been created. An asymmetric cryptographic key pair has been generated using the RSA

algorithm, which is 2048-bit. The private key is kept secret for signing, and the public key is used for verification.

2. Certificate Creation – An X.509 Certificate has been created in Python. This certificate ties the Public Key to the signer's identity (Distinguished Name) and includes validity dates. It's often self-signed for internal use, acting as the document's trust anchor.

Listing 5.1: Code Block 1: Key and Certificate Generation

```
# --- Configuration (Placeholders used) ---
# NOTE: The actual file names used in our operational system have been
    replaced
# with generic names for publication.
PRIVATE_KEY_FILE = "private_key.pem"
CERTIFICATE_FILE = "certificate.pem"
KEY_SIZE = 2048 # Recommended key size for RSA
CERT_COMMON_NAME = u"Digital Signature"
# NOTE: Specific organizational/geographic details have been replaced with
# generic examples for publication.
CERT_ORG_NAME = u"Research Identity"

def generate_key_and_cert():
print("Starting key and certificate generation...")

# 1. Create the Private Key (Your secret file)
# NOTE: public_exponent and key_size remain standard for reproducibility.
private_key = rsa.generate_private_key(
public_exponent=65537,
key_size=KEY_SIZE,
)

# 2. Define the Certificate Details
# NOTE: The values for Country, State, and Common Name are replaced with
    generic examples.
subject = issuer = x509.Name([
x509.NameAttribute(NameOID.COUNTRY_NAME, u"US"),       # Replaced actual
    country ('Kerala')
x509.NameAttribute(NameOID.STATE_OR_PROVINCE_NAME, u"ExampleState"), #
    Replaced actual state
x509.NameAttribute(NameOID.COMMON_NAME, CERT_COMMON_NAME),   # Used
    placeholder CN
x509.NameAttribute(NameOID.ORGANIZATION_NAME, CERT_ORG_NAME),
])

# 3. Build and Sign the Certificate (Self-Signed)
cert = x509.CertificateBuilder().subject_name(
subject
).issuer_name(
```

```
issuer
).public_key(
private_key.public_key()
).serial_number(
x509.random_serial_number()
).not_valid_before(
datetime.datetime.utcnow()
).not_valid_after(
datetime.datetime.utcnow() + datetime.timedelta(days=365)
).sign(
private_key, hashes.SHA256()
)

# 4. Save the Files to your computer (File paths use placeholders)
# NOTE: No encryption is applied to the key for simplicity in this script, as
    shown in the original.
with open(PRIVATE_KEY_FILE, "wb") as f:
f.write(private_key.private_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PrivateFormat.PKCS8,
encryption_algorithm=serialization.NoEncryption()
))

with open(CERTIFICATE_FILE, "wb") as f:
f.write(cert.public_bytes(serialization.Encoding.PEM))

print(f"Success! Files created: {PRIVATE_KEY_FILE} and {CERTIFICATE_FILE}")

# generate_key_and_cert() # Commented out the execution call
```

3. PKCS#12 (PFX/P12) Bundle / Digital ID File – The private key and certificate file have been combined into a single password-protected file in `.pfx` format. This formulates a `digital ID` file holding the signing credentials.

Listing 5.2: Code Block 2: PKCS#12 (PFX) Creation

```
        from cryptography.hazmat.primitives import serialization
        from cryptography.hazmat.primitives.serialization import pkcs12
        from cryptography.x509 import load_pem_x509_certificate

        # --- Configuration (Placeholders used) ---
        # NOTE: Specific file paths and the key password have been replaced
            for publication.
        PRIVATE_KEY_PATH = "private_key.pem" # Placeholder path
        CERTIFICATE_PATH = "certificate.pem" # Placeholder path
        PFX_OUTPUT_PATH  = "output.pfx"      # Placeholder path
        # NOTE: The actual password used to protect the PFX file has been
            replaced.
        PFX_PASSWORD = b"pfx-strong-password-1234" # Placeholder password
```

```
# Load private key
with open(PRIVATE_KEY_PATH, "rb") as key_file:
private_key = serialization.load_pem_private_key(
key_file.read(),
password=None # Original code used 'password=None'
)

# Load certificate
with open(CERTIFICATE_PATH, "rb") as cert_file:
certificate = load_pem_x509_certificate(cert_file.read())

# Create the PKCS#12 container (PFX file)
pfx_data = pkcs12.serialize_key_and_certificates(
name=b"FriendlyNameForResearch", # Replaced 'Friendly name for key
    entry'
key=private_key,
cert=certificate,
cas=None, # No additional certificates chain
encryption_algorithm=serialization.BestAvailableEncryption(
    PFX_PASSWORD) # Password protect PFX
)

# Write PFX file
with open(PFX_OUTPUT_PATH, "wb") as pfx_file:
pfx_file.write(pfx_data)

print(f"Successfully created PFX file at {PFX_OUTPUT_PATH}")
```

