Textbook for DI03016021

Theories & Fundamentals
Subject Code: DI03016021

Milav Dabgar

July 1, 2026

Textbook for DI03016021

First Edition: 2026

Author: Milav Dabgar
Website: milav.in
YouTube: @milav.dabgar

To my students,
who constantly inspire me to find new analogies,
break down complex theories, and make learning
an engineered science.

Contents

List of Figures

List of Tables

Preface

The true power of the internet is not in connecting computers, but in connecting the physical world around us.

About this Book

This textbook is meticulously designed to align with the **GTU Diploma Syllabus (4353201)** for Wireless Sensor Networks and IoT. It provides a robust, intuitive, and comprehensive foundation in the rapidly evolving fields of sensor technologies, embedded systems, and the Internet of Things. Bridging the gap between theoretical network protocols and practical hardware implementations, this book decodes complex architectures into accessible, real-world analogies and engaging visual diagrams.

Target Audience

This book is specifically crafted for Diploma engineering students in the Information and Communication Technology (ICT) branch, as well as allied fields. It serves as an essential guide to demystifying the complexities of hardware-software integration, empowering students to build and understand the next generation of smart infrastructure.

Scope and Coverage

The coverage is strategically divided into the five core units of the official syllabus:

- **Unit 1: Introduction to WSN** – Exploring the fundamental building blocks of sensor nodes, network topologies, and the critical design principles prioritizing energy efficiency.
- **Unit 2: MAC and Routing Protocols** – Navigating the intricate layers of communication, addressing collision avoidance, duty cycling, and intelligent geographic data routing.
- **Unit 3: Overview of IoT** – Understanding the conceptual framework, reference architectures, and the foundational technologies (RFID, Big Data, Cloud) driving the IoT revolution.
- **Unit 4: Architecture and Implementation of IoT** – A deep dive into practical implementation, covering sensor integration, standard protocols (MQTT, CoAP), prototyping boards (NodeMCU, Raspberry Pi), and crucial security challenges.
- **Unit 5: IoT Applications** – Exploring transformative real-world use cases, from Smart Parking and Healthcare monitoring to Smart City infrastructure.

Milav Dabgar

Acknowledgments

Writing a comprehensive book that connects the fundamental forces of Chemistry with practical engineering applications requires more than just academic knowledge—it requires a community of support. I would like to express my sincere gratitude to everyone who contributed to the realization of this project.

Specially, I extend my heartfelt gratitude to the **Department of Technical Education (DTE), Government of Gujarat**, and **Government Polytechnic, Palanpur**, for providing an environment that fosters innovation, academic rigor, and continuous professional development.

I am deeply thankful to my family for their unwavering patience and support during the countless hours spent researching, formatting, and refining these chapters.

Finally, my deepest appreciation goes to my students. Your curiosity, challenging questions, and continuous feedback are the true driving force behind the unique analogies and streamlined structure of this book. This textbook was engineered for you.

Milav Dabgar

About the Author

Milav Jayeshkumar Dabgar is an engineering educator and R&D professional with over 9 years of experience in electronics hardware development, embedded systems, AI/ML, and full-stack software engineering. He currently serves as a **Lecturer (Class - II) & Innovation Leader** at Government Polytechnic, Palanpur, under the Education Department of the Government of Gujarat.

Milav holds a Master of Engineering (ME) in Communication Systems from L.D. College of Engineering and a Bachelor of Engineering (BE) in Electronics & Communication. He is also currently pursuing a **BS in Data Science and Applications from IIT Madras**, where he has completed both the **Diploma in Programming** and the **Diploma in Data Science** with distinction.

Most recently, he completed the **AICTE QIP PG Certificate Program in Deep Learning from SVNIT Surat** with a **perfect 10/10 CGPA**, topping his batch.

A dedicated mentor, Milav has guided numerous student teams in securing over **INR 45 lakhs** in innovation funding, including INR 25 lakhs from Shark Tank India and INR 20 lakhs through iHub Gujarat, resulting in two student patents. He is recognized as an **NPTEL Evangelist** and **NPTEL Discipline Star** in Computer Science, reflecting his commitment to continuous learning and academic excellence.

His technical expertise spans from embedded systems (8051, STM32, Arduino) to modern web technologies (Next.js, Python, PostgreSQL) and Data Science (TensorFlow, PyTorch). Through this textbook, he aims to simplify complex Engineering Chemistry concepts by integrating relatable, real-world analogies drawn from modern tech and engineering landscapes.

Milav is also an active content creator, running a popular **YouTube channel** (@milav.dabgar) where he shares technical tutorials and academic resources. He maintains a personal blog and resource hub at milav.in and can be reached for professional collaborations on LinkedIn.

CHAPTER 1

Introduction of Information & Network Security

1.1 Computer, Cyber, Information & Network Security

1.1.1 Defining the Security Landscape

As society digitizes everything from bank accounts to medical records, security is no longer an optional feature; it is the fundamental requirement of modern computing. - **Computer Security:** Protecting a single, isolated machine (its hardware, operating system, and local files) from unauthorized access or damage (e.g., locking your laptop with a password). - **Network Security:** Protecting the data *while it is moving* across a network from one computer to another (e.g., encrypting a file before sending it over the internet). - **Information Security:** The broadest term. Protecting the data itself, regardless of whether it is stored on a hard drive, transmitted over a network, or printed on a piece of paper in a filing cabinet. - **Cyber Security:** Protecting internet-connected systems (hardware, software, and data) from cyber-attacks initiated by hackers across the globe.

