Problem Solver (DI03016021) - Problem Solver

Antigravity AI

June 4, 2026

Problem Solver

First Edition: 2026

Author: Antigravity AI
Website: milav.in
YouTube: @milav.dabgar

Copyright © 2026 by Antigravity AI

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

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

Preface

About this Book

Welcome to **Computer Networks**! This textbook is meticulously designed to serve as a comprehensive problem solver and study companion. It provides a robust and intuitive foundation in the core principles of this subject, bridging the gap between theoretical models and practical applications. A unique feature of this book is its focus on decoding complex concepts using clear, step-by-step methodologies and real-world analogies.

Target Audience

This book is intended for students and professionals looking to master the concepts of Computer Networks. Whether you are revising for exams, seeking to solidify your fundamental understanding, or looking for practical examples to clarify abstract concepts, this book has been structured to support your learning journey. It serves as an accessible guide designed to remove the fear of complex topics, replacing it with an appreciation for the underlying mechanics.

Scope and Coverage

The coverage is divided logically to follow standard academic curriculums. Each chapter focuses on key topics, breaking them down into digestible sections:

- **Theoretical Insights** – Core concepts explained intuitively.
- **Method Blocks** – Standardized, step-by-step procedures for solving specific types of problems.
- **Worked Examples** – Fully solved problems that apply the methods in practice.

We hope this resource proves invaluable in your studies and helps you achieve excellence in **Computer Networks**. Happy learning!

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.

Antigravity AI

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.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction of Information and Network Security

1.1 Computer Security, Cyber Security, Information Security and Network Security

Methodology:

Distinguishing Security Domains To classify a security scenario into the correct domain (Computer Security, Cyber Security, Information Security or Network Security), follow these algorithmic steps:

1. **Identify the core asset being protected:** If it is a standalone computing system and its physical components, it falls under **Computer Security**.
2. **Check for data form:** If the goal is protecting data across all mediums (both digital and physical paper records), it is **Information Security**.
3. **Check for network transmission:** If the focus is on protecting data during transmission over interconnected computing systems, it is **Network Security**.
4. **Check for internet exposure:** If the primary goal is protecting digital assets and systems from threats originating via the internet, it is **Cyber Security**.

Worked Example:

Classifying a Data Protection Scenario **Problem:** A company implements biometric locks on its physical file archive room and deploys firewalls for its web servers. Categorize these two measures into their respective security domains.
Solution:

1. **Step 1: Analyze biometric locks.** The locks protect physical access to paper files. Protecting data in physical forms is a core function of **Information Security**.
2. **Step 2: Analyze firewalls.** The firewalls protect digital assets from external internet-based threats. This falls primarily under **Cyber Security** and **Network Security**.

1.2 Essential Network and Computer Security Requirements

Methodology:

Evaluating Security Requirements To systematically evaluate the security requirements of any system, analyze the following properties known as the CIA Triad plus Authenticity and Accountability:

1. **Confidentiality:** Determine mechanisms to ensure data is only accessible to authorized entities.

2. **Integrity:** Determine mechanisms to ensure data has not been altered in an unauthorized manner during storage or transit.
3. **Availability:** Verify that systems and data are accessible to authorized users when needed.
4. **Authenticity:** Verify the identity of users and confirm the legitimate source of data.
5. **Accountability:** Implement logging to trace security-relevant actions uniquely to the entity that performed them.

Worked Example:

Assessing Requirements for a Banking System **Problem:** An online banking application allows users to view their balance and transfer funds. Identify how the essential security requirements apply to the fund transfer feature.
Solution:

1. **Step 1: Confidentiality.** The transaction details (amount and account numbers) must be encrypted so eavesdroppers cannot read them.
2. **Step 2: Integrity.** The transfer amount must not be altered during transmission.
3. **Step 3: Availability.** The banking servers must be online and resilient to denial-of-service attacks so users can transfer funds anytime.
4. **Step 4: Authenticity.** The system must verify the user's identity using a password and OTP before allowing the transfer.
5. **Step 5: Accountability.** The transaction must be securely logged with a timestamp and user ID so the bank can prove the user initiated the transfer.

1.3 OSI Security Architecture

Methodology:

Analyzing OSI Security Architecture Components The OSI Security Architecture provides a systematic framework for defining security. To analyze a system under this framework, perform the following procedure:

1. **Identify Security Attacks:** Determine the potential actions that compromise the security of information owned by an organization.
2. **Identify Security Services:** Determine the processing or communication services required to enhance the security of data processing systems and counter the identified attacks.
3. **Identify Security Mechanisms:** Select the specific algorithms or protocols designed to detect, prevent, or recover from the security attack and implement the chosen service.

Worked Example:

Mapping an OSI Security Scenario **Problem:** An employee sends a contract via email. To ensure the contract is not secretly modified by a competitor and to prove it came from the employee, a digital signature is applied. Map this to the OSI Security Architecture.
Solution:

1. **Step 1: Security Attack.** The potential attacks are message modification (changing the contract terms) and masquerade (forging the sender's identity).
2. **Step 2: Security Service.** The required services are **Data Integrity** (to prevent and detect

modification) and **Data Origin Authentication** (to verify the sender).

3. **Step 3: Security Mechanism.** The applied mechanism to fulfill these services is the **Digital Signature**.

1.4 Security Attacks: Passive Attacks and Active Attacks

Methodology:

Classifying Security Attacks To determine whether a security incident is a passive or active attack, use the following algorithm:

1. **Check for system alteration:** Does the attack alter system resources, modify data, or affect operation?
2. **If No (Passive Attack):** The goal is merely to obtain information. Sub-types include:
   - *Release of message contents:* Reading the actual data of an email or file.
   - *Traffic analysis:* Observing packet patterns, frequency, and lengths without reading the encrypted content.
3. **If Yes (Active Attack):** The attacker alters data or system operations. Sub-types include:
   - *Masquerade:* Pretending to be a different entity.
   - *Replay:* Capturing legitimate messages and retransmitting them to produce an unauthorized effect.
   - *Modification of messages:* Altering the content of a legitimate message.
   - *Denial of Service (DoS):* Preventing or inhibiting the normal use of communication facilities.

Worked Example:

Attack Scenario Analysis **Problem:** An attacker captures an encrypted login token sent by an administrator over a network. Later, the attacker sends the exact same token to the server to gain unauthorized administrative access. Classify this attack.
Solution:

1. **Step 1: Check for alteration.** The attacker is actively transmitting data to the server to affect its operation and gain access. Thus, it is an **Active Attack**.
2. **Step 2: Determine Sub-type.** The attacker did not decrypt or modify the token. Instead, they captured a legitimate message and retransmitted it at a later time. Therefore, the specific attack type is a **Replay Attack**.

1.5 Security Services

Methodology:

Identifying Security Services To select appropriate security services based on ITU-T X.800 architecture, map system requirements to these fundamental categories:

1. **Authentication:** Assures that the communicating entity is the one that it claims to be (Peer Entity Authentication and Data Origin Authentication).
2. **Access Control:** Prevents unauthorized use of a resource.

3. **Data Confidentiality:** Protects data from unauthorized disclosure (Connection Confidentiality, Connectionless Confidentiality, Selective-Field Confidentiality, Traffic Flow Confidentiality).
4. **Data Integrity:** Assures that data received are exactly as sent by an authorized entity (Connection Integrity with/without recovery, Connectionless Integrity).
5. **Non-repudiation:** Protects against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Worked Example:

Matching Services to Threats **Problem:** A trading platform must ensure that a user cannot deny having placed a stock purchase order, and competitors cannot see the stock volume being purchased. Identify the security services needed.
Solution:

1. **Step 1: Address the denial threat.** To prevent the user from denying the stock order, the system requires the **Non-repudiation** service.
2. **Step 2: Address the visibility threat.** To prevent competitors from reading the stock volume data, the system requires the **Data Confidentiality** service.

