Mathematics and Mechanisms of Digital Security

Jus somethin i felt like doing

Rupak R. Gupta (RRG)

2025

Veermata Jijabai Technological Institute (VJTI)

For Her

Abstract

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

Acknowledgements

- Claude Shannon
- Veermata Jijabai Technological Institute
- Indian Institute of Technology, Bombay

Contents

CHAPTER 1

Network Security

1.1 Introduction to Network Security

1.1.1 CIA Triad

Confidentiality–Integrity–Availability (CIA)

1.2 Network Models

1.2.1 Client-Server

1.2.2 Peer-to-Peer (P2P)

Peer-to-Peer (P2P)

1.3 Network Attacks

1.3.1 Passive and Active Attacks

1.3.2 Eavesdropping

1.3.3 Spoofing

1.3.4 Man-in-the-Middle

1.3.5 Denial of Service

Denial of Service (DoS), Distributed Denial of Service (DDoS)

1.4 Security Protocols

1.4.1 TLS/SSL

Transport Layer Security (TLS)/Secure Socket Layer (SSL)

1.4.2 HTTPS

Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS)

1.4.3 SSH

Secure Socket Shell (SSH)

1.5 Virtual Private Networks (VPN)

Virtual Private Network (VPN)

1.5.1 Tunneling Protocols

1.5.2 VPN use in P2P Contexts

CHAPTER 2

Cryptography

2.1 Introduction to Cryptography

Cryptography is the practice of developing and using coded algorithms to protect and obscure transmitted information so that it can only be read by those with permission and the ability to decrypt it.[1] Put differently, cryptography obscures communications so that unauthorized parties cannot access them.

In practice, cryptography is used mainly to transform messages into an unreadable format (known as *ciphertext*[1]) that can only be decrypted into a readable format (known as *plaintext*[1]) by the intended authorized recipient using a specific secret key.

The idea behind cryptography has coalesced around four main principles.

1. **Confidentiality**: Encrypted information can only be accessed by the person for whom it is intended and no one else.[1]
2. **Integrity**: Encrypted information cannot be modified in storage or in transit between the sender and the intended receiver without any alterations being detected.[1]
3. **Non-repudiation**: The creator or sender of encrypted information cannot deny their intention to send the information.[1]
4. **Authentication**: The identities of the sender and receiver, as well as the origin and destination of the information, are confirmed.[1]

2.2 Types of Cryptography

There are two main types of encryption that are in use today: symmetric cryptography and asymmetric cryptography. There are also hybrid cryptosystems that combine both.

2.2.1 Symmetric-Key Cryptography

Symmetric-key cryptography uses a shared single key for both encryption and decryption. Both the sender and receiver of an encrypted message will have access to the same secret key.[1] Algorithms such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are symmetric systems.

Some main attributes of symmetric encryption include:

- **Speed**: The encryption process is comparatively fast.[1]
- **Efficiency**: Single-key encryption is well suited for large amounts of data and requires fewer resources.[1]
- **Confidential**: Symmetric encryption effectively secures data and prevents anyone without the key from decrypting the information.[1]

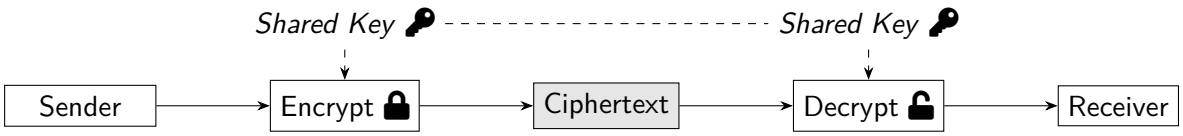


Figure 2.1: Symmetric Cryptosystem

2.2.2 Asymmetric-Key Cryptography

Asymmetric-key cryptography (also referred to as *public-key cryptography*) uses one private key and one public key. Data that are encrypted with a public and private keys require both the public key and the recipient's private key to be decrypted.[1]

Public-key cryptography enables secure key exchange over an insecure medium without the need to share a secret decryption key because the public key is only used in the encryption, but not the decryption process.[1] Asymmetric encryption adds another layer of security because an individual's private key is never shared. Rivest–Shamir–Adleman (RSA) is one of the most common public key encryption algorithms.