1.1.2 Essential Security Requirements (The CIA Triad)

Every security system ever designed is built to enforce one or more of these core requirements: 1. **Confidentiality:** Ensures that data is completely unreadable to anyone who is not authorized to see it. (e.g., Your bank balance should only be visible to you and the bank. If a hacker intercepts the data, it should look like gibberish). 2. **Integrity:** Ensures that data has not been secretly altered, corrupted, or destroyed during storage or transit. (e.g., If you transfer $10 to a friend, a hacker should not be able to alter the message mid-transit to make it $10,000). 3. **Availability:** Ensures that authorized users have uninterrupted, immediate access to the data whenever they need it. (e.g., The bank's website must not crash due to a hacker's DDoS attack; it must remain online). 4. **Authenticity:** Proving that the user or the sender is exactly who they claim to be (e.g., Verifying a user with a password and OTP). 5. **Accountability (Non-repudiation):** Ensuring that a user cannot mathematically deny performing an action. (e.g., Once a user clicks "Transfer Funds," the system logs irrefutable proof that they clicked it).

1.2 The OSI Security Architecture and Security Attacks

1.2.1 The OSI Security Architecture

To design a secure network, engineers follow the ITU-T X.800 standard (The OSI Security Architecture). This framework organizes security into three distinct categories: 1. **Security Attacks:** Any action that attempts to compromise the security of information. 2. **Security Mechanisms:** The actual mathematical algorithms (like encryption) or software tools (like firewalls) designed to detect, prevent, or recover from a security attack. 3. **Security Services:** The overall security goal (like Confidentiality or Authentication) provided by the system, which is achieved by implementing specific Security Mechanisms.

1.2.2 Security Attacks: Passive vs. Active

Hackers attack networks in two fundamentally different ways.

1. Passive Attacks

In a passive attack, the hacker does not alter the data and does not damage the system. Their only goal is to silently listen and steal information. These are incredibly dangerous because they are almost impossible to detect. - **Release of Message Contents:** The hacker eavesdrops on a network cable and reads an unencrypted email containing a password.

- **Traffic Analysis:** Even if the email is heavily encrypted and the hacker cannot read the text, they can analyze the *pattern* of the traffic. By noting that a military base suddenly started sending ten times more encrypted messages to headquarters, the hacker can deduce that a major military operation is about to happen, without ever decrypting a single word.

2. Active Attacks

In an active attack, the hacker aggressively alters data or disrupts the system. These are easier to detect but cause immediate, catastrophic damage. - **Masquerade:** The hacker steals an administrator's login credentials and logs in pretending to be them. - **Replay:** The hacker intercepts a legitimate, encrypted message (e.g., "Transfer $100 to account X") and silently re-sends that exact same encrypted message to the bank 50 times in a row, draining the victim's account. - **Modification of Messages:** The hacker intercepts a message in transit, changes the data, and forwards it to the destination (Integrity violation). - **Denial of Service (DoS):** The hacker commands thousands of infected computers to simultaneously send fake traffic to a web server, overwhelming its CPU and crashing the website, violating the *Availability* requirement.

1.3 Security Services

1.3.1 Fulfilling the Security Requirements

A **Security Service** is a processing or communication service provided by a system to provide a specific kind of protection to system resources. The X.800 architecture defines five major categories of security services.

1. **Authentication Service:** This service ensures that the communication is authentic. - *Peer Entity Authentication:* Confirms the identity of the user connecting to the system (e.g., proving you are actually Alice when logging into a bank). - *Data Origin Authentication:* Confirms that a specific piece of data actually came from the claimed sender, even if it is received hours later.

2. **Access Control Service:** Once a user is authenticated, this service dictates exactly what they are allowed to do. It restricts access to host systems and applications. (e.g., A student can log into the university database, but the Access Control service prevents them from modifying their own grades, a privilege reserved only for the professor).

3. **Data Confidentiality Service:** This service protects transmitted data from passive attacks (eavesdropping). It ensures that even if a hacker intercepts the data packet on the Wi-Fi network, they cannot read the contents. This is primarily achieved through mathematical encryption.

4. **Data Integrity Service:** This service protects against active attacks (modification). It ensures that messages are received exactly as they were sent, with no duplication, insertion, modification, reordering, or replays. If a single bit of the message is altered in transit, the system will immediately detect the corruption and reject the message.

5. **Nonrepudiation Service:** This prevents either the sender or the receiver from denying that they participated in a communication. - *Origin Nonrepudiation:* Proves mathematically that the sender actually sent the message (they cannot claim someone else sent it). - *Destination Nonrepudiation:* Proves mathematically that the receiver actually received the message (they cannot claim it was lost in the mail).

1.4 Security Mechanisms & Model for Network Security

1.4.1 Security Mechanisms

A security service (like Confidentiality) is just an abstract goal. To actually achieve that goal, we must use a **Security Mechanism** (a concrete tool or algorithm). The X.800 standard divides mechanisms into two types:

1. **Specific Security Mechanisms:** Tools implemented in a specific layer of the network to provide a specific service. - *Encipherment:* Using mathematical algorithms to transform readable data into unreadable ciphertext. (Provides Confidentiality). - *Digital Signatures:* Appending a unique mathematical cryptographic code to a document. (Provides Non-repudiation and Authentication). - *Data Integrity Controls:* Appending a checksum or hash value to a message to detect modification. (Provides Integrity).