5.3.2 PDF Signing (The Signing Process)

This phase applies the cryptographic credentials to the document.

1. Document Hashing – The entire content of the input PDF (excluding the future signature field) is processed through SHA-256 to create a unique, fixed-length message digest.

2. Signature Creation (Encryption) – The generated message digest is encrypted using the Private Key contained in the password-protected PFX file. The result is the Digital Signature. This ensures non-repudiation and authenticity.

3. Embedding the Signature – The Digital Signature and the Public Key Certificate (from the PFX/P12 file) are embedded into the PDF structure, usually in a visible signature field. This creates the final signed PDF.

Listing 5.3: Code Block 3: PDF Digital Signing

```
from pyhanko.sign import signers
from pyhanko.pdf_utils.incremental_writer import IncrementalPdfFileWriter

# --- Configuration (Sanitized & Placeholder) ---
PFX_PASSWORD = b'pfx-strong-password-1234'

# Load the PFX file and create a signer object
signer = signers.SimpleSigner.load_pkcs12(
pfx_file='output.pfx',
passphrase=PFX_PASSWORD
)

# Open the PDF you want to sign
with open('input.pdf', 'rb') as infile:
# FIX: Added allow_hybrid_xrefs=True to resolve the previous SigningError
writer = IncrementalPdfFileWriter(infile, allow_hybrid_xrefs=True)

# Sign the PDF incrementally
signed_pdf = signers.sign_pdf(
writer,
signers.PdfSignatureMetadata(field_name='Signature'),
signer=signer,
)

# Write the signed PDF to a new file
with open('signed_output.pdf', 'wb') as outfile:
outfile.write(signed_pdf.getbuffer())

print("PDF signed successfully as signed_output.pdf")
```

5.3.3 Signature Validation (Verification)

This phase confirms the signature's authenticity and integrity using a third-party application—Adobe Acrobat.

1. **Hash Re-generation** – Adobe Acrobat extracts the PDF content and re-generates its message digest using the same hash algorithm specified in the signature metadata.

2. **Signature Decryption** – Adobe Acrobat extracts the Digital Signature and uses the embedded Public Key from the certificate to decrypt it, retrieving the original hash generated by the signer.

3. Comparison and Validation – The validation software compares the re-generated hash with the decrypted hash. If they match, the signature is valid, confirming that the document's content has not been changed since it was signed. Acrobat also checks the validity of the embedded certificate (e.g., expiry date, and trust status).

5.4 Validation And Results

This section presents the outcomes of the digital signing process and the following verification of the signature to confirm its authenticity and integrity.

Verification Process

To validate the digital signature, we opened the signed PDF document using Adobe Acrobat Reader, which has built-in verification features. The following steps were taken:

1. **Open the signed document:** We opened the digitally signed file in Adobe Acrobat Reader.
2. **Access Signature Panel:** We used the Signatures panel to view the details of the signature.
3. **Verify Signature Validity:** Adobe automatically checked the signature using the embedded certificate.
4. **Check Status Message:** A pop-up message confirmed that the signature was valid, indicating:
    - The document has not been altered since signing.
    - The signer's identity is trusted and verified.