1.6 Security Mechanisms

Methodology:

Selecting Security Mechanisms Security mechanisms are implemented to provide the designated security services. To choose a mechanism, align it with the required service:

1. **Encipherment:** Use mathematical algorithms to transform data into a form that is not readily intelligible (Provides Confidentiality).
2. **Digital Signature:** Append data or a cryptographic transformation to a data unit (Provides Authentication, Data Integrity, and Non-repudiation).
3. **Access Control Mechanisms:** Use identity verification and privileges (Provides Access Control).
4. **Data Integrity Mechanisms:** Append a checksum or hash to detect modification (Provides Data Integrity).
5. **Authentication Exchange:** Use information exchange to ensure identity (Provides Authentication).
6. **Traffic Padding:** Insert bits into gaps in data streams to frustrate traffic analysis (Provides Traffic Flow Confidentiality).

Worked Example:

Implementing Mechanisms for a Secure Connection **Problem:** A military communication system needs to hide the frequency of messages being sent between two bases to prevent traffic analysis, and also ensure the messages cannot be read. Select the appropriate security mechanisms.
Solution:

1. **Step 1: Hide message frequency.** To protect against traffic analysis (a passive attack), the system should use **Traffic Padding** to generate continuous traffic.
2. **Step 2: Ensure messages cannot be read.** To provide confidentiality, the system must use **Encipherment** (Encryption) on the message contents.

1.7 Model for Network Security

Methodology:

Applying the Network Security Model The general model for network security outlines a standard procedure for secure communications. To implement this model, execute the following four computational tasks:

1. **Algorithm Design:** Design or select a cryptographic algorithm for performing the security-related transformation (e.g., AES or RSA).
2. **Secret Generation:** Compute and generate the secret information (keys) to be used alongside the algorithm.
3. **Secret Distribution:** Develop methods and protocols for securely distributing and sharing the secret keys between the communicating principals.
4. **Protocol Design:** Specify a network protocol enabling the two principals to utilize the security algorithm and secret information to achieve the desired security service over an insecure channel.

Worked Example:

Designing a Secure Message Exchange Protocol **Problem:** Two servers need to establish a secure link to exchange database backups over the public internet. Apply the four tasks of the network security model to establish this link.
Solution:

1. **Step 1: Algorithm Design.** The engineers select the Advanced Encryption Standard (AES) as the encryption algorithm.
2. **Step 2: Secret Generation.** A cryptographic random number generator is invoked on Server A to compute a 256-bit secret key.
3. **Step 3: Secret Distribution.** Server A uses the RSA public-key protocol to securely transmit the generated 256-bit AES key to Server B.
4. **Step 4: Protocol Design.** The servers implement a protocol where Server A encrypts the database file using AES and the shared key, transmits the ciphertext via TCP/IP, and Server B decrypts the received payload using the same shared key.

1.8 Cryptography: Concept and Classification

Methodology:

Classifying Cryptographic Systems Cryptographic systems are computationally classified based on the number of keys utilized in their algorithms. To classify a system, apply the following logic:

1. **Analyze the inputs:** Does the cryptographic algorithm require a secret key to perform its mathematical transformation?
2. **If No (Key-less Cryptography):** The system relies entirely on the mathematical algorithm and the input data.
    - *Use case:* Cryptographic Hash Functions (e.g., SHA-256) compute a fixed-length digest from variable-length data without any key.
3. **If Yes, count the keys:**
    - **Single-key Cryptography (Symmetric):** Exactly one key is used for both the encryption and

decryption processes. Both the sender and receiver must possess this shared secret key. Examples include AES and DES.

- **Two-key Cryptography (Asymmetric):** Two mathematically linked keys are generated: a Public Key (used for encryption) and a Private Key (used for decryption). The keys are distinct, and computing the private key from the public key is computationally infeasible. Examples include RSA and ECC.

Worked Example:

Determining the Cryptographic Approach **Problem:** A software distributor wants to achieve two goals: (1) allow users to verify that a downloaded software patch is exactly the one published and hasn't been corrupted, and (2) ensure that confidential configuration files sent between internal servers can be encrypted and decrypted extremely fast. Classify the necessary cryptographic approaches.
Solution:

1. **Step 1: Verifying software integrity.** The distributor computes a fixed-length checksum of the software patch using an algorithm like SHA-256 and publishes it. Users compute the same checksum. Since no key is required to compute this checksum, this relies on **Key-less Cryptography**.

2. **Step 2: Fast internal encryption.** The internal servers need to encrypt and decrypt large files quickly. Symmetric algorithms are computationally faster than asymmetric ones. The servers will share a single secret key to perform these operations. This approach is classified as **Single-key Cryptography**.

CHAPTER 2

Number Theory in Cryptography

2.1 Divisibility and Division Algorithm

Methodology:

Division Algorithm The Division Algorithm states that for any integer a and any positive integer n, there exist unique integers q (quotient) and r (remainder) such that:

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

where $0 \leq r < n$.
Steps to find q and r:

1. Divide a by n using standard division.
2. The integer part of the result is the quotient $q = \lfloor a/n \rfloor$.
3. Calculate the remainder $r = a - q \cdot n$.

If a is negative, be careful to choose q such that $0 \leq r < n$.

Worked Example:

Find the quotient and remainder when $a = 42$ is divided by $n = 5$. **Step 1:** Perform the division $42/5 = 8.4$.
Step 2: Extract the integer part as quotient $q = 8$.
Step 3: Calculate the remainder r:

$$r = a - q \cdot n = 42 - 8 \cdot 5 = 42 - 40 = 2$$

Final Answer: Quotient $q = 8$ and Remainder $r = 2$. Check: $42 = 8 \cdot 5 + 2$.

Worked Example:

Find the quotient and remainder when $a = -24$ is divided by $n = 7$. **Step 1:** Perform the division $-24/7 \approx -3.428$.
Step 2: The quotient must be the floor of the division to keep the remainder positive. $q = \lfloor -3.428 \rfloor = -4$.
Step 3: Calculate the remainder r:

$$r = a - q \cdot n = -24 - (-4) \cdot 7 = -24 + 28 = 4$$

Final Answer: Quotient $q = -4$ and Remainder $r = 4$. Check: $-24 = (-4) \cdot 7 + 4$.

2.2 Modulo Operator

Methodology:

Modulo Arithmetic Operations The modulo operator (mod) finds the remainder of division of one number by another. The expression $a \pmod{n}$ returns the remainder r when a is divided by n.
Properties of Modulo Arithmetic:

1. Addition: $(a+b) \pmod{n} = ((a \pmod{n}) + (b \pmod{n})) \pmod{n}$
2. Subtraction: $(a-b) \pmod{n} = ((a \pmod{n}) - (b \pmod{n})) \pmod{n}$
3. Multiplication: $(a \cdot b) \pmod{n} = ((a \pmod{n}) \cdot (b \pmod{n})) \pmod{n}$

For negative numbers $a \pmod{n}$, if the result is negative, add n to make it positive in the range $[0 \dots n-1]$.

Worked Example:

Calculate (15 + 22) mod 7 and (15 * 22) mod 7. **Addition:**

1. Find $15 \pmod{7} = 1$ (since $15 = 2 \cdot 7 + 1$).
2. Find $22 \pmod{7} = 1$ (since $22 = 3 \cdot 7 + 1$).
3. Add the results: $(1+1) \pmod{7} = 2 \pmod{7} = 2$.

Multiplication:

1. Use the mod values from above: $15 \pmod{7} = 1$ and $22 \pmod{7} = 1$.
2. Multiply the results: $(1 \cdot 1) \pmod{7} = 1 \pmod{7} = 1$.

Final Answer: The addition result is 2 and the multiplication result is 1.

Worked Example:

Calculate -18 mod 5. **Step 1:** Perform the division $-18/5$. The nearest multiple of 5 smaller than -18 is -20.
Step 2: Find the remainder: $-18 - (-20) = 2$.
Alternatively: Compute $18 \pmod{5} = 3$. Then $-18 \pmod{5} = -3$. Add the modulus 5 to get a positive result: $-3 + 5 = 2$.
Final Answer: $-18 \pmod{5} = 2$.

2.3 Set of Residues: Zn

Methodology:

Sets Zn and Zn* **Set of Residues ($\mathbb{Z}_n$):** The set of non-negative integers less than n.

$$\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$$

Set of Multiplicative Inverses ($\mathbb{Z}_n^*$): The subset of $\mathbb{Z}_n$ consisting of elements that are relatively prime to n. An element a is in $\mathbb{Z}_n^*$ if and only if $\text{GCD}(a, n) = 1$.

