Csdf (4361601) - Problem Solver

Antigravity AI

June 4, 2026

Csdf

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

Overview of Information Security and Cryptography

1.1 Introduction of Information Security and Cryptography

In this section we focus on the fundamental computational models used to secure information through basic cryptography. Cryptography relies heavily on mathematical structures, specifically modular arithmetic, to transform plaintext into ciphertext and vice versa.

Methodology:

Modular Arithmetic for Cryptography Modular arithmetic is the mathematical foundation for many encryption algorithms. The operation $a \pmod m$ finds the remainder when a is divided by m. In cryptography we often use modulo 26, where the letters of the alphabet A through Z are mapped to the numbers 0 through 25.
Steps to perform modular addition and subtraction: 1. Assign numerical values to the alphabet: $A = 0, B = 1, C = 2, \ldots, Z = 25$. 2. For addition (encryption): Calculate $C = (P + K) \pmod{26}$. 3. For subtraction (decryption): Calculate $P = (C - K) \pmod{26}$. 4. If the result is negative, add 26 to it until it falls within the range of 0 to 25.

Worked Example:

Basic Modular Arithmetic Operations Calculate the following modular arithmetic expressions commonly used in cryptography: 1. $(15 + 20) \pmod{26}$ 2. $(5 - 12) \pmod{26}$
Solution:
Step 1: Solve the addition problem. $15 + 20 = 35$
Divide 35 by 26 to find the remainder: $35 = 26 \times 1 + 9$
So, $(15 + 20) \pmod{26} = 9$.
Step 2: Solve the subtraction problem. $5 - 12 = -7$
Since the result is negative, add the modulus (26) to it: $-7 + 26 = 19$
So, $(5 - 12) \pmod{26} = 19$.

Methodology:

The Caesar Cipher Algorithm The Caesar Cipher is a substitution cipher where each letter in the plaintext is shifted a certain number of places down the alphabet.
Let P be the plaintext letter and C be the ciphertext letter. Let K be the shift key.
Encryption Algorithm: 1. Convert the plaintext letter to its numerical equivalent $(0 \leq P \leq 25)$. 2. Apply the encryption formula: $C = (P + K) \pmod{26}$. 3. Convert the resulting number C back to a letter.
Decryption Algorithm: 1. Convert the ciphertext letter to its numerical equivalent $(0 \leq C \leq 25)$. 2. Apply the decryption formula: $P = (C - K) \pmod{26}$. 3. Convert the resulting number P back to a letter.

Worked Example:

Encrypting with Caesar Cipher Encrypt the plaintext "HELLO" using a Caesar cipher with a shift key of $K = 3$.

Solution:

Step 1: Map the letters to numbers. H = 7 E = 4 L = 11 L = 11 O = 14

Step 2: Apply the encryption formula $C = (P + 3) \pmod{26}$.

For H (7): $C = (7 + 3) \pmod{26} = 10 \pmod{26} = 10 \rightarrow$ K

For E (4): $C = (4 + 3) \pmod{26} = 7 \pmod{26} = 7 \rightarrow$ H

For L (11): $C = (11 + 3) \pmod{26} = 14 \pmod{26} = 14 \rightarrow$ O

For L (11): $C = (11 + 3) \pmod{26} = 14 \pmod{26} = 14 \rightarrow$ O

For O (14): $C = (14 + 3) \pmod{26} = 17 \pmod{26} = 17 \rightarrow$ R

Step 3: Combine the letters. The ciphertext is "KHOOR".

Worked Example:

Decrypting with Caesar Cipher Decrypt the ciphertext "FDHVDU" using a Caesar cipher with a shift key of $K = 3$.

Solution:

Step 1: Map the letters to numbers. F = 5, D = 3, H = 7, V = 21, D = 3, U = 20

Step 2: Apply the decryption formula $P = (C - 3) \pmod{26}$.

For F (5): $P = (5 - 3) \pmod{26} = 2 \pmod{26} = 2 \rightarrow$ C

For D (3): $P = (3 - 3) \pmod{26} = 0 \pmod{26} = 0 \rightarrow$ A

For H (7): $P = (7 - 3) \pmod{26} = 4 \pmod{26} = 4 \rightarrow$ E

For V (21): $P = (21 - 3) \pmod{26} = 18 \pmod{26} = 18 \rightarrow$ S

For D (3): $P = (3 - 3) \pmod{26} = 0 \pmod{26} = 0 \rightarrow$ A

For U (20): $P = (20 - 3) \pmod{26} = 17 \pmod{26} = 17 \rightarrow$ R

Step 3: Combine the letters. The plaintext is "CAESAR".

