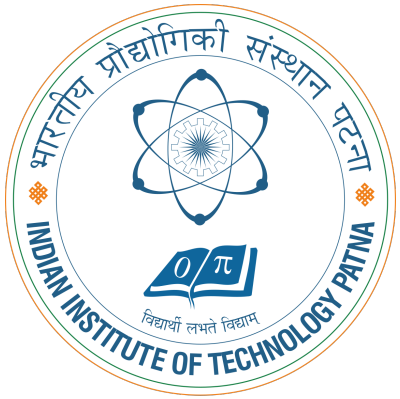
Chapter Numerical Problems

Dr. Mayank Agarwal

Department of CSE
IIT Patna

CS457 Big Data Security
Jan-April 2026



Numerical Problem : CIA Triad Prioritization I

- **Problem:** A hospital's electronic health record (EHR) system requires 99.999% uptime for emergency access. Patient data must be accurate and unmodified. Unauthorized disclosure is prohibited by law. Assign percentage weights to Confidentiality (C), Integrity (I), and Availability (A) that sum to 100%. Justify your weights.
- **Solution:**
  - Availability: 50% (Life-critical emergency access)
  - Integrity: 35% (Medical decisions require accurate data)
  - Confidentiality: 15% (HIPAA compliance, but less critical than A/I)

Emergency situations make availability paramount. Incorrect data (integrity failure) can cause death. Confidentiality violations cause legal/financial harm but not immediate death. Weights may vary by hospital policy.

Numerical Problem : Risk Calculation I

- **Problem:** A company estimates that a database breach will cost $5,000,000. The vulnerability has been present for 3 years. Historical data shows similar systems are breached once every 10 years. Calculate the Annualized Loss Expectancy (ALE).
- **Solution:**
  - Single Loss Expectancy (SLE) = $5,000,000
  - Annual Rate of Occurrence (ARO) = $1/10 = 0.1$
  - ALE = SLE $\times$ ARO = $5,000,000 $\times$ 0.1 = $500,000 per year

The company should spend up to $500,000 annually on controls to mitigate this risk. This is the fundamental economics of security investment. Controls costing less than the ALE are cost-justified.

Numerical Problem : Password Entropy Calculation I

- **Problem:** Calculate the entropy (in bits) for: (a) 8-character random password using 26 lowercase letters, (b) 8-character random password using 95 printable ASCII characters. Which is stronger and by what factor?
- **Solution:**
  - (a) $E = \log_2(26^8) = 8 \times \log_2(26) = 8 \times 4.70 = 37.6$ bits
  - (b) $E = \log_2(95^8) = 8 \times \log_2(95) = 8 \times 6.57 = 52.6$ bits is $95^8/26^8 = (95/26)^8 \approx 15,000\times$ stronger

Entropy measures the number of guesses needed to guarantee cracking. Each additional bit doubles the difficulty. ASCII passwords provide 15 bits more entropy than lowercase-only passwords.

Numerical Problem : Password Cracking Time I

- **Problem:** An attacker can test 1 billion passwords per second. How long to crack a 10-character password from: (a) 26 lowercase, (b) 95 ASCII? Assume worst-case (all combinations tested).
- **Solution:**
  - (a) $26^{10} = 1.41 \times 10^{14}$ combinations
  - Time $= 1.41 \times 10^{14}/10^{9} = 141,000$ seconds ˜ 39 hours
  - (b) $95^{10} = 5.99 \times 10^{19}$ combinations
  - Time $= 5.99 \times 10^{19}/10^{9} = 5.99 \times 10^{10}$ seconds ˜ 1,900 years

This demonstrates why complexity requirements matter. Modern GPUs can achieve billions of hashes per second for weak algorithms (MD5, NTLM). Strong hashing (bcrypt) reduces attempts to thousands per second.

Numerical Problem : Salt Effectiveness I

- **Problem:** A password database has 10 million users. Without salt, precomputing a rainbow table for all 95-char, 8-char passwords requires 95^8 hashes (˜ 10 TB storage). With 32-bit salt, how much storage is needed for precomputation? Is this feasible?
- **Solution:**
  - Rainbow table for one salt value: 95^8 hashes
  - Number of possible salts: $2^{32} = 4.29 \times 10^9$
  - Total hashes needed: $95^8 \times 2^{32} \approx 6.6 \times 10^{28}$ hashes
  - Storage ˜ $6.6 \times 10^{28} \times 16$ bytes ˜ 10^{30} bytes (impossible)

Salt forces attackers to crack each password individually. Even weak passwords become expensive to crack when properly salted. This is why salted hashes are the standard.