Some main attributes of asymmetric encryption include:

- **Security**: Asymmetric encryption is considered more secure.[1]
- **Robust**: Public-key cryptography offers more benefits, providing confidentiality, authenticity and non-repudiation.[1]
- **Resource intensive**: Unlike single key encryption, asymmetric encryption is slow and requires greater resources, which can be prohibitively expensive in some cases.[1]

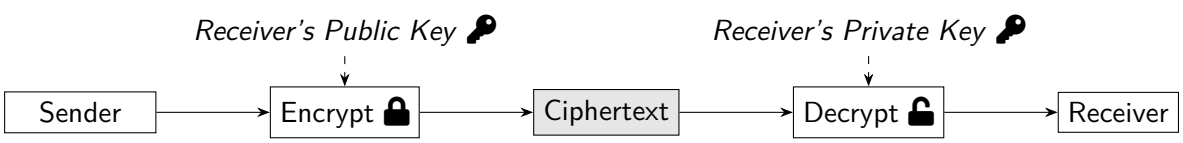


Figure 2.2: Asymmetric Cryptosystem

2.2.3 Properties of a Secure Cipher

In cryptography, **confusion** and **diffusion** are two properties of the operation of a secure cipher.[2] Both of these properties are needed to prevent the deduction of the secret by applying statistics or other forms of cryptanalysis.[2]

Confusion

Confusion refers to the idea of obscuring the relationship between the encryption key and the ciphertext.[2] Confusion is achieved when each resultant bit of the cipher depends upon many bits of the key.[2]

Diffusion

Diffusion refers to the idea of information of the plaintext being evenly spread across the ciphertext. This disallows the occurrence of statistical text patterns in the cipher, which could otherwise be used to deduce information.[2]

2.3 Key Exchange Mechanisms

Before moving onto encryption algorithms, we must first understand how shared secrets are established between the sender and the recipient over a public (insecure) channel. The commonly used key exchange mechanism is **Diffie–Hellman**.

2.3.1 Diffie–Hellman

Diffie–Hellman key exchange is a mathematical method for securely exchanging cryptographic keys over an insecure channel. It was developed by Whitfield Diffie and Martin Hellman in 1976.

Suppose Alice and Bob want to exchange a cryptographic key with each other. The algorithm works as follows:[3]

- Alice and Bob agree on a large prime number p and a generator g which is a *primitive root*[1] modulo p.
- Alice chooses a secret integer a and sends Bob $A \coloneqq g^a \bmod p$.
- Bob chooses a secret integer b and sends Alice $B \coloneqq g^b \bmod p$.
- Alice computes $A^b \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p$.
- Bob computes $B^a \bmod p = (g^b)^a \bmod p = g^{ab} \bmod p$.

The result of the equations is their shared secret $K \coloneqq g^{ab} \bmod p$ which can be used as an encryption key. The protocol is illustrated in fig. 2.3.

[1] An integer g is a primitive root modulo n if for every integer a relatively prime to n there exists an integer z such that $a \equiv g^z \pmod{n}$.

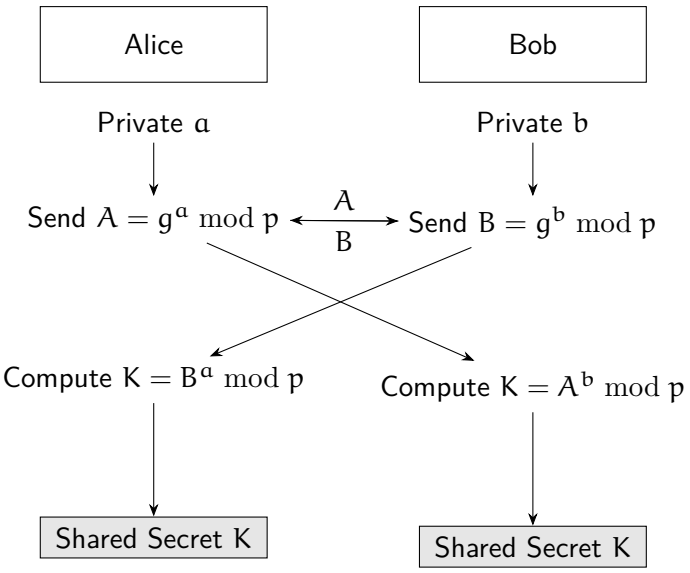


Figure 2.3: Diffie–Hellman key exchange

Why Diffie–Hellman is secure

Diffie–Hellman is secure because it relies on the difficulty of the *discrete logarithm* problem,[4] i.e., for known values of g, p and A, it is difficult[2] to obtain the exponent a that satisfies $g^a \equiv A \pmod{p}$, without additional information.[4]

The above algorithm can be summarised as follows:

Algorithm 1: Diffie–Hellman Key Exchange

Input: Public parameters: prime p, generator g
Output: Shared secret key K
Alice: Choose secret $a \in \{1, \ldots, p-1\}$
Compute $A \leftarrow g^a \bmod p$
Send A to Bob
Bob: Choose secret $b \in \{1, \ldots, p-1\}$
Compute $B \leftarrow g^b \bmod p$
Send B to Alice
Alice: Compute $K_A \leftarrow B^a \bmod p$
Bob: Compute $K_B \leftarrow A^b \bmod p$
return $K = K_A = K_B$

2.4 Symmetric Encryption Algorithms

In symmetric encryption, both the sender and recipient use the same shared key, which is usually established using Diffie–Hellman (§2.3.1). There are two types of symmetric encryption algorithms: **Block ciphers** and **stream ciphers**.

Block cipher Data is encrypted and decrypted in fixed-size blocks. If a message is larger

[2]Here, "difficult" means having no known polynomial time solution.

than the block size, it is divided into sequential blocks and processed one at a time.[5]

Stream cipher Data is encrypted bit by bit or byte by byte, designed for real-time encryption and transmission of data streams.[5]

2.4.1 One-time pad (OTP)

One-time pad (OTP) is a simple symmetric **stream cipher** encryption technique. It requires the use of a **single-use**, **pre-shared** key K that is larger than[3] or equal to the size of the message M being sent.

The algorithm works by using a binary representation of the message M and key K. We use the binary `XOR` ($\oplus$) operation to generate our ciphertext C.

Key generation

Suppose the message M is a bitstring of length n. OTP is *provably unbreakable* if the generated key meets the following conditions:

- The key must be **truly random** and must be generated by a **non-deterministic**, **non-repeatable** process. If the key is generated by an algorithm, it will not work.
- The key **must never be reused**. Use of the same key to encrypt different messages, no matter how trivially small, compromises the cipher.
- The key must be kept **completely secret** by the communicating parties.

Algorithm 2: OTP Key Generation

Input: Message M of length n
Output: Key K of length n
Generate a truly random bitstring K of length n
return K

Encryption

Suppose the message M and key K have the same length of n bits. The ciphertext C will also be of n bits and it is given as follows:

$$\forall i \in [\![1, n]\!],\ C_i = M_i \oplus K_i$$

E.g. consider $M \coloneqq 1011001001$ and $K \coloneqq 0010110101$. C is given by $M \oplus K$.

$$\begin{array}{rl} & 1011001001 \\ \oplus & 0010110101 \\ \hline & 1001111100 \quad \rightarrow C \end{array}$$

[3] If the key K is longer than the message M, then random bits may be appended at the end of M to match the length.

Algorithm 3: OTP Encryption

Input: Message M, key K

Output: Ciphertext C

for $i \leftarrow 1$ **to** n **do**

  $C_i \leftarrow M_i \oplus K_i$

return C

Decryption

OTP is what is called a *reciprocal cipher*, i.e., the same transformation is used to decrypt the message as the one used to encrypt it. Therefore,

$$\forall i \in [\![1, n]\!],\ M_i = C_i \oplus K_i$$

E.g. consider the previous key K = 0010110101 and the obtained ciphertext C = 1001111100.

$$\begin{array}{rl} & 1001111100 \\ \oplus & 0010110101 \\ \hline & 1011001001 \quad \rightarrow M \end{array}$$

Algorithm 4: OTP Decryption

Input: Ciphertext C, key K

Output: Decrypted message M

for $i \leftarrow 1$ **to** n **do**

  $M_i \leftarrow C_i \oplus K_i$

return M

The reciprocal cipher works because $C \oplus K = (M \oplus K) \oplus K = M$.

2.4.2 Advanced Encryption Standard (AES)

The Rijndael cipher, or the AES, is a symmetric **block cipher** algorithm that uses 128, 192 or 256-bit keys to transform a message block of 128 bits into 128 bits of ciphertext.

A few terminologies need to be defined to understand AES.

State

The encryption algorithm inputs 128-bit (16 bytes) blocks of the plaintext. These 16 bytes are represented as a 4×4 matrix called a state.

E.g. consider a plaintext P represented as a string of bytes,

$$P \coloneqq s_{15}s_{14}s_{13}s_{12}s_{11}s_{10}s_9s_8s_7s_6s_5s_4s_3s_2s_1s_0$$

where $s_i \in \{0,1\}^8 \ \forall i \in [\![0,15]\!]$. Its corresponding state matrix is given as shown in fig. 2.4.

s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}

or

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Figure 2.4: AES state matrix

Key expansion

The AES algorithm applies N_r rounds of transformations to the plaintext during encryption, where N_r is a number determined by the size of the key K. Depending on the size of the key, there are three different variations of AES, as shown in table 2.1.