Methodology:

Brute Force Attack on Shift Ciphers A brute force attack or exhaustive key search involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained.

Algorithm for Brute Force Attack: 1. Start with a key $K = 1$. 2. Apply the decryption formula $P = (C - K) \pmod{26}$ to a segment of the ciphertext. 3. Examine the resulting text. If it is readable and makes sense, the correct key is found. 4. If not readable, increment K by 1 ($K = K + 1$) and repeat from Step 2. 5. Continue up to $K = 25$ since a shift of 26 results in the original text.

Worked Example:

Performing a Brute Force Attack An intercepted message is "WTAAD". It is known to be encrypted using a Caesar cipher. Find the original plaintext using a brute force attack.

Solution:

Step 1: Map ciphertext letters to numbers. W = 22, T = 19, A = 0, A = 0, D = 3

Step 2: Try possible keys systematically.

Key (K)	**Decryption Calculation** $P = (C - K) \pmod{26}$	**Decrypted Word**
1	(21, 18, 25, 25, 2)	VSZZC
2	(20, 17, 24, 24, 1)	URYYB
3	(19, 16, 23, 23, 0)	TQXXA
...	...	...
14	(8, 5, 12, 12, 15)	IFMMP
15	(7, 4, 11, 11, 14)	HELLO

Step 3: Analyze the results. When $K = 15$, the decrypted word is "HELLO", which is an intelligible English word. Thus, the plaintext is "HELLO" and the encryption key was 15.

CHAPTER 2

Network and System Security

2.1 IP Addressing and Subnetting Calculations

Methodology:

Subnetting Calculation To find the network details for a given IP address and CIDR prefix (e.g. 192.168.1.50/26):

1. **Identify the Subnet Mask**: Convert the CIDR prefix (number of 1s) into a 32-bit binary number and then to dotted-decimal format.
2. **Calculate the Network Address**: Perform a bitwise AND operation between the IP address and the Subnet Mask.
3. **Calculate the Broadcast Address**: Perform a bitwise OR operation between the Network Address and the inverted Subnet Mask (Wildcard Mask).
4. **Determine the Host Range**: The usable host IP addresses range from (Network Address + 1) to (Broadcast Address - 1).
5. **Find Number of Usable Hosts**: Calculate $2^{(32-\text{prefix})} - 2$.

Worked Example:

Calculate Subnet Details for 192.168.10.75/28 **Step 1: Identify the Subnet Mask**
A /28 prefix means 28 bits are 1s and 4 bits are 0s.
Binary: 11111111.11111111.11111111.11110000
Decimal Subnet Mask: 255.255.255.240
Step 2: Calculate the Network Address
IP Address: 192.168.10.75 $\rightarrow$ 192.168.10.01001011
Subnet Mask: 255.255.255.240 $\rightarrow$ 255.255.255.11110000
Bitwise AND of the last octet: 01001011 AND 11110000 = 01000000 (64 in decimal)
Network Address: 192.168.10.64
Step 3: Calculate the Broadcast Address
Wildcard Mask (Inverted Subnet Mask): 0.0.0.15 (Binary: 00001111)
Bitwise OR with Network Address last octet: 01000000 OR 00001111 = 01001111 (79 in decimal)
Broadcast Address: 192.168.10.79
Step 4: Determine the Host Range
First Usable Host: 192.168.10.64 + 1 = 192.168.10.65
Last Usable Host: 192.168.10.79 − 1 = 192.168.10.78
Host Range: 192.168.10.65 to 192.168.10.78
Step 5: Number of Usable Hosts
$2^{(32-28)} - 2 = 2^4 - 2 = 16 - 2 = 14$ usable hosts.

2.2 Firewall Rule Evaluation

Methodology:

Evaluating Access Control Lists Access Control Lists (ACLs) are evaluated top-down. To determine the fate of a packet:

1. Compare the packet's parameters (Source IP, Destination IP, Protocol, Destination Port) against the first rule.
2. If all conditions match, apply the rule's action (ALLOW or DENY) and **stop** evaluating further rules.
3. If the conditions do not match, proceed to the next rule.
4. If no rules match the packet, apply the implicit **Deny All** rule at the bottom of the list.