2. **Pervasive Security Mechanisms:** Broad tools that apply universally across the entire system, not tied to any specific network protocol. - *Trusted Functionality:* Ensuring the operating system itself is secure. - *Security Labels:* Tagging files as "Top Secret" or "Unclassified" so the system knows how to handle them. - *Event Detection and Security Audit Trails:* Logging every single mouse click and login attempt into a permanent database so forensic investigators can figure out how a hacker broke in.

1.4.2 Model for Network Security

To design a secure communication system over the internet, we follow a general four-step model: 1. **Design an algorithm:** Create a mathematical algorithm for performing the security transformation (e.g., the AES encryption algorithm). 2. **Generate secret information:** Create the secret keys used by the algorithm (e.g., a 256-bit binary password). 3. **Distribute the secret:** Figure out a secure way to share that secret key between the sender and the receiver without the hacker stealing it. 4. **Specify a protocol:** Write the exact rules for how the sender and receiver will communicate using the algorithm and the keys.

1.5 Cryptography: Concept & Classification

1.5.1 The Science of Secrecy

Cryptography is the mathematical science of writing secret codes to secure communication in the presence of malicious third parties (adversaries). - **Plaintext:** The original, readable message (e.g., "ATTACK AT DAWN"). - **Ciphertext:** The scrambled, unreadable message (e.g., "XQZPM OQ ZTBM"). - **Encryption:** The mathematical process of converting Plaintext into Ciphertext. - **Decryption:** The mathematical process of converting Ciphertext back into Plaintext. - **Key:** A secret piece of information (usually a massive number) that tells the encryption algorithm exactly *how* to scramble the text.

1.5.2 Classification of Cryptographic Systems

Cryptographic algorithms are fundamentally classified into three distinct categories based on how they use "Keys."

1. **Key-less Cryptography (Hash Functions):** These algorithms do not use any secret keys at all. They take a file of any size (even a 10 GB movie) and run it through a one-way mathematical meat-grinder to produce a short, fixed-length string of random characters (a Hash). - *Property:* It is a "one-way" function. You cannot mathematically reverse a hash back into the original file. - *Use Case:* Checking file integrity and securely storing passwords.

2. **Single-Key (Symmetric) Cryptography:** The sender and the receiver use the **exact same secret key**. - *Process:* Alice locks a box with a physical key. She sends the box to Bob. Bob uses a duplicate copy of that exact same physical key to unlock the box. - *Advantage:* Mathematically very simple and incredibly fast. Used to encrypt massive amounts of data (like a hard drive). - *Disadvantage:* The "Key Distribution Problem." How does Alice securely give the duplicate key to Bob over the internet without a hacker stealing it?

3. **Two-Key (Asymmetric / Public-Key) Cryptography:** This was the greatest revolution in the history of cryptography. Every user generates **two different but mathematically linked keys**: A Public Key and a Private Key. - *Process:* The Public Key is published on the internet for the whole world to see. The Private Key is kept absolutely secret on the user's computer. If Alice wants to send a secret message to Bob, Alice uses Bob's *Public Key* to encrypt the message. Once encrypted with the Public Key, the message can *only* be decrypted by Bob's *Private Key*. - *Advantage:* Solves the Key Distribution problem completely. Two strangers can securely communicate over the internet without ever having met to exchange a secret key.

CHAPTER 2

Number Theory in Cryptography

2.1 Divisibility and Division Algorithm

2.1.1 The Mathematics of Secrets

Modern cryptography is not based on hiding letters; it is based on manipulating incredibly large numbers. To understand algorithms like RSA, we must study **Number Theory**, specifically the properties of integers.

2.1.2 Divisibility

An integer a is perfectly divisible by an integer b (where $b \neq 0$) if there is no remainder when a is divided by b. Mathematically, we write this as:

$$b|a$$

(Read as "b divides a"). This means there exists some integer k such that:

$$a = b \times k$$

Example: $4|12$ is true because $12 = 4 \times 3$. However, $5|12$ is false because division leaves a remainder.

2.1.3 The Division Algorithm

Despite its name, this is a mathematical theorem, not a computer algorithm. It states that if you divide any integer a by a positive integer n, you will get exactly one specific quotient (q) and one specific remainder (r).

Given any integer a and a positive integer n, there exist unique integers q and r such that:

$$a = q \times n + r$$

where the remainder r must be greater than or equal to zero, and strictly less than n ($0 \leq r < n$).

Example: If we divide $a = 17$ by $n = 5$:

$$17 = (3 \times 5) + 2$$

Here, the quotient $q = 3$ and the remainder $r = 2$.

2.2 Modulo Operator & Modular Arithmetic Properties

2.2.1 The Modulo Operator (Clock Arithmetic)

In cryptography, we are rarely interested in the quotient; we are obsessed with the remainder. The **modulo operator** (denoted as mod) mathematically extracts the remainder of a division operation.

$$a \bmod n = r$$

This means that when a is divided by n, the remainder is r. *Example:* $17 \bmod 5 = 2$.

This is often called "clock arithmetic." On a standard 12-hour clock, if it is currently 10:00 and you add 5 hours, it does not become 15:00. The clock resets at 12, so the time becomes 3:00. Mathematically: $(10 + 5) \bmod 12 = 15 \bmod 12 = 3$.