Numerical Problem : Multi-Factor Authentication Risk Reduction I

- **Problem:** Password phishing success rate = 30%. SMS 2FA interception rate = 2%. Hardware token cloning rate = 0.01%. Calculate the effective success rate for an attacker targeting (a) password only, (b) password + SMS, (c) password + hardware token.
- **Solution:**
  - (a) P(success) = $0.30 = 30\%$
  - (b) P(success) = $0.30 \times 0.02 = 0.006 = 0.6\%$
  - (c) P(success) = $0.30 \times 0.0001 = 0.00003 = 0.003\%$

MFA multiplies risk reduction factors. Hardware tokens provide phishing-resistant authentication because they cannot be intercepted remotely. This is why security keys (WebAuthn) are the gold standard.

Numerical Problem : RSA Key Strength Scaling I

- **Problem:** Breaking 512-bit RSA requires ~ 2^{60} operations. Breaking 1024-bit RSA requires ~ 2^{75} operations. Breaking 2048-bit RSA requires ~ 2^{90} operations. How many times harder is 2048-bit vs 1024-bit? Express in years if 2^{60} operations $= 1$ hour.
- **Solution:**
  - Ratio $= 2^{90}/2^{75} = 2^{15} = 32,768\times$ harder
  - 2^{60} ops $= 1$ hour
  - 2^{75} ops $= 2^{15}$ hours $= 32{,}768$ hours ~ 3.74 years
  - 2^{90} ops $= 2^{30}$ hours $= 1{,}073{,}741{,}824$ hours ~ 122,500 years

RSA key strength does not scale linearly with bit length. Each additional bit doubles the factorization difficulty. 2048-bit RSA is currently considered secure; 1024-bit was deprecated years ago.

Numerical Problem : Integer Overflow Example I

- **Problem:** In a 32-bit unsigned integer system, `len = 0xFFFFFFF7`. The code does: `size = len + 0x0C`. Show the calculation and explain the security implication.
- **Solution:**
  - 0xFFFFFFF7 = 4,294,967,287
  - 0xFFFFFFF7 + 0x0C = 0x100000003
  - 32-bit truncation = 0x00000003 (value: 3)
  - Buffer allocated: 3 bytes
  - Memory copy uses original len: 4,294,967,287 bytes
  - Buffer overflow by 4GB = 0xFFFFFFF7 - 0x00000003 = 0xFFFFFFF4 bytes = 4,294,967,284 bytes, which is 4 GB.

This is a classic integer overflow leading to buffer overflow. The check passes (3 = MAX) but the copy operation overflows massively. Always validate before arithmetic, not after.

Numerical Problem : XOR Encryption Analysis I

- **Problem:** Given plaintext "SECURITY" in ASCII hex: 53 45 43 55 52 49 54 59. Ciphertext: 3A 70 7C 6A 6D 76 6B 66. Find the XOR key. If this key is reused for another message, what is the security implication?
- **Solution:**
  - XOR each byte: $53 \oplus 3A = 69$, $45 \oplus 70 = 35$, $43 \oplus 7C = 3F$, $55 \oplus 6A = 3F$
  - $52 \oplus 6D = 3F$, $49 \oplus 76 = 3F$, $54 \oplus 6B = 3F$, $59 \oplus 66 = 3F$
  - Key = 69 35 3F 3F 3F 3F 3F 3F (repeating pattern suspected)
  - Actual key = 0x69,0x35,0x3F,0x3F,0x3F,0x3F... but shows pattern leakage
  - Reused key enables known-plaintext attacks and crib dragging.