Worked Example:

Evaluate Packet against Firewall Rules Given the following Firewall Rules:

Rule	Action	Source IP	Dest IP	Protocol	Dest Port
1	ALLOW	10.0.0.0/24	Any	TCP	80
2	DENY	Any	Any	UDP	53
3	ALLOW	192.168.1.50	10.0.0.5	TCP	22
4	DENY	Any	Any	Any	Any

Determine the action for the following packets:
Packet A: Source IP 10.0.0.15, Dest IP 192.168.5.5, Protocol TCP, Dest Port 80
Packet B: Source IP 192.168.1.50, Dest IP 10.0.0.5, Protocol UDP, Dest Port 53
Packet C: Source IP 172.16.0.5, Dest IP 10.0.0.5, Protocol TCP, Dest Port 22
Solution:
Packet A Evaluation:

- Rule 1: Source 10.0.0.15 matches 10.0.0.0/24. Dest IP matches 'Any'. Protocol TCP matches. Dest Port 80 matches.
- Result: **ALLOW** (Evaluation stops).

Packet B Evaluation:

- Rule 1: Protocol is UDP but rule requires TCP. No match.
- Rule 2: Source 'Any' matches. Dest 'Any' matches. Protocol UDP matches. Dest Port 53 matches.
- Result: **DENY** (Evaluation stops).

Packet C Evaluation:

- Rule 1: Source 172.16.0.5 does not match 10.0.0.0/24. No match.
- Rule 2: Protocol TCP does not match UDP. No match.
- Rule 3: Source 172.16.0.5 does not match 192.168.1.50. No match.
- Rule 4: Matches all remaining packets.
- Result: **DENY**.

2.3 Intrusion Detection System Rule Matching

Methodology:

Analyzing IDS Rules IDS rules (like Snort) define patterns to identify malicious traffic. The structure is typically:
`action protocol source_ip source_port -> dest_ip dest_port (options)`

1. **Action**: What the IDS should do (e.g. alert or drop).
2. **Header**: Checks protocol, source IP/port, direction (`->`), and destination IP/port.
3. **Options**: Enclosed in parentheses and separated by semicolons. Contains specific payload content to match, thresholding, and alert messages.
4. A packet triggers the rule only if it matches **both** the header criteria and **all** specified options (like `content`).

Worked Example:

Evaluate Packet against Snort IDS Rule **Given Rule:**
`alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"SQL Injection Attempt"; content:"%27+OR+1%3D1"; sid:100001;)`
Assume:

- `$EXTERNAL_NET` = Any IP outside 192.168.1.0/24
- `$HTTP_SERVERS` = 192.168.1.100

Determine if the following packets trigger an alert:
Packet 1: Source: 203.0.113.5:45678, Destination: 192.168.1.100:80, Protocol: TCP
Payload: `GET /login.php?user=%27+OR+1%3D1 HTTP/1.1`
Packet 2: Source: 192.168.1.50:50123, Destination: 192.168.1.100:80, Protocol: TCP
Payload: `GET /login.php?user=%27+OR+1%3D1 HTTP/1.1`
Solution:
Packet 1 Evaluation:

- Protocol: TCP (Matches)
- Source: 203.0.113.5 is in `$EXTERNAL_NET` (Matches)
- Source Port: 45678 matches 'any' (Matches)
- Destination: 192.168.1.100 is `$HTTP_SERVERS` (Matches)
- Dest Port: 80 (Matches)
- Payload Content: Contains "%27+OR+1%3D1" (Matches)
- **Result:** Rule is triggered. An alert is generated.

Packet 2 Evaluation:

- Source: 192.168.1.50 is an internal IP so it does **not** match `$EXTERNAL_NET`.
- **Result:** Rule is NOT triggered due to Header mismatch.

2.4 Risk Calculation

Methodology:

Calculating Risk Level Risk is mathematically defined as the product of the probability of an event occurring and its impact.

1. Assign a numerical value to **Likelihood** (e.g. 1=Low, 2=Medium, 3=High).
2. Assign a numerical value to **Impact** (e.g. 1=Low, 2=Medium, 3=High).
3. **Calculate Risk**: Risk = Likelihood × Impact.
4. **Determine Severity**:
   - 1 to 2: Low Risk
   - 3 to 4: Medium Risk
   - 6 to 9: High Risk

Worked Example:

Calculate Server Risk Level A company hosts a database server containing critical customer data. The likelihood of a successful SQL injection attack is estimated as Medium (2) because a Web Application Firewall is in place, but the application has outdated code. The impact of a successful breach is High (3) because sensitive customer records would be stolen. Calculate the Risk Level.