2.2.2 Properties of Modular Arithmetic

Modular arithmetic behaves similarly to standard arithmetic, but everything is wrapped around the modulus n. This wrapping property is what scrambles data in encryption algorithms.

1. **Addition Property:**

$$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$$

2. **Subtraction Property:**

$$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$$

Note on Negative Numbers: In cryptography, remainders must be positive. If $a = -4$ and $n = 5$, what is $-4 \bmod 5$? We keep adding the modulus (5) until the number becomes positive: $-4 + 5 = 1$. Therefore, $-4 \bmod 5 = 1$. 3. **Multiplication Property:**

$$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$$

These properties are critically important in computing because they allow a computer to perform massive calculations on huge numbers by calculating the modulo at every step, preventing the numbers from overflowing the computer's memory.

2.3 Set of Residues: Zn and Inverses

2.3.1 The Set of Residues (Z_n)

When we perform operations modulo n, the only possible answers are the remainders from 0 up to $n - 1$. This finite collection of numbers is called the **Set of Residues** and is denoted mathematically as Z_n.

$$Z_n = \{0, 1, 2, 3, ..., (n - 1)\}$$

For example, in modulo 5, the set is $Z_5 = \{0, 1, 2, 3, 4\}$. No matter how large numbers you add or multiply, the final answer in modulo 5 will always be one of these five numbers.

2.3.2 Inverses in Modular Arithmetic

In standard arithmetic, subtraction is the inverse of addition, and division is the inverse of multiplication. In modular arithmetic (which we use for decryption), things are more complex.

1. Additive Inverse

Two numbers in Z_n are additive inverses of each other if their sum, modulo n, equals 0.

$$(a + b) \bmod n = 0$$

To find the additive inverse of a in Z_n, you simply calculate $(n - a)$. *Example in Z_{10}:* The additive inverse of 3 is $(10 - 3) = 7$. Proof: $(3 + 7) \bmod 10 = 10 \bmod 10 = 0$.

2. Multiplicative Inverse (Z_n^*)

Two numbers in Z_n are multiplicative inverses of each other if their product, modulo n, equals 1.

$$(a \times b) \bmod n = 1$$

This is much harder to find. In standard arithmetic, the multiplicative inverse of 3 is 1/3 (a fraction). But in modular arithmetic, fractions do not exist! We must find a whole number b that satisfies the equation. *Example in Z_{10}:* What is the multiplicative inverse of 3? We must test numbers. - $(3 \times 1) \bmod 10 = 3$ - $(3 \times 2) \bmod 10 = 6$ - $(3 \times 7) \bmod 10 = 21 \bmod 10 = 1$ Therefore, in Z_{10}, the multiplicative inverse of 3 is 7.

Critical Rule: An integer a only has a multiplicative inverse in Z_n if a and n are **Relatively Prime** (meaning their Greatest Common Divisor is 1). The set of all numbers in Z_n that have a multiplicative inverse is denoted as Z_n^*.

2.4 Calculation of GCD and Multiplicative Inverse

2.4.1 1. Euclidean Algorithm for GCD

To determine if a number has a multiplicative inverse, we must find its Greatest Common Divisor (GCD). The GCD of two integers a and b is the largest integer that divides both of them without a remainder.

The **Euclidean Algorithm** is a fast, 2,000-year-old method for finding the GCD without factoring the numbers. It is based on the theorem that $\gcd(a, b) = \gcd(b, a \bmod b)$.

Example: Find $\gcd(2740, 1760)$ Step 1: Divide the larger by the smaller. $2740 = 1 \times 1760 + 980$ Step 2: Shift the divisor and remainder to the left. Divide 1760 by 980. $1760 = 1 \times 980 + 780$ Step 3: Repeat until the remainder is 0. $980 = 1 \times 780 + 200$ $780 = 3 \times 200 + 180$ $200 = 1 \times 180 + 20$ $180 = 9 \times 20 + 0$ When the remainder hits 0, the last non-zero remainder is the GCD. Therefore, $\gcd(2740, 1760) = 20$.

2.4.2 2. Extended Euclidean Algorithm

If $\gcd(a, n) = 1$, then a has a multiplicative inverse modulo n. We use the **Extended Euclidean Algorithm** to mathematically calculate that exact inverse. This algorithm is the absolute backbone of the RSA public-key encryption system (it is used to calculate the Private Key).

The Extended Euclidean Algorithm calculates the GCD, but also simultaneously calculates two integers x and y such that:

$$a \cdot x + n \cdot y = \gcd(a, n)$$

If $\gcd(a, n) = 1$, then the equation becomes $a \cdot x + n \cdot y = 1$. Taking the modulo n of both sides mathematically eliminates the $n \cdot y$ term (because any multiple of n is 0 in modulo n). This leaves:

$$(a \cdot x) \bmod n = 1$$

Therefore, the coefficient x calculated by the Extended Euclidean Algorithm is the exact multiplicative inverse of a modulo n.

CHAPTER 3

Classical Encryption Techniques

3.1 Symmetric-key Encryption & Cryptanalysis

3.1.1 The Symmetric Cipher Model

For centuries, governments and militaries used Symmetric-key Encryption exclusively. In this model, there are five essential ingredients: 1. **Plaintext:** The readable message. 2. **Encryption Algorithm:** The mathematical rule used to scramble the text. 3. **Secret Key:** The specific value input into the algorithm that controls exactly how the text is scrambled. The sender and receiver must possess the exact same secret key. 4. **Ciphertext:** The scrambled message produced as output. 5. **Decryption Algorithm:** The reverse of the encryption algorithm, which takes the ciphertext and the secret key to produce the original plaintext.