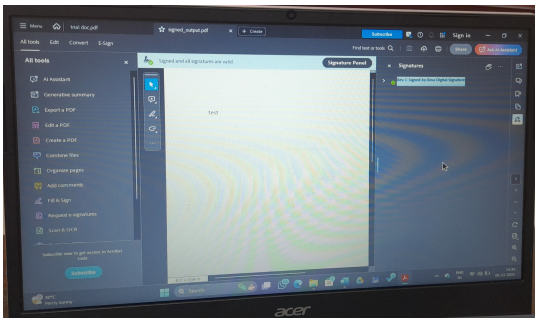

Figure 5.1:

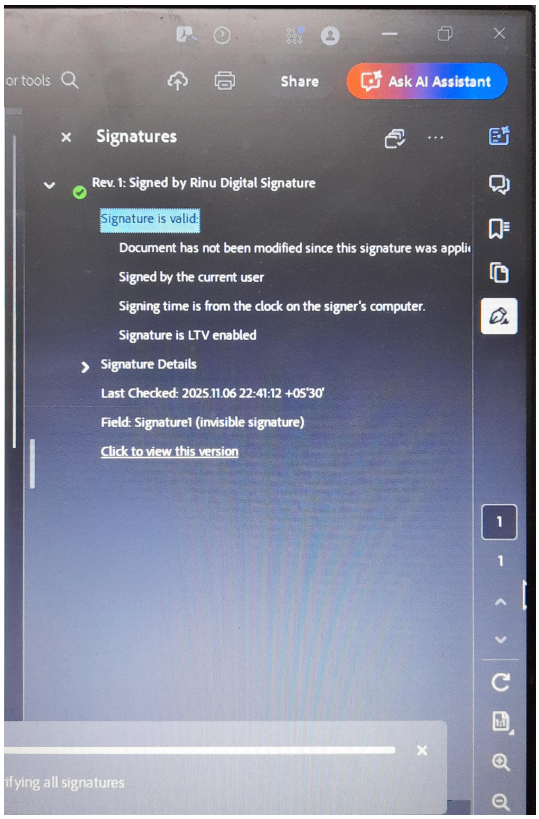

Figure 5.2:

Security Analysis

The RSA algorithm ensures document authenticity and integrity through public-key cryptography. It protects against two main threats:

Forgery: The private key used to generate the digital signature is known only to the signer. Without this key, no one can create a valid signature. This prevents impersonation or forgery.

Repudiation: Once a document is signed, the signer cannot deny authorship because the signature can be verified with their public key.

Tampering after Signing: If someone modifies the signed document after signing (for example, changing text or values), the cryptographic hash of the document will differ from the original. When verification is attempted in Adobe Acrobat Reader, the signature will be marked as "Invalid" or "Document has been altered or corrupted after signing." This confirms that the document's integrity has been compromised. Thus, the RSA-based digital signature mechanism successfully ensures authentication, integrity, and non-repudiation. It provides strong protection against unauthorised modifications or forgeries.

CONCLUSION

The project successfully implemented a working model for digital document security using a Public Key Infrastructure (PKI) methodology. This process included key steps such as generating RSA key pairs, creating a self-signed X.509 certificate, securely bundling credentials into a PFX Digital ID, and using custom Python code to sign a PDF document. The final validation in Adobe Acrobat confirms that the basic requirements of a digital signature—authenticity, which proves the signer's identity, and data integrity, which verifies there was no tampering after signing—were technically met.

The project lays a foundational understanding of cryptographic signing. However, its future success depends on addressing the current trust model and key management vulnerabilities. By following the outlined research scope, the methodology can develop into a secure, scalable, and fully compliant digital signature system suitable for demanding enterprise environments.

Further Scope for Research and Enterprise Implementation

Establishing Trust via PKI Integration: *Research Focus:* Explore and integrate certificates issued by globally trusted Commercial Certificate Authorities (CAs) for external documents, or create an internal Organizational Certificate Authority (OCA) using technologies like Active Directory Certificate Services. *Goal:* Create an automatic chain of trust that ensures immediate and seamless validation in all standard document viewers.