Step 1: Identify Likelihood

Likelihood = 2 (Medium)

Step 2: Identify Impact

Impact = 3 (High)

Step 3: Calculate Risk

Risk = Likelihood × Impact

Risk = $2 \times 3 = 6$

Step 4: Determine Severity

A risk score of 6 falls into the 6 to 9 range. Therefore, the overall severity is **High Risk**.

CHAPTER 3
Cyber Crime

3.1 Cyber Crime

In the context of computer science and digital forensics, dealing with cyber crime often requires structured methods for analyzing digital threats, evaluating vulnerabilities, and assessing the strength of security mechanisms. This section covers computational techniques for password analysis, risk assessment, and phishing detection.

3.1.1 Password Entropy Calculation

Password entropy is a measure of how unpredictable a password is, typically expressed in bits. It helps in evaluating the strength of passwords against brute-force cracking attacks.

Methodology:

Calculating Password Entropy To calculate the entropy E of a password:

1. Identify the pool size R of the character set used in the password:
   - Lowercase letters: 26
   - Uppercase letters: 26
   - Digits: 10
   - Special characters: 32

   Sum the sizes of the sets used. For example if a password uses lowercase and digits then $R = 26 + 10 = 36$.
2. Count the length of the password L (number of characters).
3. Calculate the entropy E using the formula:

$$E = L \times \log_2(R)$$

4. The resulting entropy E is measured in bits. A higher number indicates a stronger password.

Worked Example:

Calculate Password Entropy Calculate the entropy for the password `cYb3r!2024`.
Step 1: Identify the pool size R The password contains:

- Lowercase letters (`c, y, r`) $\rightarrow$ 26
- Uppercase letters (`Y`) $\rightarrow$ 26
- Digits (`3, 2, 0, 4`) $\rightarrow$ 10
- Special characters (`!`) $\rightarrow$ 32

Since it uses all four character sets, the pool size is $R = 26 + 26 + 10 + 32 = 94$.
Step 2: Count the length L The password `cYb3r!2024` has 10 characters. Thus $L = 10$.
Step 3: Calculate entropy E

$$E = 10 \times \log_2(94)$$

We know that $\log_2(94) \approx 6.554$.

$$E = 10 \times 6.554 = 65.54 \text{ bits}$$

Conclusion: The password has an entropy of approximately 65.54 bits which provides a moderate to strong level of security against brute-force attacks.

3.1.2 Cyber Risk Assessment Calculation

Quantitative risk assessment helps in understanding the potential financial impact of specific cyber crimes on an organization. This helps prioritize defensive measures.

Methodology:

Annualized Loss Expectancy Calculation To calculate the Annualized Loss Expectancy (ALE) for a cyber crime scenario:

1. Determine the Asset Value (AV) which is the total financial worth of the compromised asset.
2. Estimate the Exposure Factor (EF) representing the percentage of asset loss expected from a single incident (expressed as a decimal between 0 and 1).
3. Calculate the Single Loss Expectancy (SLE):

$$SLE = AV \times EF$$

4. Estimate the Annualized Rate of Occurrence (ARO) which is the estimated number of times the threat will occur in a year.
5. Calculate the Annualized Loss Expectancy (ALE):

$$ALE = SLE \times ARO$$

Worked Example:

Calculate Ransomware Attack Risk A company hosts a database containing customer data valued at \$500000. If a ransomware attack occurs it is estimated that 40% of the asset's value would be lost due to downtime and recovery efforts. Historical data suggests such an attack is likely to succeed once every 5 years. Calculate the SLE and ALE.
Step 1: Determine AV and EF Asset Value (AV) = \$500000 Exposure Factor (EF) = 40% = 0.40
Step 2: Calculate SLE

$$SLE = AV \times EF$$

$$SLE = 500000 \times 0.40 = 200000$$

The expected loss from a single ransomware attack is \$200000.
Step 3: Determine ARO The attack occurs once every 5 years so:

$$ARO = \frac{1}{5} = 0.20 \text{ times per year}$$

Step 4: Calculate ALE

$$ALE = SLE \times ARO$$

$$ALE = 200000 \times 0.20 = 40000$$

Conclusion: The company should anticipate an annualized loss of \$40000 due to ransomware attacks and should budget for security investments accordingly.