3.1.2 Cryptanalysis and Types of Attacks

Cryptanalysis is the science of breaking the ciphertext *without* knowing the secret key. The cryptanalyst relies on mathematical analysis, recognizing patterns in the ciphertext, and a deep understanding of the algorithm.

Common types of cryptanalytic attacks include: - **Ciphertext-only Attack:** The hacker only possesses the ciphertext. This is the hardest attack to execute because the hacker has the least amount of information. - **Known-plaintext Attack:** The hacker has a copy of the ciphertext AND the original plaintext for one specific message. They use this pair to mathematically deduce the secret key used, allowing them to decrypt all future messages.

3.1.3 The Brute-Force Attack

This is the ultimate, brute-strength method used by computers. A Brute-Force attack does not rely on clever mathematics; it relies on raw processing power. The computer simply attempts to decrypt the message using every single possible key in existence, one by one, until the output resembles readable text. - *Defense:* The only defense against a brute-force attack is to use a massive key size. If the key is 256 bits long, there are 2^{256} possible combinations. Even if a supercomputer tests 1 billion keys per second, it would take longer than the age of the universe to try them all.

3.2 Substitution Ciphers

A substitution cipher is a technique where letters in the plaintext are systematically replaced by other letters or numbers.

3.2.1 1. Mono-alphabetic Ciphers

In these ciphers, a specific letter in the plaintext is always replaced by the exact same specific letter in the ciphertext. If 'A' becomes 'D', then every 'A' in the message will be 'D'. - **Additive (Shift/Caesar) Cipher:** Every letter is shifted a fixed number of positions down the alphabet. (e.g., Key = 3: A $\rightarrow$ D, B $\rightarrow$ E). - **Multiplicative Cipher:** Instead of adding, the plaintext letter's numerical value is multiplied by a key modulo 26. - **Affine Cipher:** A combination of both. It uses two keys (k_1 and k_2). The encryption function is $C = (P \times k_1 + k_2) \bmod 26$. - *Vulnerability:* Mono-alphabetic ciphers are easily broken using **Frequency Analysis**. In English, the letter 'e' is used far more often than any other letter. A cryptanalyst simply counts the most frequent letter in the ciphertext and assumes it is 'e'.

3.2.2 2. Poly-alphabetic Ciphers

To defeat frequency analysis, poly-alphabetic ciphers use multiple substitution rules. The same letter 'A' in the plaintext might be encrypted as 'D' the first time, and 'X' the second time. - **Vigenère Cipher:** Uses a repeating keyword (e.g., "LEMON") and a mathematical grid (a Vigenère tableau) to shift different letters by different amounts. - **Playfair Cipher:** Encrypts pairs of letters (digraphs) simultaneously using a 5×5 matrix of letters constructed from a keyword, rather than encrypting single letters. - **Hill Cipher:** The most mathematically advanced classical cipher. It uses matrix multiplication and linear algebra to encrypt blocks of plaintext. It requires the secret key to be an invertible matrix modulo 26. - **One-Time Pad:** The only theoretically unbreakable cipher in existence. It requires a truly random key that is *exactly as long as the plaintext message itself*, and the key can never be reused.

3.3 Transposition Ciphers & Stream vs. Block Ciphers

3.3.1 Transposition Ciphers

While substitution ciphers replace letters, **Transposition Ciphers** merely rearrange (permute) the order of the letters in the plaintext without changing their identity. - **Keyless Transposition (Rail Fence):** The plaintext is written downwards diagonally on successive "rails" of an imaginary fence, then read off in rows. *(e.g., "HELLO" on 2 rails becomes HLOEL)*. - **Keyed Rectangular (Columnar) Transposition:** The plaintext is written out in rows of a fixed length forming a grid. The ciphertext is generated by reading down the columns. To make it secure, a numerical **Key** dictates the exact order in which the columns are read. - **Key Inversion:** To decrypt a keyed transposition cipher, the receiver must possess the key (the column order) and perform a key inversion to mathematically reverse the column permutations to reconstruct the original grid.

3.3.2 Stream vs. Block Ciphers

Modern symmetric encryption algorithms are categorized by how they process data. 1. **Stream Ciphers:** These encrypt data one bit or one byte at a time as a continuous stream. They are extremely fast and require very little memory, making them ideal for encrypting live video streams or voice calls. The encryption heavily relies on generating a pseudorandom "keystream" that is XORed with the plaintext. 2. **Block Ciphers:** These process data in massive chunks (blocks) of fixed sizes, usually 64 or 128 bits at a time. The algorithm waits until it has a full block of data, encrypts the entire block as a single unit using complex mathematical substitutions and permutations, and outputs a block of ciphertext. Almost all modern file encryption (like AES used by banks) uses block ciphers because they are vastly more secure than stream ciphers.

