CNS QP CAT 2 Answers

1.B Explain in detail about Euler's Totient function?

Ans.1.B)

Euler's Totient Function, often denoted by φ(n), is a mathematical function that counts the number of positive integers less than or equal to n that are coprime (relatively prime) to n. The term "coprime" refers to numbers that have no common factors other than 1.

The Euler's Totient Function is named after the Swiss mathematician Leonhard Euler, who made significant contributions to number theory.

Euler's Totient Function Formula:

$$\varphi(n) = n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_k}\right)$$

Here,

n is a positive integer, and $p_1, p_2, \ldots, p_k$ are the distinct prime factors of *n*.

Properties and Observations:

1. Euler's Product Formula:
   - The formula expresses $\phi(n)$ as a product over the prime factors of n.
2. Coprime Relationship:
   - $\phi(n)$ gives the count of positive integers coprime to *n*.
3. Special Cases:
   - For a prime number $\phi(p)=p-1$, as all numbers less than p are coprime to p.
   - For $n=p \cdot q$ (product of two distinct primes), $\phi(n)=(p-1) \cdot (q-1)$.
4. Euler's Theorem:
   - Euler's Totient Function is used in Euler's Totient Theorem, which states that $a^{\phi(n)} \equiv 1 \bmod n$
     if *a* and *n* are coprime.

1.C Illustrate the working RSA algorithm with suitable example.

Ans.1.C)

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client’s public key and sends the encrypted data.
3. The client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.

2.B) In public key system using RSA you intercept the cipher text C = 10 sent

to the user whose public key is e = 5, n = 35, what is the plaintext M?

Ans.2.B)

In the RSA algorithm, the encryption and decryption processes are based on the modular exponentiation operation. The encryption operation is defined as $C \equiv M^e \bmod n$, where C is the ciphertext, M is the plaintext, e is the public exponent, and n is the modulus.

Given C=10, e=5, and n=35, the goal is to find the plaintext M.

The decryption operation is defined as $M \equiv C^d \bmod n$, where d is the private exponent. In RSA, d is calculated as the modular multiplicative inverse of e modulo (n), where $\phi(n)$ is Euler's totient function.

To find d, we need to calculate $\phi(n)$. For n=35, $\phi(35)$ is calculated as follows:

$\phi(35) = \phi(5\times7) = (5-1) \times (7-1) = 4\times6 = 24$

Now, find d such that $5 \times d \equiv 1 \bmod 24$.

In this case, d=5 because $5 \times 5 \equiv 1 \bmod 24$.

Now, use d to decrypt the ciphertext C=10 and find the plaintext M:

$M \equiv C^d \bmod n$

$M \equiv 10^5 \bmod 35$

Calculate M:

$M \equiv 100000 \bmod 35$

$M \equiv 25 \bmod 35$

So, the plaintext M is 25.

2.C) Demonstrate the working of Diffie-Hellman key exchange algorithm with suitable example.

Ans.2.C)

The Diffie-Hellman key exchange algorithm is a method for two parties to securely agree on a shared secret key over an insecure communication channel. The security of the algorithm relies on the difficulty of the discrete logarithm problem.

Here's a simplified example of the Diffie-Hellman key exchange algorithm:
Steps:

1. Initialization:
    - Choose a large prime number *p* and a primitive root modulo *p*, denoted as *g*. These values are public and agreed upon by both parties.

Let's choose *p*=23 and *g*=5 for this example.

2. Public Key Exchange:

- Alice and Bob publicly share their choices of *p* and *g*.
- Alice chooses a private key *a* (a random number), and Bob chooses a private key *b*.

Alice's private key: *a*=6

Bob's private key: *b*=15

3. Compute Partial Keys:

- Both Alice and Bob independently compute their partial keys using the public values *p* and *g* and their private keys *a* and *b*.