3.1.3 Phishing Email Detection Scoring

Detecting cyber crime activities like phishing often involves heuristic scoring based on specific indicators.

Methodology:

Phishing Indicator Scoring To compute a basic phishing threat score for an email:

1. Start with a Threat Score (TS) of 0.
2. Add points for each identified indicator:
   - Sender address domain does not match the organization (+3).
   - Email contains urgent language or threats (+2).
   - Email contains a mismatched URL (+4).
   - Email includes unexpected attachments (+3).
   - Email has poor spelling and grammar (+1).
3. Evaluate the final Threat Score:
   - $TS \geq 7$: High probability of phishing.
   - $4 \leq TS \leq 6$: Suspicious requires further investigation.
   - $TS \leq 3$: Likely benign.

Worked Example:

Evaluate Suspicious Email An employee receives an email claiming to be from "IT Support" with the sender address `admin@it-helpdesk-update.com` instead of the internal domain. The email reads: "Urgent! Your password expires in 2 hours. Click here to keep your same password." The link text "Click here" points to `http://192.168.100.45/login.php`. No attachments are present and the grammar is mostly correct. Evaluate the email.
Step 1: Initial Score Threat Score (TS) = 0.
Step 2: Evaluate Indicators

- Sender domain (`it-helpdesk-update.com`) does not match the internal organization domain: +3 points.
- Email contains urgent language ("Urgent! Your password expires in 2 hours"): +2 points.
- Mismatched/Suspicious URL (Points to a raw IP address instead of internal SSO): +4 points.
- Unexpected attachments: None (+0).
- Poor grammar: None (+0).

Step 3: Calculate Final Score

$$TS = 3 + 2 + 4 + 0 + 0 = 9$$

Step 4: Determine Classification Since $TS = 9$ which is ≥ 7 the email is classified as a **High probability of phishing**.

CHAPTER 4
Ethical Hacking

4.1 Ethical Hacking

Ethical hacking involves authorized attempts to gain unauthorized access to computer systems applications or data. In computational terms ethical hackers often rely on specific algorithms and formulas to scope network boundaries quantify vulnerabilities and assess password strength. This section covers the essential quantitative methods used in ethical hacking engagements.

4.1.1 Calculating Network Scoping and Scanning Ranges

Methodology:

Network Scoping and Target IP Calculation To properly scope an ethical hacking engagement one must determine the exact range of usable IP addresses in a given subnet (using CIDR notation).

1. Identify the IP address and the CIDR prefix P (e.g. /24).
2. Calculate the total number of IP addresses in the subnet: $N = 2^{32-P}$.
3. Identify the Network Address (first address) and Broadcast Address (last address) based on the block size.
4. Calculate the number of usable host IPs: $H = N - 2$.
5. Define the authorized scan range from Network Address + 1 to Broadcast Address - 1.

Worked Example:

Determine Scan Range for Subnet An ethical hacker is given the authorized scope `192.168.10.0/27`. Determine the total IP addresses the number of usable host IPs and the exact IP range to scan.
Solution:

1. The given CIDR prefix is $P = 27$.
2. Total IP addresses: $N = 2^{32-27} = 2^5 = 32$.
3. The network address is `192.168.10.0` (given).
4. The broadcast address is the network address plus 31 which is `192.168.10.31`.
5. Number of usable host IPs: $H = 32 - 2 = 30$.
6. The valid scan range consists of all usable hosts: `192.168.10.1` to `192.168.10.30`.

4.1.2 Vulnerability Scoring using CVSS

Methodology:

CVSS Base Score Calculation The Common Vulnerability Scoring System (CVSS) provides a numerical representation of vulnerability severity. A simplified version of the Base Score algorithm relies on the Impact Subscore (ISC) and Exploitability Subscore (ESC).

1. Determine ESC based on base metrics such as Attack Vector Attack Complexity Privileges Required and User Interaction.
2. Determine ISC based on Confidentiality Integrity and Availability metrics.
3. If Impact is 0 then the Base Score is 0 (the system is safe).
4. If Scope is Unchanged calculate the Base Score as: $\text{BaseScore} = \min(10.0, \text{ESC} + \text{ISC})$.
5. Apply standard rounding rules to represent the severity on a scale of 0.0 to 10.0.