AES variation	Size of key K	Number of rounds (N_r)
AES-128	128 bits (16 bytes)	10
AES-192	192 bits (24 bytes)	12
AES-256	256 bits (32 bytes)	14

Table 2.1: AES variations

The number of four-word (128 bits) blocks in the key, N_k is obtained by dividing the key size by 32 (i.e., $N_k \coloneqq \text{Key size} \div 32$). The number of rounds N_r is obtained by the formula $N_r \coloneqq N_k + 6$[4].

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Transformations

There are four transformations applied in AES:

[4]The $+6$ is a design constant chosen by the AES designers. It is derived from empirical cryptanalysis as opposed to a mathematical formula.

1. **Substitute Bytes**: Each byte s_{ij} in the state matrix is with a SubByte $s'_{ij} \coloneqq S(s_{ij})$ from a lookup table, using an 8-bit substitution box. This S-box used is derived from the multiplicative inverses over the Galois field[5] $\mathrm{GF}(2^8)$, or $\mathbb{F}_{256}$. This done to remove bit-level similarities inside each byte.

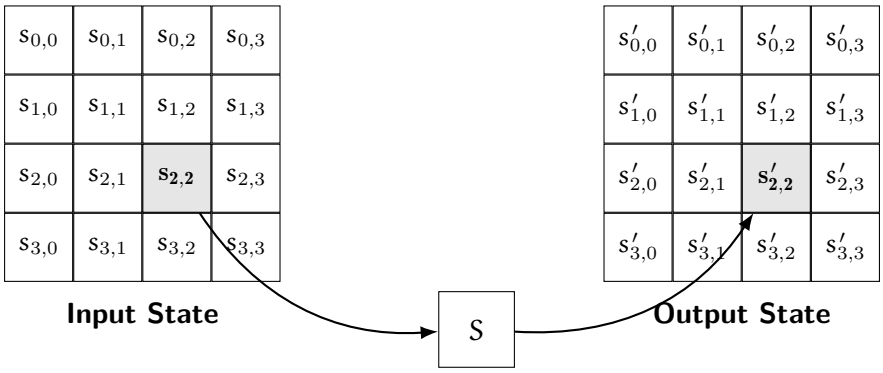


Figure 2.5: SubBytes

   The inverse substitution applied during decryption is denoted as InvSubBytes.

2. **Shift Rows**: Here, the rows of the state are *cyclically permuted*[6] according to the following rule:
   - The first row is *unchanged* $\left(s_{0,j} \mapsto s_{0,j}\right)$.
   - The second row is shifted *one position* to the left $\left(s_{1,j} \mapsto s_{1,(j-1) \bmod 4}\right)$.
   - The third row is shifted *two positions* to the left $\left(s_{2,j} \mapsto s_{2,(j-2) \bmod 4}\right)$.
   - The fourth row is shifted *three positions* to the left $\left(s_{3,j} \mapsto s_{3,(j-3) \bmod 4}\right)$.

   The inverse row shift applied during decryption is denoted as InvShiftRows.

3. **Mix Columns**: The four bytes of each column of a state are combined using an *invertible linear transformation*[7] with a fixed matrix.

   E.g., a column vector $\mathbf{s}_j$ is transformed to the vector $\mathbf{s}'_j$ using a transformation illustrated in eq. (2.1).

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} \coloneqq \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} \tag{2.1}$$

   Since the state bytes are in $\mathrm{GF}(2^8)$, this transformation is done modulo a polynomial $1 + x^4$. However, this shit is going over my head as well so I will leave this as an exercise to the reader.

[5] A Galois field (GF) is a field with a finite number of elements. A finite field $\mathbb{F}_n$ equals a ring $\mathbb{Z}_n$ when n is prime.

[6] Cyclic permutation refers to the arrangement of a set of objects around a fixed circle. Here, we use modular arithmetic to emulate a circular shift.

[7] Symbolically, a linear transformation refers to the multiplication of a matrix T with a vector $\mathbf{x}$ to produce another vector $\mathbf{y}$. It is said to be *invertible* iff there exists another matrix T^{-1} s.t. $\mathbf{x} = T^{-1}\mathbf{y}$.