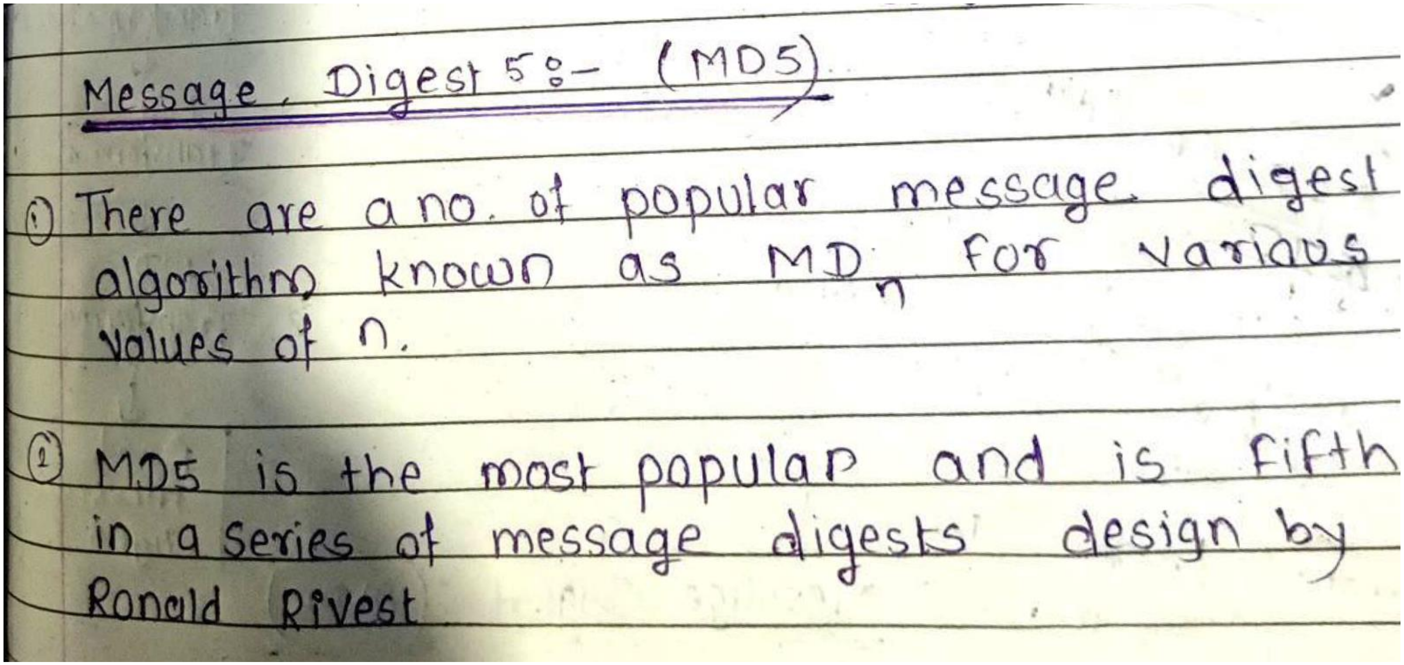
Alice's partial key: A = g^a mod p = 5^6 mod 23 = 8

Bob's partial key: B = g^b mod p = 5^{15} mod 23 = 19

Now, both Alice and Bob have computed the same shared secret key (*s*=2) without explicitly exchanging it over the insecure channel. This shared secret can be used as a symmetric key for encryption and decryption in subsequent communication.

3.B Explain in detail about MD5 Message Digest Algorithm.

Ans.3.B)



Message Digest 5 :- (MD5)

(1) There are a no. of popular message digest algorithms known as MD_n for various values of n.

(2) MD5 is the most popular and is fifth in a series of message digests design by Ronald Rivest

This algorithm operates on message 512 bits at a time.
Message not multiple of 512 bits are padded with :

(i) A string consisting of 1 followed by zeroes and
(ii) 64-bit integer that indicates the the length of original message, to make the length of the composite message multiples of 512 bits.

| original message | padding |

1000 bits

$512 \times 1 = 512$
$512 \times 2 = 1024$
$512 \times 3 = 1536$

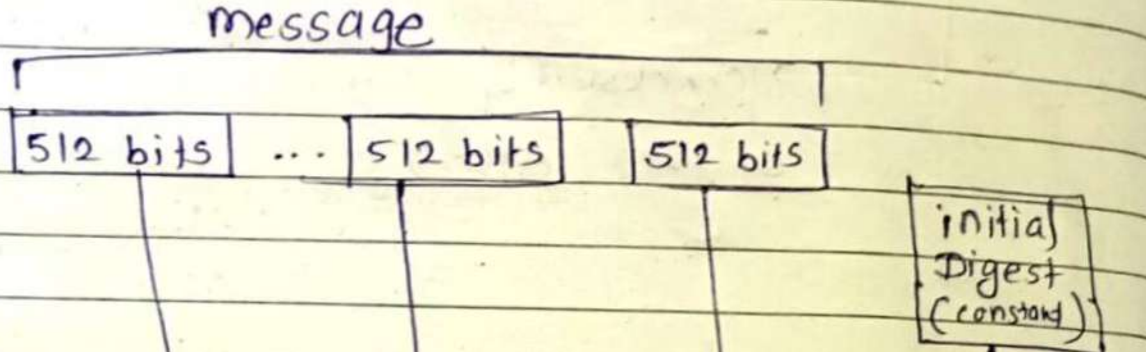
$$\begin{array}{r} 1536 \\ -\ 64 \\ \hline 1472 \end{array}$$

$$\begin{array}{r} 1000 \\ +\ 472 \\ \hline 1472 \end{array}$$

(padding add)