Enhancing Key Management via Hardware Security: *Research Focus:* Use Hardware Security Modules (HSMs) or cryptographic Smart Cards that adhere to PKCS#11 standards. This requires changing the Python signing logic to work with the hardware, ensuring that the private key is generated and stored securely, never leaving the device. *Goal:* Significantly strengthen the private key's

security, improving non-repudiation and reducing theft risks.

Developing a Centralized Signing Service: *Research Focus:* Create a scalable, web-based signing service, like a REST API. This server would manage the entire Certificate Lifecycle, including issuance, renewal, and revocation, while handling high volumes of signing requests and maintaining control and audit logs for compliance. *Goal:* Automate the process, provide a centralized control point, and enable easy integration across various business applications.

Bibliography

[1] Nurhayat Varol, Abdalbasit Mohammed, "A Review Paper on Cryptography," Conference Paper, June 2019.

[2] Bhanu Sankhyan, Anupam Baiyan, Abhishek Kumar, "Review on Symmetric and Asymmetric Cryptography," International Journal for Research, March 2024.

[3] Rajeev Sobti, G. Geetha, "Cryptographic Hash Functions: A Review," International Journal of Computer Science, March 2012.

[4] Prabhat Mahato, Aayush Shah, "A Review of Prime Numbers, Squaring Prime Pattern, Different Types of Primes and Prime Factorization Analysis."

[5] Mit Patel, Alok Patel, G. Rajiv, "Prime Numbers and Their Analysis."

[6] Anjula Gupta, Navpreet Kaur Walia, "Cryptography Algorithms: A Review," IJEDR, Volume 2, Issue 2, ISSN: 2321-9939, 2014.

[7] R. L. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, 1978.

[8] P. W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," IEEE FOCS, 1994.

[9] R. Crandall, C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer, 2005.

[10] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[11] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson, 2017.

[12] J. Hoffstein, J. Pipher, J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer, 2008.

[13] J. Katz, Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2007.

[14] W. Diffie, M. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, 1976.

[15] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer, 1994.

[16] D. Boneh, "The Decision Diffie–Hellman Problem," ANTS, 1998.

[17] S. Goldwasser, M. Bellare, *Lecture Notes on Cryptography*, MIT, 2008.

[18] Modern_Cryptographic_Schemes_Application.pdf, personal project reference, 2025.

[19] "Modern Cryptographic Schemes Application – Shared Document," Google Drive, Available: https://share.google/s8UTl2TdOK2aSm6CCt.

[20] A. Almeshal, R. Hassan, Z. Shukur, "A Hybrid Framework for Secure Password Authentication and Data Transmission Using RSA and Hash-Based Encryption," Int. Journal of Computer Applications, 184(32), 2022.

[21] M. Anwar, S. Raza, M. Imran, "Improved Cloud Computing and Big Data Security Using RSA in VPN Tunneling Environments," Journal of Cloud and Distributed Systems, 2023.

[22] P. Sharma, E. Thomas, "A Security Framework for Electronic Banking and Card Transactions Using RSA Encryption and OTP-Based Authentication," Int. Journal of Information Security Research, 8(2), 2021.

[23] NIST, "Secure Hash Standard (SHS)," FIPS PUB 180-4, 2015.

[24] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, 48(177), 203–209, 1987.

[25] P. Singh, R. K. Chauhan, "A Survey on Comparisons of Cryptographic Algorithms Using Certain Parameters in WSN," International Journal of Electrical and Computer Engineering, 7(4), 2017.

[26] R. Sasikumar, S. Nagarajan, "Review and Analysis of Cryptography Techniques in Cloud Computing," International Journal of Cloud Computing Research, 2024.

[27] NIST, "Digital Signature Standard (DSS)," FIPS Publication 186-4, 2013.

[28] RSA Laboratories, "PKCS #12: Personal Information Exchange Syntax (Version 1.0)," 1999.

[29] Adobe Systems, "Digital Signatures in PDF: An Implementation Guide." Available: `http://www.adobe.com/devnet/pdf/pdf_reference.html`

[30] The Cryptography Developers, "Cryptography Documentation." Available: `https://cryptography.io/`