MixColumns is used to add diffusion to the cipher. Unlike the other three transformations, MixColumns is dropped in the last round to provide a symmetric structure to the algorithm, else an additional inverse transformation (InvMixColumns) would be needed during decryption.

4. **Add Round Key**: Lorem ipsum

Encryption

During encryption, the transformations SubBytes, ShiftRows, MixColumns and addition of the round key are all performed sequentially on the state matrix for each round except the last one. In the last round, all transformations except MixColumns are performed in the same sequence.

Algorithm 5: AES Encryption

Input: Plaintext block P, cipher key K
Output: Ciphertext block C
Expand key K into round keys $K_0, K_1, \ldots, K_{N_r}$
$S \leftarrow P$
$S \leftarrow S \oplus K_0$
for $r \leftarrow 1$ **to** $N_r - 1$ **do**
    $S \leftarrow \text{SubBytes}(S)$
    $S \leftarrow \text{ShiftRows}(S)$
    $S \leftarrow \text{MixColumns}(S)$
    $S \leftarrow S \oplus K_r$
$S \leftarrow \text{SubBytes}(S)$
$S \leftarrow \text{ShiftRows}(S)$
$S \leftarrow S \oplus K_{N_r}$
return $C \leftarrow S$

Decryption

During decryption, the final round key is added followed by the inverse transformations InvShiftRows and InvSubBytes to the cipher for the first round. For the subsequent rounds, addition of the round key, InvMixColumns, InvShiftRows and InvSubBytes are performed sequentially.

Algorithm 6: AES Decryption

Input: Ciphertext block C, cipher key K
Output: Plaintext block P
Expand key K into round keys $K_0, K_1, \ldots, K_{N_r}$
$S \leftarrow C$
$S \leftarrow S \oplus K_{N_r}$
$S \leftarrow \text{InvShiftRows}(S)$
$S \leftarrow \text{InvSubBytes}(S)$
for $r \leftarrow N_r - 1$ **to** 1 **do**
    $S \leftarrow S \oplus K_r$
    $S \leftarrow \text{InvMixColumns}(S)$
    $S \leftarrow \text{InvShiftRows}(S)$
    $S \leftarrow \text{InvSubBytes}(S)$
$S \leftarrow S \oplus K_0$
return $P \leftarrow S$

Initialization vector

Initialization vector (IV)Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

2.5 Asymmetric Encryption Algorithms

Asymmetric encryption algorithms use a pair of keys: a public key for encryption and a private key for decryption. The most important asymmetric cryptosystems are RSA and ECC.

2.5.1 Rivest–Shamir–Adleman (RSA)

RSA is an asymmetric encryption algorithm devised by the three scientists Rivest, Shamir and Adleman. It is one of the most common algorithms used in modern cryptography.

Idea behind RSA

RSA relies on the difficulty of the *integer factorization problem*, i.e., it is difficult to factorize a large integer into a product of its factors on a *classical* (non-quantum) computer.

An algorithm that efficiently factors an arbitrary integer would render RSA-based public-key cryptography **insecure**.

Key generation

We first choose two arbitrary large prime numbers p and q. We obtain a semiprime modulus n as the product of p and q. All of our operations will be performed *modulo* n, i.e., in the ring $\mathbb{Z}_{\mathrm{n}}$.

$$\mathrm{n} = \mathrm{p} \times \mathrm{q}$$

We obtain the *totient*[8] of n via *Euler's totient function* φ, i.e. $\varphi(\mathrm{n})$.

Now, the **encryption exponent** e is chosen such that $1 < e < \varphi(\mathrm{n})$ and it is **relatively prime** to the totient $\varphi(\mathrm{n})$.

The **decryption exponent** d is obtained as the **modular inverse** of e modulo $\varphi(\mathrm{n})$, i.e.

$$\mathrm{d} \equiv e^{-1} \pmod{\varphi(\mathrm{n})}$$

The public key is denoted as the tuple (n, e)[9] and the private key is the decryption exponent d.

The above algorithm can be summarized as follows:

Algorithm 7: RSA Key Generation

Input: Two large prime numbers p, q
Output: Public key (e, n) and private key d
$\mathrm{n} \leftarrow \mathrm{p} \cdot \mathrm{q}$
$\varphi \leftarrow (\mathrm{p}-1)(\mathrm{q}-1)$
Choose e such that $1 < e < \varphi$ and $\gcd(e, \varphi) = 1$
$\mathrm{d} \leftarrow \textsc{ModInverse}(e, \varphi)$
return (n, e), d

Key distribution

For a sender to securely send a message to a recipient, the sender must know the recipient's public key to encrypt the message, and the recipient must use its private key to decrypt the message.