Worked Example:

Calculate Simplified CVSS Base Score During a penetration test an ethical hacker discovers a vulnerability with an Exploitability Subscore of 3.9 and an Impact Subscore of 4.2 with an Unchanged Scope. Calculate the overall CVSS Base Score.
Solution:

1. Exploitability Subscore (ESC) = 3.9.
2. Impact Subscore (ISC) = 4.2.
3. Since ISC $\neq 0$ we proceed with the calculation formula.
4. $\text{BaseScore} = \min(10.0, 3.9 + 4.2) = \min(10.0, 8.1) = 8.1$.
5. The simplified CVSS Base Score is 8.1 indicating a High Severity vulnerability.

4.1.3 Password Cracking and Entropy

Methodology:

Password Entropy and Cracking Time Password entropy measures the unpredictability of a password indicating how resistant it is to brute-force cracking attempts.

1. Identify the character pool size S (e.g. 26 for lowercase letters 10 for digits).
2. Identify the password length L.
3. Calculate Entropy E in bits using the formula: $E = L \times \log_2(S)$.
4. Calculate total possible combinations $C = S^L$.
5. Given a cracking speed R (measured in hashes per second) calculate the maximum time to crack T in seconds: $T = \frac{C}{R}$.

Worked Example:

Estimating Brute Force Cracking Time An ethical hacker recovers a password hash. The target application's password policy enforces exactly 8 characters using only lowercase English letters. The ethical hacker's cracking rig can test 10^9 hashes per second. Calculate the password entropy total combinations and the maximum time required to crack the password.

Solution:

1. Character pool size $S = 26$ (all lowercase letters).
2. Password length $L = 8$.
3. Entropy $E = 8 \times \log_2(26) \approx 8 \times 4.70 \approx 37.6$ bits.
4. Total possible combinations $C = 26^8 \approx 2.088 \times 10^{11}$.
5. Cracking speed $R = 10^9$ hashes/second.
6. Maximum time $T = \frac{2.088 \times 10^{11}}{10^9} = 208.8$ seconds.
7. It will take a maximum of 208.8 seconds (approximately 3.5 minutes) to guarantee a successful crack via brute-force.

CHAPTER 5
Digital Forensics

5.1 Digital Forensics

Digital forensics is the process of uncovering and interpreting electronic data to preserve evidence in its most original form. In this section, we focus strictly on the computational and analytical techniques applied by forensic investigators: verifying data integrity, extracting hidden information via steganography, carving files from raw disk dumps, and chronologically analyzing file metadata.

5.1.1 File Integrity Verification using Checksums

To ensure that digital evidence has not been tampered with during acquisition or analysis, investigators compute a checksum or cryptographic hash. A basic computational model for this is the sequential XOR checksum.

Methodology:

XOR Checksum Calculation To verify data integrity using an 8-bit XOR checksum:

1. Divide the digital evidence data into fixed-length blocks (e.g. 8-bit bytes).
2. Take the first byte and perform a bitwise XOR operation with the second byte.
3. Take the resulting byte and perform a bitwise XOR with the third byte.
4. Repeat this process until all bytes in the data stream have been processed.
5. The final resulting byte is the checksum. Compare this with the original checksum to verify integrity. If they differ, the data has been altered.

Worked Example:

Computing and Verifying XOR Checksums An investigator securely transmits three bytes of critical evidence data along with its original 8-bit XOR checksum. The original checksum is known to be `0x70`. The data received is: `0x12 0x35 0x56`. Determine if the data has been tampered with.

Step 1: Convert received hex bytes to binary.

$$\begin{aligned} \texttt{0x12} &= \texttt{0001 0010} \\ \texttt{0x35} &= \texttt{0011 0101} \\ \texttt{0x56} &= \texttt{0101 0110} \end{aligned}$$

Step 2: XOR the first and second bytes.

$$\begin{array}{rl} & \texttt{0001 0010 (0x12)} \\ \oplus & \texttt{0011 0101 (0x35)} \\ \hline & \texttt{0010 0111 (0x27)} \end{array}$$

Step 3: XOR the result with the third byte.

$$\begin{array}{r} \texttt{0010 0111 (0x27)} \\ \oplus\ \texttt{0101 0110 (0x56)} \\ \hline \texttt{0111 0001 (0x71)} \end{array}$$

Conclusion: The computed checksum of the received data is `0x71`. Since the original checksum was `0x70`, the checksums do not match. The evidence has been tampered with or corrupted during transfer.