3.4 Asymmetric-key Cryptosystem & RSA Algorithm

3.4.1 The Ingredients of Public-Key Encryption

The invention of Public-Key (Asymmetric) Cryptography in 1976 solved the massive problem of distributing secret keys over the internet. The system relies on six ingredients: 1. **Plaintext:** The readable message. 2. **Encryption Algorithm:** Performs transformations on the plaintext. 3. **Public Key (PU):** A key made available to the entire world. Used for encryption. 4. **Private Key (PR):** A key kept absolutely secret by the owner. Used for decryption. 5. **Ciphertext:** The scrambled output. 6. **Decryption Algorithm:** Recovers the plaintext using the matching Private Key.

3.4.2 Confidentiality vs. Authentication

The beauty of Asymmetric cryptography is that it provides two entirely different services depending on how you use the keys. - **Confidentiality:** Alice wants to send a secret to Bob. Alice uses Bob's *Public Key* to encrypt it. Only Bob possesses his *Private Key*, so only Bob can decrypt it. The message is confidential. - **Authentication (Digital Signatures):** Alice wants to prove to Bob that she wrote a document. Alice uses *her own Private Key* to encrypt the document. Bob receives it and uses Alice's *Public Key* to decrypt it. Because Alice's Public Key successfully decrypted the file, Bob has mathematical proof that only Alice's Private Key could have encrypted it. This authenticates the sender.

3.4.3 The RSA Algorithm

Developed by Rivest, Shamir, and Adleman, RSA is the most widely used public-key algorithm in the world. It is based on the mathematical difficulty of factoring massive prime numbers.

RSA Key Generation Process: 1. Choose two very large, distinct prime numbers, p and q (often 2048 bits long). 2. Calculate the modulus: $n = p \times q$. This n will be published as part of the Public Key. 3. Calculate Euler's Totient function: $\phi(n) = (p-1) \times (q-1)$. 4. Choose an integer e (the public exponent) such that $1 < e < \phi(n)$ and e is relatively prime to $\phi(n)$. (Meaning $\gcd(e, \phi(n)) = 1$). 5. Calculate d (the private exponent) as the multiplicative inverse of e modulo $\phi(n)$. This uses the Extended Euclidean Algorithm: $(e \times d) \bmod \phi(n) = 1$. 6. **The Keys:** - Public Key = $\{e, n\}$ - Private Key = $\{d, n\}$

If a hacker intercepts the Public Key $\{e, n\}$, they cannot calculate the Private Key d unless they know $\phi(n)$. To calculate $\phi(n)$, they must factor the massive number n back into the original primes p and q. For a 4096-bit number, this would take the world's fastest supercomputers billions of years.

CHAPTER 4

Web Security

4.1 Web Security Considerations & Threats

4.1.1 The Vulnerability of the Web

The World Wide Web was originally designed by scientists in 1989 to easily share research documents across the globe. Security was not a priority. Today, the web facilitates trillions of dollars in global commerce, making it the primary target for organized cybercrime. Web security is uniquely difficult because a web server must deliberately remain open to the public internet to accept connections from strangers, creating a massive attack surface.

4.1.2 1. Web Security Threats

A web threat can compromise the server, the network, or the user's browser. - **Integrity Threats:** A hacker breaks into a corporate web server and defaces the homepage or silently modifies the source code to redirect users to a malicious phishing site. - **Confidentiality Threats (Eavesdropping):** If a user types their credit card into an HTTP website, a hacker sitting in a coffee shop can use a packet sniffer on the public Wi-Fi to steal the credit card number in plain text. - **Denial of Service (DoS):** Flooding the web server with massive amounts of fake traffic until it crashes, denying service to legitimate customers. - **Authentication Threats:** Stealing passwords via phishing, keyloggers, or brute-force dictionary attacks.

4.1.3 2. Web Traffic Security Approaches

To secure web traffic against these threats, security can be implemented at different layers of the OSI model: - **Network Level (IPSec):** Secures all traffic automatically between two routers or VPN gateways, regardless of the application being used. - **Transport Level (SSL/TLS):** This is the most common approach for the web. Security is implemented right above TCP. It encrypts the data stream before it is sent. (This is how HTTPS works). - **Application Level (S/MIME, SET):** Security is baked directly into the specific application. For example, an email client encrypts a specific email document before sending it, rather than encrypting the entire network connection.

4.2 SSL and TLS Protocol Stacks

4.2.1 Overview of the SSL Protocol Stack

Secure Socket Layer (SSL) was invented by Netscape in 1994 to encrypt web browsing. It operates at the Transport Layer (Layer 4) of the OSI model. It is not a single protocol, but a layered stack of two sub-protocols.

1. **The SSL Record Protocol:** This is the bottom layer, sitting directly on top of TCP. It provides two fundamental services: - *Confidentiality:* It encrypts the data using symmetric encryption (like AES). - *Message Integrity:* It appends a MAC (Message Authentication Code) to the data so the receiver can mathematically verify that the data wasn't corrupted or altered in transit.

2. **The SSL Management Protocols (Handshake, Change Cipher Spec, Alert):** These sit on top of the Record Protocol. The most critical is the **SSL Handshake Protocol**. Before any application data is sent, the client (web browser) and the server use the Handshake Protocol to: - Authenticate the server to the client using digital certificates (proving it is the real Amazon.com, not a fake). - Negotiate which cryptographic algorithms they will use (e.g., "Let's use AES-256 for encryption and SHA-256 for hashing"). - Securely generate and exchange the massive, shared Symmetric Secret Key that the Record Protocol will use to encrypt the actual web traffic.