To enable the sender to send its encrypted messages, the recipient transmits its public key (n, e) to the sender through a reliable but not necessarily secret route. The private key is never transmitted.

The above protocol is illustrated in fig. 2.6.

[8] The totient of a positive integer n is the number of integers between 1 and n relatively prime to n.
[9] Some authors may denote the public key as (e, n), but that has no practical impact on the algorithm.

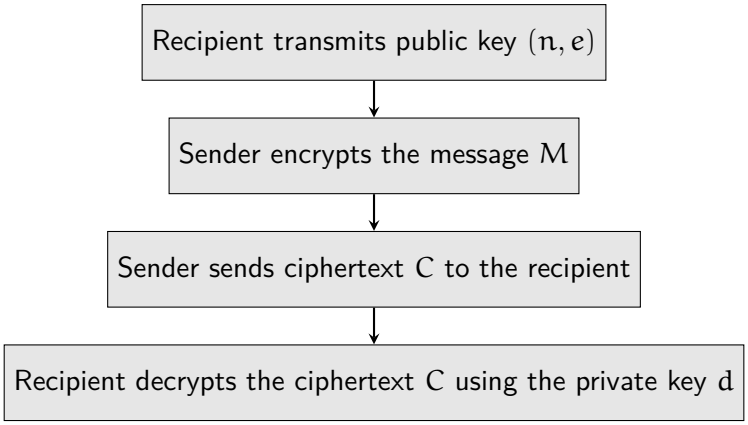


Figure 2.6: RSA key distribution

Encryption

A message M (where $M \in \mathbb{Z}$) can be converted to a ciphertext $C \in \mathbb{Z}_n$ by raising M to the encryption exponent e modulo n.

$$C \equiv M^e \pmod{n}$$

Algorithm 8: RSA Encryption

Input: Message M, public key (e, n)
Output: Ciphertext C
$C \leftarrow M^e \bmod n$
return C

Since most messages are text, typically ASCII or Unicode encoding is used to convert text to a corresponding numeric form before encryption by the sender.

Decryption

At the recipient, the private key d is used to decipher the received ciphertext C. The original message is recovered by raising C to the decryption exponent d.

The reason this works is that the decryption exponent d is the *modular inverse* of the encryption exponent e. Therefore,

$$\begin{aligned} & e \times d \equiv 1 \pmod{\varphi(n)} \\ \implies & e \times d = k\varphi(n) + 1 \text{ for some } k \in \mathbb{Z} \end{aligned} \tag{2.2}$$

Consider the following cases:

Case 1 $\gcd(M, n) = 1$. Then, by *Euler's theorem*[10],

$$\begin{aligned} M^{\varphi(n)} &\equiv 1 \pmod{n} \\ \implies M^{k\varphi(n)} &\equiv 1^k \equiv 1 \pmod{n} \\ \implies M^{k\varphi(n)+1} &\equiv M \pmod{n} \\ \therefore M^{e\times d} &\equiv M \pmod{n} \qquad \text{(From eq. (2.2))} \\ \implies (M^e)^d &\equiv \boxed{C^d \equiv M \pmod{n}} \end{aligned}$$

Case 2 $\gcd(M, n) \neq 1$. This implies that $M \equiv 0 \pmod{p}$ or $M \equiv 0 \pmod{q}$.

Without loss of generality, assume $M \equiv 0 \pmod{p}$. So by the *Chinese remainder theorem*[11],

$$M^{e\times d} \equiv \boxed{C^d \equiv M \pmod{n}}$$

The proof of the above result is left as an exercise to the reader.

Algorithm 9: RSA Decryption

Input: Ciphertext C, private key (n, d), modulus n

Output: Decrypted message M

$M \leftarrow C^d \bmod n$

return M

After decryption, the message M is typically converted back to text using the encoding method used by the sender.