5.1.2 Least Significant Bit Steganography Extraction

Steganography is the practice of hiding a secret message inside an ordinary file. The most common computational method is Least Significant Bit (LSB) steganography in images, where the 8th bit of each pixel value is replaced with a bit from the secret message.

Methodology:

LSB Extraction Algorithm To extract a hidden message encoded via LSB in an 8-bit grayscale image:

1. Identify the sequence of pixel values (in decimal or hexadecimal).
2. Convert each pixel value into its 8-bit binary representation.
3. Extract the rightmost bit (the Least Significant Bit) from each binary number.
4. Group the extracted bits sequentially into 8-bit blocks (bytes).
5. Convert each 8-bit block back to its decimal equivalent.
6. Map the decimal value to its corresponding ASCII character to reveal the message.

Worked Example:

Extracting Hidden Text from Pixels A forensic analyst suspects an image contains hidden text. A sequence of 8 grayscale pixel values is extracted from the image: 154, 155, 152, 153, 154, 154, 155, 154. Extract the hidden ASCII character.

Step 1 & 2: Convert to binary and extract the LSB.

Pixel (Decimal)	Binary Value	LSB
154	1001101**0**	0
155	1001101**1**	1
152	1001100**0**	0
153	1001100**1**	1
154	1001101**0**	0
154	1001101**0**	0
155	1001101**1**	1
154	1001101**0**	0

Step 3: Group the extracted bits into an 8-bit block. Reading the LSBs from top to bottom yields the binary sequence: `01010010`.

Step 4: Convert the binary block to decimal.

$$(01010010)_2 = (0 \times 128) + (1 \times 64) + (0 \times 32) + (1 \times 16) + (0 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1)$$

$$= 64 + 16 + 2 = 82$$

Step 5: Convert decimal to ASCII. The decimal value 82 corresponds to the uppercase ASCII character **'R'**.

5.1.3 File Carving via Magic Numbers

When file systems are corrupted or files are deleted, the file tables (like FAT or MFT) may be lost. Forensic investigators use "file carving" to recover files directly from the raw disk hex dump by looking for known file signatures, often called "magic numbers."

Methodology:

Hexadecimal Signature File Carving

1. Scan the raw hexadecimal dump of the disk byte by byte.
2. Search for the standard Header Signature (Magic Number) that indicates the beginning of a specific file type.
3. Once the header is found, record its start address (offset).
4. Search forward from the header to locate the standard Footer Signature that denotes the end of the file.
5. Record the end address of the footer.
6. Extract all bytes from the start address to the end address, inclusive, and save them as a recovered file.

Worked Example:

Carving a JPEG File from a Hex Dump An analyst is reviewing a raw memory dump and needs to carve out a deleted JPEG image. The standard JPEG header is `FF D8 FF E0` and the footer is `FF D9`. Find the start offset, end offset, and total size of the carved file in bytes from the snippet below.

Offset	Hex Data
0000	00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
0010	FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 48
0020	00 48 00 00 FF DB 00 43 00 01 02 03 04 05 06 07
0030	3A 4B 5C 6D 7E 8F 90 A1 B2 C3 D4 E5 FF D9 00 11

Step 1: Locate the Header. Scanning the data, the sequence `FF D8 FF E0` begins at offset `0010`. The start address is `0010`.
Step 2: Locate the Footer. Continuing from the header, the sequence `FF D9` appears in the last row. The first byte of the row is at `0030`. Let's count the columns: `3A` (0), `4B` (1), `5C` (2), `6D` (3), `7E` (4), `8F` (5), `90` (6), `A1` (7), `B2` (8), `C3` (9), `D4` (A), `E5` (B), `FF` (C), `D9` (D). The sequence `FF D9` ends at offset `003D`. The end address is `003D`.
Step 3: Calculate the Size. The size of the carved file in bytes is:

$$\text{Size} = (\text{End Offset} - \text{Start Offset}) + 1$$

$$\text{Size} = (\texttt{003D})_{16} - (\texttt{0010})_{16} + 1$$

Convert to decimal: $\texttt{003D} = (3 \times 16) + 13 = 61$. $\texttt{0010} = (1 \times 16) + 0 = 16$.

$$\text{Size} = (61 - 16) + 1 = 45 + 1 = 46 \text{ bytes.}$$

The JPEG image is successfully carved from offset `0x0010` to `0x003D`, totaling 46 bytes.