64 bit less than exact multiple of 512

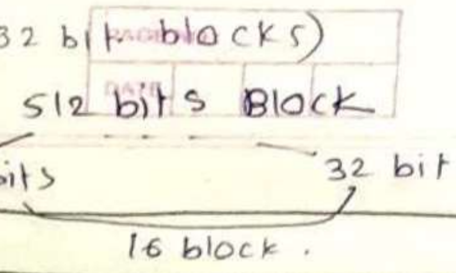
| original msg | padding | length |



Message Digest 5 (MD5)

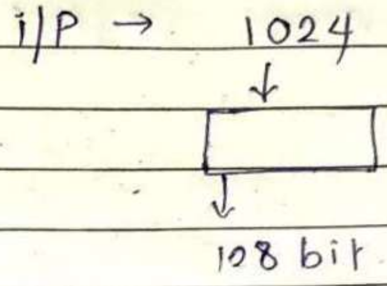
(ii) append original length bet^n padding (modulo 64)
1000 | length mod 2^{64}
(iii) divide it in 512 bit blocks.

process Blocks
→ copy four chaining variables into corresponding variables.
{A = a, B = b, C = c, D = d}
→ divide 512 bit block into 16 (32 bit blocks)
→ four rounds

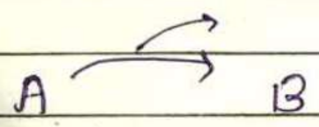


MD5 Algorithm

1. cryptographic hash function algorithm
2. by Ronald Rivest in 1992
3. a Series of message digest
4. Digest size – 128 bit.
5. block size – 512 bit.
6. No. of Rounds – 4



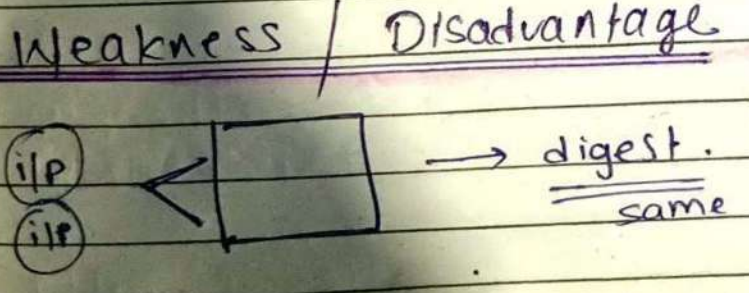
purpose → integrity check



Applications / Uses of MD5

1. secure passwords of users Hello ⟶ 1a954
2. in 128 bit format
3. to verify data integrity
4. used for file authentication

Weakness / Disadvantage



even a small change in msg will result in a diff. hash value

3.C Explain the Hash Functions and their Security.

Ans.3.C)

A hash function is a mathematical function that takes an input (or 'message') and produces a fixed-size string of characters, which is typically a hash value or hash code. The output, often a 'digest,' is unique to the specific input, and even a small change in the input should produce a substantially different hash.

Properties of Hash Functions:

1. Deterministic:
   - For the same input, the hash function will always produce the same output.
2. Fixed Output Length:
   - The output of the hash function is of a fixed size, regardless of the input size.
3. Efficient to Compute:
   - It should be computationally efficient to calculate the hash value for any given input.
4. Pre-image Resistance:
   - Given a hash value, it should be computationally infeasible to reverse the process and find an input that produces that hash.
5. Collision Resistance:
   - It should be unlikely that two different inputs produce the same hash value.

Security of Hash Functions:

1. Data Integrity:
   - Hash functions are widely used to ensure data integrity. By comparing hash values before and after transmission or storage, one can verify if the data has been altered.
2. Password Hashing:
   - Hash functions are used to store passwords securely. Rather than storing plaintext passwords, systems store the hash of the password. This way, even if the hash is compromised, the original password is not easily obtainable.
3. Digital Signatures:
   - In digital signatures, hash functions are used to generate a fixed-size representation of a message, which is then signed. The recipient can use the sender's public key to verify the signature.
4. Cryptographic Applications:
   - Hash functions are essential in various cryptographic protocols, including HMAC (Hash-based Message Authentication Code), digital certificates, and blockchain.
5. Blockchain Technology:
   - Hash functions play a crucial role in creating a chain of blocks in blockchain. The hash of a block is included in the subsequent block, ensuring the integrity of the entire chain.