2.5.2 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC)

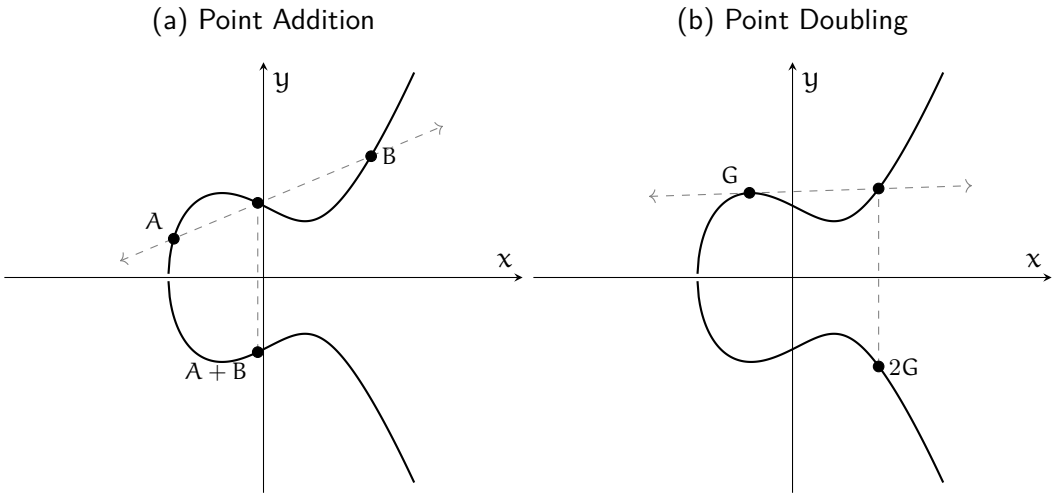


Figure 2.7: The elliptic curve $y^2 = x^3 + ax + b$

[10] If $\gcd(a, n) = 1$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

[11] If $\gcd(n_1, n_2) = 1$, then the system $x \equiv a_1 \pmod{n_1}$ and $x \equiv a_2 \pmod{n_2}$ has a unique solution modulo $n_1 n_2$, where $a_1, a_2 \in \mathbb{Z}$.

Algorithm 10: ECC Key Generation

Input: Elliptic curve E over field $\mathbb{F}_q$, base point G, order n

Output: Public-private key pair (d, Q)

Choose random integer $d \in [\![1, n-1]\!]$

Compute public key $Q \leftarrow d \cdot G$

return *Private key* d*, public key* Q

Algorithm 11: ECC Encryption (ElGamal-style)

Input: Message M, recipient's public key Q, curve E, base point G

Output: Ciphertext pair (C_1, C_2)

Encode M as a point P_M on E

Choose random integer $k \in [\![1, n-1]\!]$

$C_1 \leftarrow k \cdot G$

$C_2 \leftarrow P_M + k \cdot Q$

return (C_1, C_2)

Algorithm 12: ECC Decryption

Input: Ciphertext (C_1, C_2), private key d

Output: Recovered message M

Compute $P_M \leftarrow C_2 - d \cdot C_1$

Decode point P_M to get message M

return M

2.6 Hash Functions

2.7 Hashing Techniques

2.7.1 Message-Digest (MD)

Message-Digest (MD)

2.7.2 Secure Hashing Algorithm (SHA)

Secure Hashing Algorithm (SHA)

2.8 Digital Signatures

2.8.1 RSA and DSA signatures

Digital Signature Algorithm (DSA)

2.8.2 ECDSA

Elliptic Curve Digital Signature Algorithm (ECDSA)

2.9 Post-Quantum Cryptography (PQC)

Post-Quantum Cryptography (PQC)

2.9.1 Quantum Threat Model

2.10 Cipher Suites

2.10.1 Role of Cipher Suites

2.10.2 Components of Cipher Suites

2.10.3 Supported Cipher Suites

CHAPTER 3

Blockchain

3.1 Introduction to Blockchain

3.2 Architecture of Blockchain

3.2.1 Block Structure

3.2.2 Hashing and Merkle Trees

3.2.3 Chaining Mechanism

3.3 Consensus Mechanisms

3.3.1 Proof-of-Work

Proof-of-Work (PoW)

3.3.2 Proof-of-Stake

3.3.3 Other Variants (DPoS, PoA)

3.4 Transactions and Wallets

3.4.1 Transaction Format

Unspent Transaction Output (UTXO)

3.4.2 Digital Signatures and Validation

3.4.3 Wallet Types

3.5 Mining and Network Participation

3.6 Smart Contracts

3.6.1 Definition

3.6.2 Ethereum and Solidity

3.7 Blockchain Applications

3.7.1 Cryptocurrencies

3.7.2 Supply Chain, Voting, Identity

3.7.3 Decentralized Finance (DeFi)

Decentralized Finance (DeFi)

APPENDIX A

Quantum Computing

A.1 Introduction to Quantum Computing

A.1.1 Why Quantum Computing

A.1.2 Classical vs Quantum Computation

A.2 Quantum Bits (Qubits)