XOR encryption with reused key is broken. Modern cryptography uses unique keys per message and authenticated encryption to prevent manipulation and analysis. Even if the attacker doesn't know plaintexts, they can guess common words ("HTTP", "GET", "From:", "password", spaces, etc.) and slide that guess across the XOR result

Numerical Problem : TCP SYN Flood Attack I

- **Problem:** A server has 65,536 ports and maintains connection state for 75 seconds. Each SYN packet creates 256 bytes of state memory. Attacker sends 10,000 SYN packets per second. (a) How long until memory exhaustion if server has 1GB RAM? (b) How many spoofed IPs needed?
- **Solution:**
  - State created per second:

$$10{,}000 \times 256 = 2{,}560{,}000 \text{ bytes/s} \approx 2.56 \text{ MB/s}$$

  - Half-open connections held (75s timeout):

$$10{,}000 \times 75 = 750{,}000 \text{ entries}$$

  - State held at steady state:

$$750{,}000 \times 256 = 192{,}000{,}000 \text{ bytes} \approx 192 \text{ MB}$$

Numerical Problem : TCP SYN Flood Attack II

- **(a) Memory exhaustion (1GB):** *Not reached under these assumptions.* Memory rises for ~ 75 seconds then stabilizes around ~ 192 MB due to timeouts.
- **(b) Spoofed IPs needed:** Up to $750{,}000$ distinct source IPs to make all entries appear unique (helps evade per-IP limits), though fewer may suffice depending on defenses.

SYN cookies mitigate SYN floods by avoiding per-connection state until the final ACK arrives.

Numerical Problem : DDoS Amplification Factor I

- **Problem:** A DNS amplification attack uses 60-byte queries to generate 4,000-byte responses. Attacker has 1 Gbps botnet bandwidth. What is the attack traffic volume after amplification? How many targets can be saturated at 10 Gbps each?
- **Solution:**
  - Amplification factor = 4,000 / 60 = 66.7×
  - Attack bandwidth = 1 Gbps × 66.7 = 66.7 Gbps
  - 10 Gbps targets = 66.7 / 10 = 6.67 (6 targets saturated)

Amplification attacks are dangerous because they multiply the attacker's limited bandwidth. Defense requires disabling open resolvers, response rate limiting, and BCP38 to prevent spoofing.

Numerical Problem : Access Control Matrix I

- **Problem:** 500 users, 10,000 files. Represented as ACL vs Capabilities. Assuming average 100 users per file for ACL, and 200 files per user for capabilities. Calculate storage for (a) ACL, (b) Capabilities if each entry requires 8 bytes.
- **Solution:**
  - ACL (Access Control List) : object-centered
  - Each file stores a list: "which users can access me".
  - Capabilities : subject-centered
  - Each user stores a list: "which files can I access"
  - ACL: 10,000 files $\times$ 100 entries $\times$ 8 bytes = 8,000,000 bytes ˜ 7.63 MB
  - Capabilities: 500 users $\times$ 200 entries $\times$ 8 bytes = 800,000 bytes ˜ 0.76 MB
  - Capabilities use 10$\times$ less storage for this access pattern

ACLs are efficient when many users share access to few files; capabilities when many files per user. Most systems use ACLs because revocation is easier. Capabilities need revocation mechanisms (indirection, expiration).

Numerical Problem : CSRF Token Entropy I

- **Problem:** A CSRF token is 128 bits. Attacker can make 1000 requests per second. How many years to brute force the token? Assume token is not rate-limited.
- **Solution:**
  - Total combinations $= 2^{128} \approx 3.4 \times 10^{38}$
  - Attempts per year $=$ $1000 \times 60 \times 60 \times 24 \times 365$ ~ 3.15×10^{10}
  - Years to exhaust $= 3.4 \times 10^{38} / 3.15 \times 10^{10} \approx 1.08 \times 10^{28}$ years

128-bit CSRF tokens are cryptographically secure against brute force. Actual risk is token theft (XSS) or predictable token generation. Use cryptographically secure random generators.

Numerical Problem : Clickjacking Probability I

- **Problem:** Clickjacking attack overlays a transparent iframe over a decoy button. Target button is 200×80 pixels. Decoy button is 220×100 pixels. If user clicks randomly within decoy, probability of hitting target? If attacker needs 100 successful clicks, how many clicks expected?
- **Solution:**
  - Target area = 200 × 80 = 16,000 pixels
  - Decoy area = 220 × 100 = 22,000 pixels
  - P(hit) = 16,000 / 22,000 = 0.727 = 72.7%
  - Expected clicks for 100 hits = 100 / 0.727 ˜ 138 clicks

Clickjacking relies on high probability of successful clicks. Defense: X-Frame-Options or CSP frame-ancestors prevents framing. UI redress analysis tools detect these attacks.