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \text{GCD}(a, n) = 1\}$$

Multiplicative Inverse in $\mathbb{Z}_n$: An element b is the multiplicative inverse of a in $\mathbb{Z}_n$ if:

$$(a \cdot b) \pmod{n} = 1$$

Every element in $\mathbb{Z}_n^*$ has a unique multiplicative inverse in $\mathbb{Z}_n^*$.

Worked Example:

Find the sets of Z10 and Z10*. **Step 1:** List all non-negative integers less than 10 to form $\mathbb{Z}_{10}$.

$$\mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Step 2: For each element $a \in \mathbb{Z}_{10}$ check if $\text{GCD}(a, 10) = 1$.

- $\text{GCD}(0, 10) = 10 \neq 1$
- $\text{GCD}(1, 10) = 1$ (Include)
- $\text{GCD}(2, 10) = 2 \neq 1$
- $\text{GCD}(3, 10) = 1$ (Include)
- $\text{GCD}(4, 10) = 2 \neq 1$
- $\text{GCD}(5, 10) = 5 \neq 1$
- $\text{GCD}(6, 10) = 2 \neq 1$
- $\text{GCD}(7, 10) = 1$ (Include)
- $\text{GCD}(8, 10) = 2 \neq 1$
- $\text{GCD}(9, 10) = 1$ (Include)

Final Answer: $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$.

Methodology:

Calculation of Addition and Multiplication Tables To construct an arithmetic table for $\mathbb{Z}_n$:

1. Create a grid with headers $0 \ldots n-1$ for both rows and columns.
2. For the **addition table** the entry in row i and column j is $(i + j) \pmod{n}$.
3. For the **multiplication table** the entry in row i and column j is $(i \cdot j) \pmod{n}$.

Worked Example:

Construct the addition and multiplication tables for Z5. **Step 1: Addition Table for $\mathbb{Z}_5$** Compute $(i + j) \pmod 5$ for all i and $j \in \{0, 1, 2, 3, 4\}$. For example row 2 col 4: $(2 + 4) \pmod 5 = 6 \pmod 5 = 1$.

+	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Step 2: Multiplication Table for $\mathbb{Z}_5$ Compute $(i \cdot j) \pmod 5$ for all i and $j \in \{0, 1, 2, 3, 4\}$. For example row 3 col 4: $(3 \cdot 4) \pmod 5 = 12 \pmod 5 = 2$.

×	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Final Answer: The tables are computed as shown above. Notice in the multiplication table every non-zero row contains a 1 confirming that every non-zero element in $\mathbb{Z}_5$ has a multiplicative inverse.

2.4 Calculation of GCD and Multiplicative Inverse

Methodology:

Euclidean Algorithm for GCD The Euclidean Algorithm is an efficient method to compute the Greatest Common Divisor (GCD) of two integers a and b (where $a \geq b$). **Algorithm Steps:**

1. Divide a by b to get quotient q and remainder r: $a = q \cdot b + r$.
2. If $r = 0$ the GCD is b. Stop.
3. If $r \neq 0$ update $a = b$ and $b = r$.
4. Repeat from Step 1 until the remainder becomes 0.

The last non-zero remainder is the GCD of the original numbers.

Worked Example:

Find the GCD of 2740 and 1760 using the Euclidean Algorithm. Let $a = 2740$ and $b = 1760$.
Step 1: $2740 = 1 \cdot 1760 + 980$ (Remainder $r = 980 \neq 0$)
Step 2: Update $a = 1760$ and $b = 980$. $1760 = 1 \cdot 980 + 780$ (Remainder $r = 780 \neq 0$)
Step 3: Update $a = 980$ and $b = 780$. $980 = 1 \cdot 780 + 200$ (Remainder $r = 200 \neq 0$)
Step 4: Update $a = 780$ and $b = 200$. $780 = 3 \cdot 200 + 180$ (Remainder $r = 180 \neq 0$)
Step 5: Update $a = 200$ and $b = 180$. $200 = 1 \cdot 180 + 20$ (Remainder $r = 20 \neq 0$)
Step 6: Update $a = 180$ and $b = 20$. $180 = 9 \cdot 20 + 0$ (Remainder $r = 0$)
The remainder is now 0. The last non-zero remainder (the divisor of the last step) is 20.
Final Answer: $\text{GCD}(2740, 1760) = 20$.

Methodology:

Extended Euclidean Algorithm The Extended Euclidean Algorithm not only finds the GCD of a and b but also finds integers x and y such that:

$$a \cdot x + b \cdot y = \text{GCD}(a, b)$$

This is heavily used to find the multiplicative inverse of $b \pmod a$. If $\text{GCD}(a, b) = 1$ then the resulting coefficient of b is its multiplicative inverse modulo a.

Tabular Method Steps:

1. Set up a table with columns: $q\ r_1\ r_2\ r\ t_1\ t_2\ t$.
2. Initialize: $r_1 = a\ r_2 = b\ t_1 = 0\ t_2 = 1$.
3. In each row compute:
   - $q = \lfloor r_1/r_2 \rfloor$
   - $r = r_1 - q \cdot r_2$
   - $t = t_1 - q \cdot t_2$
4. For the next row update: $r_1 = r_2\ r_2 = r$ and $t_1 = t_2\ t_2 = t$.
5. Stop when $r_2 = 0$. The last t_1 value is the inverse of $b \pmod a$. If t_1 is negative add a to it.

Worked Example:

Find the multiplicative inverse of 11 modulo 26 using Extended Euclidean Algorithm. We want to find $11^{-1} \pmod{26}$. Here $a = 26$ and $b = 11$.

Step 1: Initialize the table. $r_1 = 26\ r_2 = 11\ t_1 = 0\ t_2 = 1$.

q	r_1	r_2	r	t_1	t_2	t
-	26	11	-	0	1	-

Step 2: Row 1 computations. $q = \lfloor 26/11 \rfloor = 2\ r = 26 - (2 \cdot 11) = 4\ t = 0 - (2 \cdot 1) = -2$

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2

Step 3: Shift values for Row 2. $r_1 = 11\ r_2 = 4\ t_1 = 1\ t_2 = -2$. $q = \lfloor 11/4 \rfloor = 2\ r = 11 - (2 \cdot 4) = 3$ $t = 1 - (2 \cdot -2) = 1 + 4 = 5$

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5

Step 4: Shift values for Row 3. $r_1 = 4\ r_2 = 3\ t_1 = -2\ t_2 = 5$. $q = \lfloor 4/3 \rfloor = 1\ r = 4 - (1 \cdot 3) = 1$ $t = -2 - (1 \cdot 5) = -7$

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7

Step 5: Shift values for Row 4. $r_1 = 3\ r_2 = 1\ t_1 = 5\ t_2 = -7$. $q = \lfloor 3/1 \rfloor = 3\ r = 3 - (3 \cdot 1) = 0$ $t = 5 - (3 \cdot -7) = 5 + 21 = 26$

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26

Step 6: Shift values for the final state. $r_1 = 1\ r_2 = 0$. Since $r_2 = 0$ we stop. The GCD is $r_1 = 1$. The inverse is $t_1 = -7$.

Step 7: Make the inverse positive. Inverse $= -7 \pmod{26} = -7 + 26 = 19$.

Final Answer: The multiplicative inverse of 11 modulo 26 is 19. Check: $(11 \cdot 19) \pmod{26} = 209 \pmod{26} = 1$.

CHAPTER 3

Classical Encryption Techniques

3.1 Symmetric-key Encryption

Symmetric-key encryption, also known as secret-key or single-key encryption, is a model where the sender and receiver share the same key for both encryption and decryption.