4.2.2 Overview of the TLS Protocol Stack

Transport Layer Security (TLS) is simply the modern, upgraded, and highly secure successor to SSL. (In 1999, the IETF took over SSL version 3.0 and renamed it TLS 1.0). Today, SSL is mathematically obsolete and highly vulnerable to attacks (like POODLE). Modern browsers will refuse to connect to a server using SSL. However, the industry still colloquially refers to TLS as "SSL."

The TLS protocol stack is structurally identical to the SSL stack (Record Protocol below, Handshake Protocol above) but utilizes vastly stronger cryptographic algorithms and more secure key exchange methods (like Elliptic Curve Diffie-Hellman).

4.3 HTTPS Architecture

4.3.1 The Secure Web

HTTPS (Hypertext Transfer Protocol Secure) is not a new protocol. It is simply standard HTTP traffic running on top of a secure SSL/TLS connection. When you visit an ‘http://‘ website, data is transmitted as plain text over TCP port 80. When you visit an ‘https://‘ website, the browser connects to TCP port 443, establishes a TLS encrypted tunnel, and then sends the HTTP requests through that tunnel.

4.3.2 Connection Initiation (The Handshake)

When a user types ‘https://www.bank.com‘, a complex cryptographic dance occurs in milliseconds. 1. **Client Hello:** The browser sends a list of cryptographic algorithms it supports. 2. **Server Hello & Certificate:** The bank’s server selects the strongest algorithm they both support. Crucially, the server sends its **Digital Certificate** (issued by a trusted Certificate Authority like Verisign), containing the bank’s Public Key. 3. **Authentication:** The browser mathematically verifies the certificate signature to ensure it is actually communicating with the real bank. 4. **Key Exchange:** The browser generates a random "Pre-Master Secret," encrypts it using the bank’s Public Key, and sends it to the server. The server decrypts it with its Private Key. 5. **Symmetric Key Generation:** Both the browser and the server mathematically derive the exact same Symmetric Session Key from the Pre-Master Secret. 6. **Encrypted Tunnel Established:** The handshake is complete. All subsequent HTTP traffic (passwords, HTML, images) is encrypted using that Symmetric Session Key.

4.3.3 Connection Closure

An HTTPS connection must be closed securely. A hacker could theoretically inject a fake TCP "FIN" (Finish) packet into the network, prematurely terminating the connection and causing the browser to render an incomplete web-page, which might be a security risk. To prevent this, the TLS Record Protocol requires the sending of a specific encrypted ‘close$_n otify$‘$alert message. Because this message is encrypted with the session key, a hacker cannot forge it. Both parties know t$

4.4 Secure Electronic Transactions (SET) & SSL vs. SET

4.4.1 The Flaw in E-Commerce (Why SET was Invented)

In the late 1990s, e-commerce was terrifying. When a user bought a book on Amazon using SSL/HTTPS, the credit card number was safely encrypted during transit. However, once the encrypted data reached Amazon’s servers, Amazon decrypted it and stored the plain text credit card number in their database. If a hacker breached Amazon’s database, they stole millions of credit cards. To solve this massive liability, Visa and MasterCard developed **SET (Secure Electronic Transaction)** in 1996.

4.4.2 The Basic Concept of SET

SET was an extremely complex, application-level cryptographic protocol designed specifically to protect credit card payments over the internet. - **The Core Innovation (Dual Signatures):** In SET, the customer’s browser encrypts the order information for the Merchant, but it encrypts the Credit Card number directly for the Bank. - **The Process:** The customer sends the entire package to the Merchant. The Merchant can read the order (to ship the book) but the Merchant *cannot* mathematically decrypt or read the credit card number. The Merchant forwards the payment block to the Bank. The Bank decrypts the credit card, approves the charge, and sends an authorization code back to the Merchant. - **Result:** The Merchant never sees the credit card number, meaning a hacker cannot steal it from the Merchant’s database.

4.4.3 SSL versus SET

Despite being vastly superior in security, SET completely failed and is obsolete today.

Feature	SSL / TLS (HTTPS)	SET (Secure Electronic Transaction)
Scope	A general-purpose protocol. Secures any data (passwords, emails, web browsing).	A highly specialized protocol designed exclusively for credit card payments.
Credit Card Privacy	The Merchant can see and store the customer's plain text credit card number.	The Merchant CANNOT see the credit card number (Dual Signature architecture).
Complexity	Very simple. Built into every web browser automatically. The user does nothing.	Immensely complex. Required the customer to install special software and obtain a personal digital certificate from their bank to buy a $5 book.
Outcome	Won the war due to extreme simplicity and universal browser support.	Failed completely because it was too complicated for average consumers and merchants to configure.

Table 4.1: Comparison of SSL and SET

CHAPTER 5

System Security

5.1 Intrusion Detection Systems (IDS)

5.1.1 The Inevitability of Breach

No cryptographic algorithm or firewall is 100% secure. A hacker will eventually bypass the perimeter defenses. When this happens, a system must rely on **Intrusion Detection Systems (IDS)**. An IDS is like a silent burglar alarm; it constantly monitors network traffic or system logs looking for suspicious activity, alerting the system administrator if a breach occurs.