Security Concerns:

1. Collision Attacks:
   - A collision occurs when two different inputs produce the same hash output. While hash functions aim to minimize the likelihood of collisions, the existence of collision-resistant hash functions is a topic of ongoing research.
2. Length Extension Attacks:
   - Some hash functions are vulnerable to length extension attacks, where an attacker can extend the hash value without knowing the original input.
3. Algorithmic Vulnerabilities:
   - Cryptographic hash functions need to resist various attacks, such as birthday attacks and differential cryptanalysis.

Common Hash Functions:

1. MD5 (Message Digest Algorithm 5):
   - Previously widely used but now considered insecure due to vulnerabilities.
2. SHA-1 (Secure Hash Algorithm 1):
   - Also deprecated due to vulnerabilities; SHA-256 and SHA-3 are more secure alternatives.
3. SHA-256 (Secure Hash Algorithm 256-bit):
   - Part of the SHA-2 family, commonly used in blockchain and other security protocols.
4. SHA-3 (Secure Hash Algorithm 3):
   - The latest member of the Secure Hash Algorithm family, designed to provide the same security as SHA-2.

4.B Explain the concept of Kerberos.

Ans.4.B)

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. In Kerberos Authentication server and database is used for client authentication. Kerberos runs as a third-party trusted server known as the Key Distribution Center (KDC). Each user and service on the network is a principal.

The main components of Kerberos are:

- Authentication Server (AS):
  The Authentication Server performs the initial authentication and ticket for Ticket Granting Service.
- Database:
  The Authentication Server verifies the access rights of users in the database.
- Ticket Granting Server (TGS):
  The Ticket Granting Server issues the ticket for the Server.

Kerberos Limitations

- Each network service must be modified individually for use with Kerberos.
- It doesn’t work well in a timeshare environment.
- Secured Kerberos Server.
- Requires an always-on Kerberos server.
- Stores all passwords are encrypted with a single key.
- Assumes workstations are secure.
- May result in cascading loss of trust.
- Scalability

Applications of Kerberos

i. User Authentication
ii. Single Sign-On (SSO)
iii. Mutual Authentication
iv. Authorization
v. Network Security

4.C Explain in detail about X.S09 directory authentication service..

Ans.4.C)

X.509 is a digital certificate that is built on top of a widely trusted standard known as ITU or International Telecommunication Union X.509 standard, in which the format of PKI certificates is defined. X.509 digital certificate is a certificate-based authentication security framework that can be used for providing secure transaction processing and private information. These are primarily used for handling the security and identity in computer networking and internet-based communications.

Working of X.509 Authentication Service Certificate:

The core of the X.509 authentication service is the public key certificate connected to each user. These user certificates are assumed to be produced by some trusted certification authority and positioned in the directory by the user or the certified authority. These directory servers are only used for providing an effortless reachable location for all users so that they can acquire certificates. X.509 standard is built on an IDL known as ASN.1. With the help of Abstract Syntax Notation, the X.509 certificate format uses an associated public and private key pair for encrypting and decrypting a message.

Once an X.509 certificate is provided to a user by the certified authority, that certificate is attached to it like an identity card. The chances of someone stealing it or losing it are less, unlike other unsecured passwords. With the help of this analogy, it is easier to imagine how this authentication works: the certificate is basically presented like an identity at the resource that requires authentication.

Generally, the certificate includes the elements given below:

- **Version number:** It defines the X.509 version that concerns the certificate.
- **Serial number:** It is the unique number that the certified authority issues.
- **Signature Algorithm Identifier:** This is the algorithm that is used for signing the certificate.
- **Issuer name:** Tells about the X.500 name of the certified authority which signed and created the certificate.
- **Period of Validity:** It defines the period for which the certificate is valid.
- **Subject Name:** Tells about the name of the user to whom this certificate has been issued.
- **Subject's public key information:** It defines the subject's public key along with an identifier of the algorithm for which this key is supposed to be used.
- **Extension block:** This field contains additional standard information.
- **Signature:** This field contains the hash code of all other fields which is encrypted by the certified authority private key.