A symmetric encryption scheme has five ingredients:

1. **Plaintext:** The original intelligible message or data that is fed into the algorithm as input.
2. **Encryption Algorithm:** The algorithm performs various substitutions and transformations on the plaintext.
3. **Secret Key:** The exact same key must be shared by the sender and receiver. The key acts as an independent input to the encryption algorithm.
4. **Ciphertext:** The scrambled message produced as output.
5. **Decryption Algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

3.1.1 Cryptanalysis and Brute-Force Attack

There are two general approaches to attacking a conventional encryption scheme: cryptanalysis and brute-force attacks. Cryptanalysis relies on the nature of the algorithm and mathematical patterns (like frequency analysis), while a brute-force attack systematically tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

Methodology:

Brute Force Attack Calculation To calculate the time required to perform a brute-force attack:

1. Identify the key size n in bits.
2. Calculate the total number of possible keys in the key space: $K_{total} = 2^n$.
3. Identify the computational speed of the attacker, S (decryptions per second).
4. Calculate the **Maximum Time** required to try all keys:

$$T_{max} = \frac{K_{total}}{S} \text{ seconds}$$

5. Calculate the **Average Time** required (statistically, the correct key is found after trying half the keys):

$$T_{avg} = \frac{T_{max}}{2} = \frac{2^{n-1}}{S} \text{ seconds}$$

Worked Example:

Brute Force Attack on DES Suppose an attacker wants to break the Data Encryption Standard (DES), which uses a 56-bit key. If the attacker has a specialized computing cluster capable of performing 10^9 decryptions per second, calculate the maximum and expected (average) time in days to crack the cipher.
Solution:

1. Key size $n = 56$ bits.
2. Total possible keys: $K_{total} = 2^{56} \approx 7.205 \times 10^{16}$.
3. Decryption speed $S = 10^9$ decryptions/second.
4. Maximum Time:
$$T_{max} = \frac{7.205 \times 10^{16}}{10^9} = 7.205 \times 10^7 \text{ seconds}$$
5. Convert seconds to days:
$$T_{max} \text{ in days} = \frac{7.205 \times 10^7}{60 \times 60 \times 24} \approx 834 \text{ days}$$
6. Average Time:
$$T_{avg} = \frac{834}{2} = 417 \text{ days}$$

The expected time to crack the DES key with this computing power is roughly 417 days.

3.2 Substitution Ciphers

A substitution cipher is one in which letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

3.2.1 Monoalphabetic Ciphers

A monoalphabetic substitution cipher maps each plaintext character to a single corresponding ciphertext character based on a single, fixed permutation of the alphabet. The Caesar cipher is the most foundational example of this technique.

Methodology:

Caesar Cipher Algorithm The Caesar cipher involves replacing each letter of the alphabet with the letter standing k places further down the alphabet. For computational purposes, assign a numerical value to each letter from $A = 0, B = 1, \ldots, Z = 25$.
Let P be the plaintext letter, C be the ciphertext letter, and K be the shift key.

1. **Encryption Equation:**
$$C = (P + K) \mod 26$$
2. **Decryption Equation:**
$$P = (C - K) \mod 26$$

Note: If $(C - K)$ is negative during decryption, add 26 to the result before applying modulo 26.

Worked Example:

Encrypting with Caesar Cipher Encrypt the message "HELLO" using the Caesar Cipher with a key $K = 3$.
Solution:

1. Assign numerical values to the plaintext letters:

Plaintext	H	E	L	L	O
P-Value	7	4	11	11	14

2. Apply encryption formula $C = (P + 3) \mod 26$:
   - H: $(7 + 3) \mod 26 = 10 \rightarrow$ K
   - E: $(4 + 3) \mod 26 = 7 \rightarrow$ H
   - L: $(11 + 3) \mod 26 = 14 \rightarrow$ O
   - L: $(11 + 3) \mod 26 = 14 \rightarrow$ O
   - O: $(14 + 3) \mod 26 = 17 \rightarrow$ R
3. The final ciphertext is **KHOOR**.

3.2.2 Polyalphabetic Ciphers

Polyalphabetic substitution ciphers use multiple monoalphabetic rules to improve security against frequency analysis cryptanalysis. The Vigenère cipher is a prominent example, utilizing a sequence of Caesar ciphers based on the letters of a keyword.

Methodology:

Vigenere Cipher Algorithm

1. Assign values $0 - 25$ to letters $A - Z$.
2. Repeat the Key K until it matches the length of the Plaintext P. This creates a keystream.
3. **Encryption:** For each position i:
$$C_i = (P_i + K_i) \mod 26$$
4. **Decryption:** For each position i:
$$P_i = (C_i - K_i + 26) \mod 26$$

Worked Example:

Encrypting with Vigenere Cipher Encrypt the plaintext "WEAREDISCOVERED" using the keyword "DECEPTIVE".
Solution:

1. Extend the keyword to match the length of the plaintext:

Plaintext	W	E	A	R	E	D	I	S	C	O	V	E	R	E	D
Keyword	D	E	C	E	P	T	I	V	E	D	E	C	E	P	T

2. Convert letters to numerical values and apply $C_i = (P_i + K_i) \mod 26$:

Char	P-Val	K-Val	Sum	Mod 26	C-Char
W/D	22	3	25	25	Z
E/E	4	4	8	8	I
A/C	0	2	2	2	C
R/E	17	4	21	21	V
E/P	4	15	19	19	T
D/T	3	19	22	22	W
I/I	8	8	16	16	Q
S/V	18	21	39	13	N
C/E	2	4	6	6	G
O/D	14	3	17	17	R
V/E	21	4	25	25	Z
E/C	4	2	6	6	G
R/E	17	4	21	21	V
E/P	4	15	19	19	T
D/T	3	19	22	22	W

3. The final ciphertext is **ZICVTWQNGRZGVTWT** (actually, **ZICVTWQNGRZGVTW** for 15 characters).

3.3 Transposition Ciphers

Rather than replacing letters, transposition ciphers rearrange the letters of the plaintext based on a specific algorithm.

3.3.1 Keyless Transposition

Keyless transpositions rely entirely on an agreed-upon pattern rather than a secret key string. The Rail Fence cipher is the simplest transposition cipher.

Methodology:

Rail Fence Algorithm The plaintext is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when the bottom rail is reached.

1. Identify the depth d (number of rails).
2. Write the plaintext characters in a zig-zag pattern spanning d rows.
3. To form the ciphertext, read off the letters row by row, from top to bottom.

Worked Example:

Rail Fence Cipher with Depth 2 Encrypt the message "MEETMEAFTERTHEPARTY" using a rail fence cipher of depth 2.
Solution:

1. Write the characters in a zig-zag of 2 rows:

M . E . M . A . T .
. E . T . E . F . E
E . R . H . P . R .
. T . E . A . T . Y

(Combined visually: Row 1 has M E M A T E R H P R, Row 2 has E T E F E T E A T Y)

2. Read off Row 1: MEMATERHPR

3. Read off Row 2: ETEFETEATY
4. Concatenate rows for Ciphertext: **MEMATERHPRETEFETEATY**

3.3.2 Keyed Transposition

A keyed transposition cipher involves writing the message in a rectangle, row by row, and reading the message off, column by column, but permuting the order of the columns according to a numeric key.

Methodology:

Row Transposition Algorithm

1. Obtain the numeric key (a sequence of digits without duplicates, e.g., 4-3-1-2-5-6-7).
2. Write the plaintext out in rows, where the length of each row matches the length of the numeric key.
3. Pad the final row with dummy characters (like X, Y, Z) if it is incomplete.
4. Output the ciphertext by reading the columns from top to bottom. The order in which the columns are read is determined by the numerical value of the key digits (read column labeled '1' first, then '2', etc.).

Worked Example:

Row Transposition with a Numeric Key Encrypt "ATTACKPOSTPONEDUNTILTWOAM" using Row Transposition with the key "4312567".
Solution:

1. Arrange plaintext in a grid matching the 7-digit key length. Pad with X, Y, Z:

4	**3**	**1**	**2**	**5**	**6**	**7**
A	T	T	A	C	K	P
O	S	T	P	O	N	E
D	U	N	T	I	L	T
W	O	A	M	X	Y	Z

2. Read out the columns sequentially starting from key digit 1:
   - Col 1 (3rd char): T T N A
   - Col 2 (4th char): A P T M
   - Col 3 (2nd char): T S U O
   - Col 4 (1st char): A O D W
   - Col 5 (5th char): C O I X
   - Col 6 (6th char): K N L Y
   - Col 7 (7th char): P E T Z
3. Concatenate: **TTNAAPTMTSUOAODWCOIXKNLYPETZ**

3.3.3 Stream and Block Ciphers

- **Stream Cipher:** Encrypts a digital data stream one bit or one byte at a time. Examples include the Vigenère cipher and RC4. Stream ciphers are computationally faster and require less memory.
- **Block Cipher:** A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. Examples include DES and AES. Block ciphers

allow for complex modes of operation like Cipher Block Chaining (CBC).