5.1.2 1. Classification of Intruders

Intruders are classified into three distinct categories based on their origin and authorization level: 1. **Masquerader (The Outsider):** An individual who is completely unauthorized to use the computer but penetrates the system by stealing a legitimate user's password. (e.g., A Russian hacker logging into a US Bank server). 2. **Misfeasor (The Insider):** A legitimate, authorized user who abuses their privileges by accessing data they shouldn't, or a user who has normal access but finds a way to upgrade themselves to an Administrator role. (e.g., A low-level bank teller secretly looking at the CEO's bank account). 3. **Clandestine User:** Either an insider or an outsider who hacks deep into the operating system and actively deletes the system's audit logs to cover their tracks, acting as a "ghost" in the machine.

5.1.3 2. Approaches to Intrusion Detection

An IDS software uses two entirely different philosophies to catch hackers: - **Misuse Detection (Signature-Based):** The IDS maintains a massive database of "signatures" (known patterns of famous viruses or hacking tools). It compares all incoming traffic against this database, similar to an antivirus scan. *Advantage:* Extremely fast and produces very few false alarms. *Disadvantage:* It can only catch *known* attacks. If a hacker writes a brand new, zero-day virus, the Misuse IDS will completely ignore it. - **Anomaly Detection (Behavior-Based):** The IDS uses AI and statistics to study the network for a month to learn what "normal" behavior looks like (e.g., Alice usually logs in at 9 AM from New York and downloads 5 MB of data). If Alice suddenly logs in at 3 AM from China and starts downloading 500 GB of classified files, the IDS flags this as a massive anomaly and shuts down the connection. *Advantage:* Can catch brand-new, never-before-seen zero-day attacks. *Disadvantage:* Extremely prone to "False Positives" (alerting the administrator just because Alice decided to work late from a hotel).

5.2 Firewalls: Definition, Need, and Characteristics

5.2.1 The Perimeter Defense

Before connecting a highly sensitive corporate network to the wild, unregulated internet, a security barrier must be erected. A **Firewall** is a dedicated hardware appliance or software program inserted between the internal corporate network and the external internet. Its sole purpose is to act as a choke point: it mathematically analyzes every single packet of data trying to enter or leave the building, comparing it against a strict set of security rules to decide if it should be allowed through or dropped.

5.2.2 The Need for Firewalls

- Without a firewall, every single computer in a university network is directly exposed to the internet. If one student's computer has a vulnerability, a hacker in another country can directly scan it and exploit it. - A firewall acts as a shield, hiding the internal IP addresses of the computers from the outside world. - It provides a centralized point

to implement security policies (e.g., "Block all employees from accessing Facebook during work hours" or "Block all incoming traffic from IP addresses located in North Korea").

5.2.3 Characteristics of a Firewall

For a firewall to be effective, three critical characteristics must be met: 1. **Single Choke Point:** ALL traffic flowing from the inside to the outside, and vice versa, *must* physically pass through the firewall. If an employee secretly plugs a 4G wireless modem into their office desktop, they create an unprotected backdoor that bypasses the firewall entirely. 2. **Authorized Traffic Only:** Only traffic authorized by the local security policy (defined by the administrator) is allowed to pass. Everything else is dropped by default. 3. **Immunity to Penetration:** The firewall itself must be running on a highly trusted, mathematically hardened operating system. If the hacker can hack the firewall itself, the entire network is compromised.

5.3 Types of Firewalls

Firewalls are classified by which layer of the OSI model they analyze to make blocking decisions.

5.3.1 1. Packet Filtering Firewall (Layer 3 - Network Layer)

This is the oldest, fastest, and simplest type of firewall. It acts like a bouncer at a club checking ID cards. - **How it works:** It inspects the "header" of an incoming IP packet. It looks at the Source IP address, the Destination IP address, and the Port number. It compares this against a simple list of rules (e.g., 'Rule 1: IF $\text{Source}_IP = 192.168.1.5 AND Port = 80, THEN Allow$'). $-$ ** $Advantage$: ** $Extremely fast because it only checks the envelope, not the letter inside.$ $-$ ** $Disadvantage$: ** $It is "stateless" and very easily fooled. It cannot look *inside* the packet payload. If a hacker sends a virus over an allowed port (li$

5.3.2 2. Application-Level Gateway (Layer 7 - Proxy Server)

This is a highly intelligent, but very slow firewall. It acts as an absolute middleman (a proxy) between the user and the internet. - **How it works:** If a user wants to download a file from an external FTP server, the user's computer actually connects to the Proxy Firewall. The Proxy Firewall then connects to the FTP server on the user's behalf. The firewall downloads the file, thoroughly scans the entire payload (the actual contents of the file) for viruses or malicious code, and only if it is 100% safe, hands it back to the user. - **Advantage:** Extremely secure. It completely hides the internal network and deeply inspects the actual data payload. - **Disadvantage:** Because it must reconstruct and scan entire files, it causes massive processing delays (latency) and requires heavy CPU power.

5.3.3 3. Circuit-Level Gateway (Layer 5 - Session Layer)

This is a middle ground between the Packet Filter and the Application Gateway. - **How it works:** Instead of checking every single packet, or checking the entire payload, a Circuit-Level Gateway focuses solely on the TCP Handshake. When an outside computer tries to start a connection, the firewall verifies if the TCP handshake sequence is mathematically legitimate. Once the firewall determines the connection is authentic and establishes the "circuit", it steps back and lets the data flow without inspecting the payloads. - **Advantage:** Faster than an Application Gateway and hides internal IP addresses. - **Disadvantage:** Once the initial connection is approved, a malicious insider could send a virus through the open circuit, and the firewall would not notice.