Applications of X.509 Authentication Service Certificate:

Many protocols depend on X.509 and it has many applications, some of them are given below:

- Document signing and Digital signature
- Web server security with the help of Transport Layer Security (TLS)/Secure Sockets Layer (SSL) certificates
- Email certificates
- Code signing
- Secure Shell Protocol (SSH) keys
- Digital Identities

5.B Explain in detail about Application Gateway firewall.

Ans.5.B)

An application gateway is a network device or service that provides application-layer functions, such as protocol translation, content filtering, and load balancing. It operates at the application layer (Layer 7) of the OSI model and can handle various protocols and applications.

Key functions of an application gateway may include:

1. Protocol Conversion:
   - Translating between different network protocols to enable communication between systems that use different protocols.
2. Load Balancing:
   - Distributing incoming network traffic across multiple servers to ensure no single server is overwhelmed, improving reliability and performance.
3. SSL Termination:
   - Handling the encryption and decryption of SSL/TLS traffic, offloading this process from backend servers.
4. Caching:
   - Storing frequently accessed data in a cache to reduce latency and improve response times.
5. Content Filtering:
   - Inspecting and filtering content based on predefined rules or policies.

Firewall in Computer Networks:

A firewall is a network security device or software that monitors and controls incoming and outgoing network traffic based on predetermined security rules. The primary goal of a firewall is to establish a barrier between a trusted internal network and untrusted external networks, such as the internet.

Key functions of a firewall may include:

1. Packet Filtering:
   - Examining packets of data and allowing or blocking them based on predefined rules. This is often done at the network layer (Layer 3) using rules based on source and destination IP addresses and ports.
2. Stateful Inspection:
   - Keeping track of the state of active connections and making decisions based on the context of the traffic. Stateful firewalls operate at both the network and transport layers (Layers 3 and 4).
3. Proxy Services:
   - Acting as an intermediary between clients and servers, forwarding requests and responses to enhance security and privacy.
4. Network Address Translation (NAT):
   - Modifying network address information in packet headers to allow multiple devices in a private network to share a single public IP address.

6.B Explain in detail about SQL injection?

Ans.6.B)

SQL injection is a type of cyberattack that occurs when an attacker is able to insert or manipulate malicious SQL code into a query, typically through user inputs in web applications or other software that interacts with a database. The goal of SQL injection attacks is to manipulate the SQL queries executed by the application's database, often leading to unauthorized access, data manipulation, or other malicious activities.

How SQL Injection Works:

1. User Input Vulnerability:
   - SQL injection usually occurs when a web application doesn't properly validate or sanitize user inputs before including them in SQL queries.
2. Malicious Input:
   - An attacker submits specially crafted input, often in the form of SQL code, into input fields or parameters expected by the application.
3. Injection Points:
   - The attacker aims to exploit injection points where user input is directly concatenated into SQL queries.
4. Manipulating SQL Queries:
   - The injected SQL code becomes part of the query executed by the database server, leading to unintended and potentially harmful consequences.

Types of SQL Injection:

1. Classic SQL Injection:
   - Occurs when attackers inject malicious SQL code into user input fields, such as login forms or search boxes.
2. Blind SQL Injection:
   - Attackers infer information from the database by injecting queries that result in true or false conditions. The application's response helps them deduce the structure or content of the database.
3. Time-Based Blind SQL Injection:
   - Similar to blind SQL injection, but the attacker induces the server to wait for a specified time before responding, revealing information based on the delay.
4. Union-Based SQL Injection:
   - Involves injecting a SQL UNION statement to combine the results of the original query with results from another query, allowing attackers to extract data.
5. Error-Based SQL Injection:
   - Exploits SQL errors generated by the application to extract information about the database structure or content.

Preventing SQL Injection:

1. Parameterized Queries:
   - Use parameterized queries or prepared statements that separate SQL code from user input.
2. Input Validation and Sanitization:
   - Validate and sanitize user inputs to ensure they conform to expected formats and don't include malicious code.
3. Least Privilege Principle:
   - Limit database user permissions to the minimum necessary for the application to function, reducing the potential impact of an SQL injection attack.
4. Use ORM (Object-Relational Mapping) Libraries:
   - ORM libraries often provide abstraction layers that help prevent direct SQL injection by handling SQL queries behind the scenes.
5. Stored Procedures:
   - Use stored procedures with parameterized inputs to encapsulate SQL logic and minimize the risk of injection.
6. Web Application Firewalls (WAF):
   - Implement a WAF to filter and monitor HTTP traffic, blocking known SQL injection patterns.
7. Regular Security Audits:
   - Regularly audit and test web applications for security vulnerabilities, including SQL injection, to identify and address potential risks.