3.4 Asymmetric-key Cryptosystem

Unlike symmetric systems, asymmetric encryption (public-key cryptography) utilizes two distinct keys: a public key and a private key.

3.4.1 Ingredients and Usage

An asymmetric scheme relies on the following ingredients:

1. **Plaintext:** Intelligible message.
2. **Encryption algorithm:** Applies mathematical transformations using keys.
3. **Public and Private Key Pair:** If one is used for encryption, the other must be used for decryption.
4. **Ciphertext:** The scrambled output.
5. **Decryption algorithm:** Produces plaintext given the ciphertext and the correct matching key.

Confidentiality vs. Authentication:

- **Confidentiality:** Sender encrypts the message with the Receiver's **Public Key**. Only the Receiver's **Private Key** can decrypt it.
- **Authentication (Digital Signatures):** Sender encrypts (signs) the message with their own **Private Key**. Anyone can verify the sender's identity by decrypting the signature using the Sender's **Public Key**.

3.4.2 RSA Algorithm

The Rivest-Shamir-Adleman (RSA) algorithm is the most widely deployed public-key cryptosystem. It relies on the computational difficulty of factoring large composite numbers.

Methodology:

RSA Algorithm Core Math **1. Key Generation:**

1. Choose two distinct prime numbers, p and q.
2. Compute $n = p \times q$. The value n is used as the modulus for both keys.
3. Compute Euler's totient function: $\phi(n) = (p-1) \times (q-1)$.
4. Choose an integer e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$ (i.e., $\gcd(e, \phi(n)) = 1$).
5. Determine d as the modular multiplicative inverse of e modulo $\phi(n)$. This means finding d such that:

$$(e \times d) \mod \phi(n) = 1$$

6. **Public Key:** $KU = \{e, n\}$. **Private Key:** $KR = \{d, n\}$.

2. Encryption and Decryption: Given Plaintext block M (where $M < n$):

- **Encryption:** $C = M^e \mod n$
- **Decryption:** $M = C^d \mod n$

Worked Example:

RSA Key Generation and Encryption Perform RSA key generation given prime numbers $p = 17$ and $q = 11$, and public exponent $e = 7$. Then encrypt the plaintext message $M = 88$ and demonstrate the decryption process to retrieve M.

Solution:

Part 1: Key Generation

1. Compute n:
$$n = p \times q = 17 \times 11 = 187$$
2. Compute $\phi(n)$:
$$\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$$
3. We are given $e = 7$. We check that $\gcd(7, 160) = 1$, which is true.
4. Find private key d such that $(7 \times d) \mod 160 = 1$. Using trial multiplication or the Extended Euclidean Algorithm:
$$7 \times 23 = 161$$
$$161 \mod 160 = 1$$
Therefore, $d = 23$.
5. The keys are: **Public Key** $KU = \{7, 187\}$ and **Private Key** $KR = \{23, 187\}$.

Part 2: Encryption

1. Given $M = 88$. Find $C = 88^7 \mod 187$.
2. Using modular exponentiation:
$$88^1 \equiv 88 \pmod{187}$$
$$88^2 \equiv 7744 \equiv 77 \pmod{187}$$
$$88^4 \equiv 77^2 \equiv 5929 \equiv 132 \pmod{187}$$
3. $88^7 = 88^4 \times 88^2 \times 88^1$:
$$C \equiv (132 \times 77 \times 88) \pmod{187}$$
$$132 \times 77 = 10164 \equiv 66 \pmod{187}$$
$$66 \times 88 = 5808 \equiv 11 \pmod{187}$$
4. Ciphertext $\mathbf{C = 11}$.

Part 3: Decryption

1. Given $C = 11$ and $d = 23$. Find $M = 11^{23} \mod 187$.
2. Using modular exponentiation:
$$11^1 \equiv 11 \pmod{187}$$
$$11^2 \equiv 121 \pmod{187}$$
$$11^4 \equiv 121^2 \equiv 14641 \equiv 55 \pmod{187}$$
$$11^8 \equiv 55^2 \equiv 3025 \equiv 33 \pmod{187}$$
$$11^{16} \equiv 33^2 \equiv 1089 \equiv 154 \pmod{187}$$
3. Express 23 as powers of 2 $(23 = 16 + 4 + 2 + 1)$:
$$M \equiv 11^{16} \times 11^4 \times 11^2 \times 11^1 \pmod{187}$$
$$M \equiv 154 \times 55 \times 121 \times 11 \pmod{187}$$
$$154 \times 55 = 8470 \equiv 55 \pmod{187}$$
$$55 \times 121 = 6655 \equiv 110 \pmod{187}$$
$$110 \times 11 = 1210 \equiv 88 \pmod{187}$$
4. Plaintext $\mathbf{M = 88}$, which successfully matches our original message!

CHAPTER 4

Web Security

4.1 Web Security Considerations

Web security involves protecting web applications and infrastructure from various threats. From a computational perspective, analyzing web security requires identifying the types of threats (integrity, confidentiality, denial of service, authentication) and applying the appropriate cryptographic or protocol-level approach (Network level, Transport level, or Application level).

Methodology:

Web Threat Analysis and Approach Selection To secure a web system, follow these analytical steps:

1. **Identify the Threat Vector:** Categorize the attack into one of the following:
   - *Integrity:* Modification of user data or memory.
   - *Confidentiality:* Eavesdropping on the network or theft of information.
   - *Denial of Service (DoS):* Flooding with requests or malicious payloads.
   - *Authentication:* Impersonation or forged credentials.
2. **Select the Security Approach Level:**
   - *Network Level (IPSec):* Best for end-to-end transparent security between hosts.
   - *Transport Level (SSL/TLS):* Best for securing web browser to web server communications.
   - *Application Level (S/MIME PGP SET):* Best for securing specific application payloads or transactions.
3. **Determine the Cryptographic Mechanism:** Choose encryption for confidentiality, MAC/Hashes for integrity, and Digital Signatures for authentication.

Worked Example:

Securing a Web Infrastructure A company wants to secure its internal employee portal and a public-facing e-commerce site. The internal portal suffers from IP spoofing and packet sniffing on the local network. The public e-commerce site requires secure credit card transactions over the internet without requiring users to install specialized software. Analyze the threats and select the appropriate security approaches.

Step 1: Analyze the internal employee portal.

- *Threats:* IP spoofing (Authentication) and packet sniffing (Confidentiality).
- *Environment:* Local network between known internal hosts.
- *Approach Selection:* Network Level approach (IPSec) is most suitable because it provides transparent protection against sniffing and IP spoofing for all IP traffic between the internal hosts.

Step 2: Analyze the public e-commerce site.

- *Threats:* Eavesdropping on credit card details (Confidentiality) and server spoofing (Authentication).
- *Environment:* Public internet where clients only have standard web browsers.
- *Approach Selection:* Transport Level approach (SSL/TLS) is required because it is built into all modern web browsers and secures the HTTP traffic without requiring client-side modifications.

4.2 Secure Socket Layer and Transport Layer Security

Secure Socket Layer (SSL) and Transport Layer Security (TLS) operate as protocol stacks consisting of multiple sub-protocols (Record, Handshake, Change Cipher Spec, and Alert). The most computationally intensive part is the Handshake protocol, which negotiates keys and algorithms.