5.1.4 Timeline Analysis and Timestomping Detection

File System Timeline Analysis involves extracting MAC (Modified, Accessed, Created) timestamps to reconstruct the sequence of events during a cyber attack. Malicious actors frequently alter timestamps (a technique known as "Timestomping") to hide their activity.

Methodology:

Detecting Anomalous Timestamps

1. Extract the MAC timestamps (Modified, Accessed, Created) for all files in the target directory.
2. Standardize all times to a uniform 24-hour chronological format.
3. Apply the Logical Time Rule: Under normal operating system conditions, a file's Creation Time (T_C) must be less than or equal to its Modification Time (T_M), and both are typically less than or equal to its Last Accessed Time (T_A). So, $T_C \leq T_M$ and $T_C \leq T_A$.
4. Flag any file where $T_M < T_C$ (the file was modified before it was created) as a definitive anomaly and a strong indicator of timestomping.

Worked Example:

Identifying Timestomped Files An incident responder dumps the following MAC times from a compromised web server's upload directory. Evaluate the timestamps to identify which file was subjected to timestomping.

Filename	Created (T_C)	Modified (T_M)	Accessed (T_A)
`index.php`	08:00:00	08:05:00	08:30:00
`config.bak`	09:15:00	09:15:00	09:45:00
`shell.php`	14:30:00	10:10:00	14:35:00

Step 1: Evaluate `index.php` T_C (08:00) $\leq T_M$ (08:05) $\leq T_A$ (08:30). The timeline logically follows normal creation, editing, and reading patterns. No anomaly.
Step 2: Evaluate `config.bak` T_C (09:15) $= T_M$ (09:15) $\leq T_A$ (09:45). The file was created and last modified simultaneously, then read later. This is completely normal behavior for copying or creating a new file. No anomaly.
Step 3: Evaluate `shell.php` T_C (14:30) and T_M (10:10). Here, $T_M < T_C$. The modification time (10:10:00) is more than four hours *before* the creation time (14:30:00). It is logically impossible for a file to be modified before it is created on the file system.
Conclusion: The file `shell.php` has highly anomalous timestamps, indicating that an attacker likely used a timestomping tool to roll back the modification date in an attempt to evade forensic detection.

Appendix: Key Formulae

This appendix provides a quick reference to the key abstract formulae and mathematical relationships discussed in this book, categorized by unit.

Unit 1: Cryptography Basics

Caesar Cipher

- **Encryption:**

$$C = (P + K) \pmod{26}$$

  Where C is the ciphertext letter (represented as an integer 0–25), P is the plaintext letter (0–25), and K is the key or shift value.

- **Decryption:**

$$P = (C - K) \pmod{26}$$

  Where P is the recovered plaintext, C is the ciphertext, and K is the key.

Unit 2: Risk Assessment and Networking

Risk Assessment

- **Risk Calculation:**

$$\text{Risk} = \text{Likelihood} \times \text{Impact}$$

  A fundamental formula for evaluating the overall risk of a threat based on how likely it is to occur and the severity of its impact.

Unit 3: Quantitative Risk Analysis

Loss Expectancy

- **Single Loss Expectancy (SLE):**

$$SLE = AV \times EF$$

  Where AV is the Asset Value and EF is the Exposure Factor (the percentage of the asset's value lost during a single incident).

- **Annualized Loss Expectancy (ALE):**

$$ALE = SLE \times ARO$$

  Where SLE is the Single Loss Expectancy and ARO is the Annualized Rate of Occurrence (how many times the incident is expected to happen per year).

Unit 4: Password Security and Subnetting

Password Security

- **Total Combinations (Password Space):**

$$C = S^L$$

  Where S is the size of the character set (symbol space) and L is the password length.

- **Password Entropy:**

$$E = L \times \log_2(S)$$

  Where E is the entropy measured in bits, L is the password length, and S is the size of the symbol space.

- **Time to Crack:**

$$T = \frac{C}{R}$$

  Where C is the total number of combinations and R is the cracking rate (guesses per time unit).

Networking

- **Usable Hosts in a Subnet:**

$$H = N - 2$$

  Where N is the total number of IP addresses in the subnet (calculated as $2^{\text{host bits}}$), and we subtract 2 to account for the network address and the broadcast address.

Unit 5: Data Analysis

Data Size Calculation

- **Data Size (in bytes):**

$$\text{Size} = (\text{End Offset} - \text{Start Offset}) + 1$$

  Used to calculate the total size of a block of data given its starting and ending offsets (inclusive) within a file or memory dump.