A.2.1 Superposition

A.2.2 Measurement

A.2.3 Entanglement

A.2.4 Bra-Ket Notation

$$\langle a|1\rangle$$

Ket Vectors

Bra Vectors

A.3 Quantum Gates and Circuits

A.3.1 Basic Gates

$$X, Z, H$$

A.3.2 Two-Qubit Gates

`CNOT`

A.3.3 Quantum Circuits and Reversibility

A.4 Quantum Algorithms

A.4.1 Deutsch–Jozsa Algorithm

A.4.2 Grover's Search Algorithm

A.4.3 Shor's Factoring Algorithm

List of Algorithms

Glossary

AES Advanced Encryption Standard 8, 12, 13, 15, 16, 26

CIA Confidentiality–Integrity–Availability 5

DDoS Distributed Denial of Service 5
DeFi Decentralized Finance 23
DES Data Encryption Standard 8
DoS Denial of Service 5
DSA Digital Signature Algorithm 20

ECC Elliptic Curve Cryptography 16, 19, 20, 26
ECDSA Elliptic Curve Digital Signature Algorithm 20

GF Galois field 14

HTTP Hypertext Transfer Protocol 6
HTTPS Hypertext Transfer Protocol Secure 6

IITB Indian Institute of Technology, Bombay 1
IV Initialization vector 16

MD Message-Digest 20

OTP One-time pad 11, 12, 26

P2P Peer-to-Peer 5
PoW Proof-of-Work 22
PQC Post-Quantum Cryptography 21

RRG Rupak R. Gupta 1
RSA Rivest–Shamir–Adleman 8, 16–19, 26

SHA Secure Hashing Algorithm 20

SSH Secure Socket Shell 6
SSL Secure Socket Layer 6

TLS Transport Layer Security 6

UTXO Unspent Transaction Output 22

VJTI Veermata Jijabai Technological Institute 1
VPN Virtual Private Network 6

References

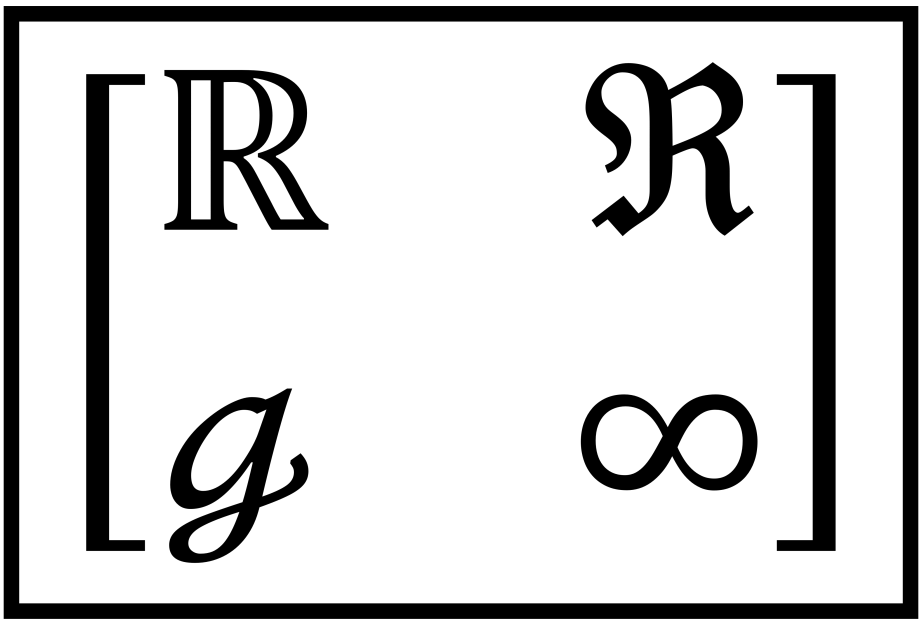
[1] What Is Cryptography? | IBM, November 2023.

[2] Spanning Tree. AES: How to Design Secure Encryption, August 2023.

[3] Computerphile. Secret Key Exchange (Diffie-Hellman) - Computerphile, December 2017.

[4] PurpleMind. The Simple Brilliance of Modern Encryption, March 2025.

[5] What is Stream Cipher and Block Cipher? | Encryption Consulting, March 2024. Section: Common Encryption Algorithms.


$\left[\begin{matrix}\mathbb{R} & \mathfrak{R} \\ g & \infty\end{matrix}\right]$

The End.