Methodology:

SSL TLS Handshake Protocol Execution The SSL/TLS Handshake involves the following sequential steps to establish a secure session:

1. **Phase 1 (Initiation):** Client sends `ClientHello` (lists supported cipher suites and a random number R_c). Server responds with `ServerHello` (selects a cipher suite and sends a random number R_s).
2. **Phase 2 (Server Authentication):** Server sends its digital certificate. Server may send a `ServerKeyExchange` message if the certificate does not contain enough data for key exchange. Server ends with `ServerHelloDone`.
3. **Phase 3 (Client Authentication and Key Exchange):** Client verifies the server's certificate. Client sends `ClientKeyExchange` containing the Pre-Master Secret (PMS) encrypted with the server's public key.
4. **Phase 4 (Session Key Generation):** Both sides independently compute the Master Secret (MS) using the PMS, R_c, and R_s:
$$MS = \text{Hash}(PMS, R_c, R_s)$$
They then generate the symmetric session keys and MAC secrets from the MS.
5. **Phase 5 (Finalization):** Both sides send `ChangeCipherSpec` to activate the new keys, followed by a `Finished` message containing a MAC of all previous handshake messages to verify integrity.

Worked Example:

Simulating the Key Derivation in SSL Handshake Suppose an SSL client and server are establishing a connection. The client generates a random number $R_c = 1011$ and the server generates $R_s = 2022$. The client securely transmits a Pre-Master Secret $PMS = 5555$ encrypted via RSA. Assume the hash function for master secret generation simplifies to $MS = (PMS \times R_c) + R_s$. Calculate the Master Secret and determine the total number of random variables involved in the key generation.
Step 1: Extract given parameters. Client Random (R_c) = 1011 Server Random (R_s) = 2022 Pre-Master Secret (PMS) = 5555
Step 2: Compute the Master Secret. Using the provided simplified formula:

$$\begin{aligned} MS &= (PMS \times R_c) + R_s \\ MS &= (5555 \times 1011) + 2022 \\ MS &= 5616105 + 2022 \\ MS &= 5618127 \end{aligned}$$

Step 3: Analyze the random variables. There are exactly three independent random values used to establish the master secret: the client random (R_c), the server random (R_s), and the pre-master secret (PMS) generated by the client. These ensure that the final MS is unique for every session.

4.3 HTTPS

HTTPS is the integration of HTTP over the SSL/TLS protocol. It defines how a web client (browser) initiates a secure connection to a web server and how it securely terminates that connection.

Methodology:

HTTPS Connection Initiation and Closure The protocol interactions for an HTTPS session are computationally defined as follows:

1. **Initiation:**
   - The client opens a TCP connection to the server on port 443.
   - The client initiates an SSL/TLS handshake over this TCP connection.
   - Upon successful completion of the handshake, cryptographic keys are established.
   - The client encapsulates HTTP requests (GET, POST) inside SSL/TLS Record payloads using the negotiated symmetric encryption and MAC.
2. **Closure:**
   - The HTTP client or server decides to terminate the connection.
   - The terminating party sends an SSL `close_notify` alert at the TLS Record level to securely signal the end of the encrypted stream.
   - Both parties then close the underlying TCP connection via a TCP FIN/ACK sequence.

Worked Example:

Tracing HTTPS Protocol Layers An application sends a 500-byte HTTP GET request over HTTPS. The negotiated SSL cipher suite uses AES-128 (block size 16 bytes) and HMAC-SHA1 (20 bytes). The SSL Record header is 5 bytes. Determine the total bytes transmitted over TCP for this single request, assuming padding is added to reach a multiple of the block size after the MAC is appended.
Step 1: Calculate the payload with MAC. Payload = HTTP Data = 500 bytes. MAC Size = 20 bytes. Data + MAC = 500 + 20 = 520 bytes.
Step 2: Determine block cipher padding. AES-128 has a block size of 16 bytes. Current size = 520 bytes. We need 520 mod 16 = 8 bytes. Thus, we need to add padding. To make the size a multiple of 16, we add 8 bytes of padding (so 520 + 8 = 528 bytes, and 528/16 = 33 blocks). Encrypted payload size = 528 bytes.
Step 3: Add the SSL Record Header. SSL Record Header = 5 bytes. Total transmitted size = Encrypted payload size + SSL Header Total = 528 + 5 = 533 bytes.
Therefore, the 500-byte HTTP request results in 533 bytes of TCP data due to HTTPS overhead.

4.4 Basic Concept of Secure Electronic Transactions

Secure Electronic Transaction (SET) is an application-level protocol designed to secure credit card transactions over the internet. The core computational innovation in SET is the *Dual Signature*, which links an order message meant for the merchant and a payment message meant for the bank, while keeping them hidden from each other.

Methodology:

SET Dual Signature Generation and Verification The Dual Signature ensures the merchant cannot see the credit card details, and the bank cannot see the order details, yet both can verify the transaction is linked.

1. **Message Preparation:** The customer prepares Order Information (OI) and Payment Information (PI).

2. **Hashing:** Calculate the hash of both messages:

$$H(OI) \quad \text{and} \quad H(PI)$$

3. **Concatenation:** Concatenate the two hashes:

$$POM = H(OI) \parallel H(PI)$$

where POM is the Payment Order Message hash.

4. **Dual Signature Generation:** The customer encrypts the concatenated hash with their private key (PR_c) to create the Dual Signature (DS):

$$DS = E(PR_c, POM)$$

5. **Transmission:** The merchant receives OI, $H(PI)$, and DS. The bank receives PI, $H(OI)$, and DS.

6. **Verification by Merchant:** The merchant computes $H(OI)$ from the received OI, concatenates it with the received $H(PI)$, hashes the result, and compares it to the decrypted DS using the customer's public key.

Worked Example:

Computing and Verifying a Dual Signature Assume a simplified hash function $H(x) = x \bmod 100$. A customer generates $OI = 456$ and $PI = 789$. The concatenation operator simply places the numbers side-by-side. The customer's private key encryption is simulated by adding 5000 to the value, and public key decryption subtracts 5000. Calculate the Dual Signature and show how the bank verifies it.
Step 1: Hashing the information. $H(OI) = 456 \bmod 100 = 56$ $H(PI) = 789 \bmod 100 = 89$
Step 2: Concatenating the hashes. Concatenation $POM = H(OI) \parallel H(PI) = 56 \parallel 89 = 5689$.
Step 3: Generating the Dual Signature. $DS = E(PR_c, POM) = 5689 + 5000 = 10689$. The customer sends $PI = 789$, $H(OI) = 56$, and $DS = 10689$ to the bank (via the merchant).
Step 4: Bank Verification Process. The bank receives $PI = 789$, $H(OI) = 56$, and $DS = 10689$. First, the bank decrypts the Dual Signature using the public key: Decrypted $DS = 10689 - 5000 = 5689$. Next, the bank hashes the received PI: Calculated $H(PI) = 789 \bmod 100 = 89$. Then, the bank concatenates the received $H(OI)$ and calculated $H(PI)$: Calculated $POM = 56 \parallel 89 = 5689$. Since the Calculated POM (5689) matches the Decrypted DS (5689), the bank successfully verifies the dual signature.

4.5 SSL versus SET

Choosing between SSL and SET involves evaluating the requirements of an e-commerce platform. SSL is universally supported but only encrypts the channel. SET is highly specialized for payment routing and data masking but requires complex certificate management.

Methodology:

Protocol Selection SSL vs SET Evaluate the deployment parameters to choose the correct protocol:

1. **Deployment Complexity:** If the system requires no client-side installation, choose **SSL**. If custom client wallets are acceptable, consider **SET**.

2. **Data Privacy from Merchant:** If the merchant must not see the credit card number, choose **SET** (utilizes Dual Signatures). If the merchant is trusted to handle credit card data, choose **SSL**.

3. **Authentication Requirements:** If only server authentication is strictly required, choose **SSL**. If explicit, certificate-based authentication of the cardholder, merchant, and payment gateway is mandatory, choose **SET**.

Worked Example:

Selecting the Protocol for an E-commerce Startup A new e-commerce startup needs to accept credit card payments. The startup expects most users to shop from mobile web browsers. They plan to use a third-party payment gateway but temporarily route the card details through their own web server for processing. Evaluate whether SSL or SET should be implemented for the connection between the customer and the startup.

Step 1: Analyze Client Constraints. The users will access the store via standard mobile web browsers. They will not install a specialized SET digital wallet app. This heavily favors SSL, as it is natively supported by all browsers.

Step 2: Analyze Privacy Constraints. The startup routes the card details through their own server, meaning the merchant (startup) sees the credit card data. SET is specifically designed to prevent this by encrypting the payment info directly for the bank. However, because the startup intentionally routes data this way, they cannot fully utilize SET's dual signature privacy without changing their architecture.

Step 3: Decision. The startup must use **SSL** (HTTPS) to secure the transport layer between the mobile browser and their web server. SET is not viable due to the lack of client-side software and the merchant's architectural need to process the payment data directly.

CHAPTER 5
System Security

In this chapter, we focus on the operational methods and algorithmic configurations used to secure computer systems. The core computational tasks in system security involve evaluating network traffic against access rules, detecting anomalous patterns, and managing secure proxy connections.

5.1 Intrusion Detection

Intrusion Detection Systems (IDS) monitor network traffic and system activities for malicious activities or policy violations. Intruders generally fall into categories such as masqueraders, misfeasors, and clandestine users. The two primary computational approaches to detect these intruders are Statistical Anomaly Detection and Rule-Based Signature Detection.

Methodology:

Statistical Anomaly Detection Statistical anomaly detection builds a profile of normal user behavior and triggers an alert when current behavior deviates significantly from this baseline.

1. **Define Metrics:** Select quantifiable metrics such as login frequency, CPU usage, or number of file accesses per minute.
2. **Establish Baseline:** Calculate the mean (μ) and standard deviation (σ) of the metric over a normal operational period.
3. **Measure Current Activity:** Continuously monitor the metric in real-time to obtain the current value x.
4. **Calculate Deviation Score:** Compute the standard score (Z-score) of the current activity:

$$Z = \frac{|x - \mu|}{\sigma}$$

5. **Evaluate Threshold:** Compare Z against a predefined threshold T. If $Z > T$, classify the activity as an anomaly and trigger an alert.

Worked Example:

Detecting Anomalous Login Attempts A system monitors the number of failed login attempts per hour. Over the past month, the average number of failed logins per hour is $\mu = 5$ with a standard deviation of $\sigma = 2$. The IDS is configured with an anomaly threshold of $T = 3$. During a specific hour, 12 failed login attempts are recorded. Determine if the IDS will trigger an alert.

Step 1: Identify the given parameters. Mean (μ) = 5, Standard Deviation (σ) = 2, Threshold (T) = 3, Current Value (x) = 12.

Step 2: Calculate the deviation score (Z-score).

$$Z = \frac{|x - \mu|}{\sigma}$$

$$Z = \frac{|12 - 5|}{2}$$

$$Z = \frac{7}{2} = 3.5$$

Step 3: Evaluate against the threshold. Compare the Z-score with the threshold T:

$$3.5 > 3$$

Since the Z-score exceeds the threshold, the activity is classified as anomalous.
Conclusion: The IDS will trigger an intrusion alert.

Methodology:

Rule Based Signature Detection Signature-based detection compares network traffic against a database of known malicious patterns or signatures.

1. **Extract Payload:** Capture the network packet and extract its payload or specific header fields.
2. **Pattern Matching:** Search for specific byte sequences or strings (signatures) within the payload.
3. **Rule Evaluation:** If a payload matches a signature rule, execute the corresponding action (e.g. generate alert or drop packet).

Worked Example:

Evaluating Traffic Against IDS Signatures An IDS is configured with the following signature rules:

- **Rule 1:** Alert if payload contains the string "USER root" on port 21.
- **Rule 2:** Alert if payload contains the hexadecimal sequence "90 90 90" (NOP sled) on port 80.

Evaluate the following incoming packets:

- **Packet A:** Destination Port = 21, Payload = "USER admin"
- **Packet B:** Destination Port = 80, Payload = "GET / HTTP/1.1 ... 90 90 90 90 ..."

Step 1: Evaluate Packet A. Packet A is on port 21. We check against Rule 1. The payload is "USER admin", which does not match the signature "USER root". No alert is triggered for Packet A.
Step 2: Evaluate Packet B. Packet B is on port 80. We check against Rule 2. The payload contains the sequence "90 90 90", which matches the signature for a NOP sled.
Conclusion: The IDS will trigger an alert for Packet B based on Rule 2, while Packet A passes without triggering an alert.

5.2 Firewall Definition Need and Characteristics

A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Firewalls are needed to establish a barrier between a trusted internal network and an untrusted external network (such as the Internet). Their primary characteristics include a defined choke point for all traffic, resistance to penetration, and the ability to enforce local security policies.

Methodology:

Firewall Access Control List Evaluation Firewalls use Access Control Lists (ACLs) to determine the fate of network packets. The evaluation algorithm follows these steps:

1. **Extract Packet Headers:** Identify the Source IP, Destination IP, Protocol (TCP/UDP/ICMP),

Source Port, and Destination Port from the incoming packet.

2. **Sequential Rule Processing:** Start from the first rule in the ACL and move downwards sequentially.
3. **Pattern Matching:** For each rule, check if the packet's headers match the rule's criteria. A criteria of "ANY" or "*" matches all values.
4. **Action Execution:** If a rule matches, immediately apply its specified action (usually PERMIT or DENY) and halt further evaluation.
5. **Default Deny:** If the packet does not match any explicit rule in the ACL, apply the default rule, which is typically to DENY the packet.

Worked Example:

Processing Packets Through a Firewall ACL A firewall has the following Access Control List (ACL):

Rule	Action	Src IP	Dest IP	Protocol	Src Port	Dest Port
1	PERMIT	192.168.1.0/24	ANY	TCP	ANY	80
2	DENY	10.0.0.5	ANY	ANY	ANY	ANY
3	PERMIT	ANY	192.168.1.100	TCP	ANY	443

Note: Implicit DENY ANY ANY is active at the end.
Evaluate the fate of the following packets:

- **Packet X:** Src IP = 192.168.1.50, Dest IP = 8.8.8.8, Protocol = TCP, Src Port = 5000, Dest Port = 80
- **Packet Y:** Src IP = 10.0.0.5, Dest IP = 192.168.1.100, Protocol = TCP, Src Port = 6000, Dest Port = 443
- **Packet Z:** Src IP = 172.16.0.5, Dest IP = 8.8.8.8, Protocol = UDP, Src Port = 4000, Dest Port = 53

Step 1: Evaluate Packet X. Check Rule 1: Src IP 192.168.1.50 is within 192.168.1.0/24. Dest IP 8.8.8.8 matches ANY. Protocol is TCP. Dest Port is 80. All conditions match. Action: PERMIT. Evaluation stops.
Step 2: Evaluate Packet Y. Check Rule 1: Src IP 10.0.0.5 is not in 192.168.1.0/24. Match fails. Check Rule 2: Src IP is 10.0.0.5. All other fields are ANY, which match. Condition matches. Action: DENY. Evaluation stops. (Note: Even though it would match Rule 3, Rule 2 is processed first).
Step 3: Evaluate Packet Z. Check Rule 1: Src IP 172.16.0.5 fails. Check Rule 2: Src IP 172.16.0.5 fails. Check Rule 3: Dest IP 8.8.8.8 fails (not 192.168.1.100). Check Implicit Default Rule: No rules matched. Action: DENY.
Conclusion: Packet X is permitted, Packet Y is denied, and Packet Z is denied.

5.3 Types of Firewall

Firewalls can be classified based on the layer of the OSI model at which they operate. The three main types are packet filtering firewalls, application-level gateways, and circuit-level gateways.

5.3.1 Packet Filtering Firewall

Packet filtering firewalls operate at the Network and Transport layers (Layers 3 and 4). They evaluate individual packets purely based on header information, without inspecting the payload or understanding the state of the connection.

Methodology:

Packet Filtering Ruleset Generation Designing a packet filter requires mapping business policies into explicit IP and port rules.

1. **Identify the Policy:** Define the desired traffic flow (e.g. "Allow external web traffic to the internal web server").
2. **Determine Inbound Rule:** Create a rule allowing traffic from the external source to the internal destination IP and specific service port.
3. **Determine Outbound Rule:** For stateless firewalls, create a corresponding rule allowing return traffic from the internal server to the external client on ephemeral ports (typically > 1023).

Worked Example:

Configuring a Stateless Packet Filter for Web Traffic **Problem Statement:** An internal web server is hosted at IP address 10.1.1.50. You need to configure a stateless packet filtering firewall to allow external clients from any IP address to access this web server over HTTP (port 80). The external clients use ephemeral ports ranging from 1024 to 65535. Design the required inbound and outbound rules.
Step 1: Analyze the Inbound Requirement. Traffic originates from the outside (ANY IP, Port > 1023) and goes to the inside web server (IP 10.1.1.50, Port 80) using TCP. Rule 1 (Inbound): Action = PERMIT, Src IP = ANY, Dest IP = 10.1.1.50, Protocol = TCP, Src Port = > 1023, Dest Port = 80.
Step 2: Analyze the Outbound Requirement. Because the firewall is stateless, it will not automatically allow return traffic. We must explicitly permit the server to respond to the clients. Traffic originates from the web server (IP 10.1.1.50, Port 80) and goes to the external client (ANY IP, Port > 1023). Rule 2 (Outbound): Action = PERMIT, Src IP = 10.1.1.50, Dest IP = ANY, Protocol = TCP, Src Port = 80, Dest Port = > 1023.
Conclusion: To successfully allow web traffic, the firewall must be configured with both the inbound rule targeting port 80 and the outbound rule permitting responses to ephemeral ports.

5.3.2 Application Level Gateway

An Application-Level Gateway (also known as a proxy server) operates at the Application layer (Layer 7). It acts as an intermediary, terminating the client's connection and establishing a separate connection to the destination. It understands application protocols like HTTP, FTP, or SMTP and can filter based on specific commands or URLs.

Methodology:

Application Level Proxy Filtering An application-level gateway inspects the full payload of the application traffic.

1. **Intercept Connection:** The gateway accepts the TCP connection from the client.
2. **Extract Application Data:** The gateway buffers the payload and parses the application-layer protocol commands (e.g. HTTP headers).
3. **Evaluate Content Rules:** Compare the extracted data against specific application policies (e.g. blocked domains, disallowed HTTP methods like PUT or DELETE).
4. **Relay or Block:** If permitted, the gateway initiates a new connection to the target server and relays the verified data. If denied, the gateway drops the connection and returns an error to the client.

Worked Example:

Filtering HTTP Requests at the Application Layer An organization uses an Application-Level Gateway configured to block all HTTP POST requests to prevent data exfiltration, while allowing standard HTTP

GET requests.
The proxy intercepts the following HTTP request from a client:

```
POST /upload_data HTTP/1.1
Host: external-server.com
Content-Length: 500

[500 bytes of sensitive data]
```

Determine the proxy's computational steps to handle this packet.
Step 1: Intercept the connection. The proxy accepts the TCP SYN from the client and completes the three-way handshake, acting as the server.
Step 2: Extract Application Data. The proxy reads the payload and parses the HTTP header. It identifies the HTTP method as "POST" and the target URL as "/upload_data".
Step 3: Evaluate Content Rules. The rule engine checks the identified method against the policy: "Block all HTTP POST requests". The condition (Method == POST) is met.
Step 4: Relay or Block. The proxy denies the request. It closes the connection to the client and does not initiate a connection to 'external-server.com'.
Conclusion: The application-level gateway successfully prevents the data exfiltration attempt by interpreting the Layer 7 HTTP command.

5.3.3 Circuit Level Gateway

A Circuit-Level Gateway operates primarily at the Session layer (Layer 5) or Transport layer (Layer 4). It establishes a virtual circuit between the client and server without inspecting the application data. It relies on verifying TCP handshakes and validating the session.

Methodology:

Circuit Level Session Relay Circuit-level gateways relay TCP segments between networks once a connection is validated.

1. **Client Authentication (Optional):** Verify the identity of the client requesting the external connection.
2. **Client TCP Handshake:** The gateway completes the TCP three-way handshake with the internal client.
3. **Server TCP Handshake:** The gateway initiates a separate TCP three-way handshake with the external server on behalf of the client.
4. **Blind Relay:** Once both connections are established, the gateway simply copies bytes back and forth between the two connections without examining the application payload.

Worked Example:

Establishing a SOCKS Proxy Connection An internal client wishes to connect to an external SSH server using a Circuit-Level Gateway implementing the SOCKS protocol. Determine the sequence of operations required to establish the circuit.
Step 1: Client to Gateway Connection. The client sends a TCP SYN to the SOCKS proxy. The proxy replies with SYN-ACK, and the client sends ACK. The connection between the client and the proxy is established.
Step 2: Connection Request. The client sends a SOCKS connection request specifying the target IP address and destination port (Port 22 for SSH).
Step 3: Gateway to Server Connection. The SOCKS proxy evaluates if the client is authorized to make this outbound connection. If approved, the proxy sends a TCP SYN to the external SSH server on Port 22. The external server replies with SYN-ACK, and the proxy sends an ACK.

Step 4: Data Relay. The proxy informs the client that the connection was successful. From this point on, whenever the client sends SSH encrypted data, the proxy blindly forwards these TCP segments to the external server, and forwards the server's responses back to the client. The proxy does not attempt to decrypt or inspect the SSH data.
Conclusion: Two separate TCP connections are established and linked by the gateway, providing a secure, opaque conduit for the client's traffic.

Appendix: Key Formulae

This appendix provides a quick reference to the key formulae and mathematical relationships discussed in this book.

Unit 2: Modular Arithmetic

Properties of Modulo Operation

The modulo operation distributes over addition, subtraction, and multiplication. For any integers a, b, and n:

$$\begin{aligned}(a+b) \pmod n &= \Big((a \pmod n) + (b \pmod n)\Big) \pmod n \\ (a-b) \pmod n &= \Big((a \pmod n) - (b \pmod n)\Big) \pmod n \\ (a \cdot b) \pmod n &= \Big((a \pmod n) \cdot (b \pmod n)\Big) \pmod n\end{aligned}$$

Division Algorithm

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

$$a = q \cdot n + r \quad \text{where } 0 \leq r < n$$

Unit 3: Cryptography and Key Management

Classical Ciphers

For a simple substitution cipher with a shift key K, the plaintext P and ciphertext C are related by:

$$\begin{aligned}\text{Encryption: } C &= (P + K) \bmod 26 \\ \text{Decryption: } P &= (C - K) \bmod 26\end{aligned}$$

For a polyalphabetic cipher (such as the Vigenère cipher) with a key stream K_i:

$$\begin{aligned}\text{Encryption: } C_i &= (P_i + K_i) \bmod 26 \\ \text{Decryption: } P_i &= (C_i - K_i + 26) \bmod 26\end{aligned}$$

RSA Algorithm

The RSA public-key cryptosystem relies on the following key generation parameters:

$$\begin{aligned}\text{Modulus: } n &= p \times q \\ \text{Euler's Totient: } \phi(n) &= (p-1)(q-1) \\ \text{Key Relation: } (e \times d) \pmod{\phi(n)} &= 1\end{aligned}$$

where p and q are distinct prime numbers, e is the public exponent, and d is the private exponent.

Brute-Force Attack Time

The time required to exhaustively search a keyspace of size $K_{total} = 2^n$ (for an n-bit key) at a search rate S (keys per second):

$$\text{Maximum Time: } T_{max} = \frac{K_{total}}{S} = \frac{2^n}{S} \text{ seconds}$$

$$\text{Average Time: } T_{avg} = \frac{T_{max}}{2} = \frac{2^{n-1}}{S} \text{ seconds}$$

Unit 4: Network Security Protocols

SSL/TLS Protocol

The Master Secret (MS) generation in SSL/TLS uses the Pre-Master Secret (PMS) along with client and server nonces (R_c and R_s):

$$MS = \text{Hash}(PMS, R_c, R_s)$$

Secure Electronic Transaction (SET)

The Dual Signature (DS) securely links Order Information (OI) and Payment Information (PI):

$$\text{Payment Order Message: } POM = H(OI) \parallel H(PI)$$

$$\text{Dual Signature: } DS = E(PR_c, POM)$$

where H is a cryptographic hash function, $\parallel$ denotes concatenation, and $E(PR_c, \cdot)$ denotes encryption using the customer's private key.

Unit 5: Intrusion Detection Systems

Statistical Anomaly Detection

The standard score (Z-score) is used to detect deviations from typical behavior profiles, measuring how many standard deviations σ an observation x is from the mean μ:

$$Z = \frac{|x - \mu|}{\sigma}$$