**Assignment 1**

1. What is the Caesar cipher? Explain its working.

The Caesar cipher is one of the simplest and oldest encryption techniques. It works by shifting each letter in the plaintext by a fixed number of positions down the alphabet. For example, with a shift of 3, A becomes D, B becomes E, and so on. After Z, it wraps around back to A.

2. How does the Caesar cipher ensure encryption and decryption?

Encryption is done by shifting letters forward by the key value, while decryption is performed by shifting them backward by the same value. If the key is known, both operations are reversible.

3. What is the role of the shift key in the Caesar cipher?

The shift key (also called the offset or rotation) determines how many positions each letter is moved. It acts as the secret key needed for both encryption and decryption.

4. Can the Caesar cipher encrypt and decrypt numbers or symbols? Why or why not?

By default, the Caesar cipher only works on alphabetic characters. Numbers and symbols are not part of the standard Caesar cipher alphabet unless the algorithm is modified to include them.

5. How can you modify the Caesar cipher to handle case sensitivity?

To handle case sensitivity, you can treat uppercase and lowercase letters separately, preserving their case during encryption and decryption (e.g., A-Z and a-z are shifted within their own ranges).

6. What is the computational complexity of the Caesar cipher algorithm?

The computational complexity is O(n), where n is the length of the input text. Each character is processed once.

7. What are the limitations of the Caesar cipher in modern cryptography?

Very limited key space (only 25 possible shifts).

Easily breakable using brute force or frequency analysis.

Not suitable for protecting modern data or communication.

8. How would you decrypt a Caesar cipher message if the shift key is unknown?

You can perform a brute-force attack by trying all 25 possible shifts and checking which output makes sense (e.g., using a dictionary or human judgment).

9. What is the difference between symmetric and asymmetric encryption, and where does the Caesar cipher fit?

Symmetric encryption uses the same key for encryption and decryption.

Asymmetric encryption uses two keys: a public and a private key.

The Caesar cipher is a symmetric encryption technique.

10. Can the Caesar cipher be considered secure? Justify your answer.

No, the Caesar cipher is not secure by modern standards. It can be easily broken due to its small key space and predictable structure. It's mostly used for educational purposes.

**Assignment 2**

1. What is the Playfair cipher, and how does it differ from the Caesar cipher?

The Playfair cipher is a digraph substitution cipher that encrypts two letters at a time using a 5x5 key matrix. Unlike the Caesar cipher, which shifts single letters, Playfair considers letter pairs, making it more complex and harder to break using frequency analysis.

2. Explain the significance of the key matrix in the Playfair cipher.

The key matrix is a 5x5 grid containing the letters of the alphabet (usually omitting 'J') filled using a keyword. It determines how letter pairs are encrypted based on their position in the matrix.

3. How are repeated letters handled in the plaintext pairs?

If both letters in a digraph are the same (e.g., "LL"), a filler character (commonly 'X') is inserted between them to avoid repetition. For example, "BALLOON" becomes "BA LX LO ON".

4. Why are 'I' and 'J' treated as the same letter in the Playfair cipher?

Since the cipher uses a 5x5 matrix (25 spaces) but the alphabet has 26 letters, 'I' and 'J' are merged to fit the matrix. This is a standard convention to maintain matrix size.

5. What are the encryption rules when the two letters in a pair are in the same row or column?

Same Row: Replace each letter with the letter to its right, wrapping around to the start if needed.

Same Column: Replace each letter with the letter below it, wrapping around to the top if needed.

6. What happens if the letters in a pair form a rectangle in the key matrix?

If the letters are at the corners of a rectangle, replace each letter with the one in the same row but the opposite corner (i.e., swap columns).

7. How is a filler character used, and why is it necessary?

A filler character, usually 'X', is inserted to:

Separate repeated letters in a digraph.

Ensure that the plaintext has an even number of letters.

It maintains the digraph structure required for encryption.

8. What are the limitations of the Playfair cipher in modern cryptography?

Still vulnerable to frequency analysis.

Fixed matrix size limits flexibility.

Not suitable for encrypting large or binary data.

Less secure compared to modern encryption algorithms like AES.

9. Can the Playfair cipher encrypt and decrypt numeric data? Why or why not?

Not by default. The Playfair cipher is designed for alphabetic characters only. Numeric data must be converted to text or the algorithm must be modified to support numbers.

10. How would you adapt the Playfair cipher to handle special characters or different alphabets?

You can:

Expand the matrix to include additional characters (e.g., 6x6 grid for A–Z and 0–9).

Use custom encoding schemes (e.g., substitute special characters with placeholder letters).

Modify the algorithm to handle Unicode or other language characters.

**Assignment 3**

1. What is the Rail Fence Cipher, and how does it work?

The Rail Fence Cipher is a form of transposition cipher where the letters of the plaintext are written in a zigzag pattern across multiple lines (rails), then read row by row to form the ciphertext.

2. Explain the encryption and decryption process using the zigzag pattern.

Encryption: Write the message in a zigzag across the specified number of rails and read it row by row.

Decryption: Reconstruct the zigzag pattern using the ciphertext and read the characters diagonally to get the original message.

3. What are the main steps in encrypting and decrypting a message using the Rail Fence Cipher?

Encryption:

Choose number of rails.

Write the message in a zigzag (down and up) across the rails.

Read and combine the letters row by row.

Decryption:

Create a blank zigzag pattern with the given length and rails.

Fill in the ciphertext row by row.

Read the message in zigzag order.

4. How does the number of rails affect the security of the Rail Fence Cipher?

More rails slightly increase complexity, but the cipher still remains weak overall. Too many or too few rails can make the pattern more predictable or easy to guess.

5. What happens if the length of the plaintext is not a multiple of the number of rails?

It doesn't cause a problem. The zigzag pattern adjusts automatically, and some rails will simply have one fewer character.

6. Explain the concept of "zigzag" writing in the Rail Fence Cipher. How does it contribute to encryption?

The zigzag pattern rearranges the order of letters, hiding the original structure of the message. It's a visual transposition, not substitution, that contributes to confusion in the ciphertext.

7. Can the Rail Fence Cipher be used for longer or more complex messages? Why or why not?

Technically yes, but it's not recommended because it's not secure for long or sensitive messages. The pattern becomes easier to detect with longer texts.

8. How would you adapt the Rail Fence Cipher to handle spaces and punctuation marks in the plaintext?

You can:

Preserve spaces and punctuation as-is during encryption and decryption.

Or remove them temporarily and reinsert them after decryption (less common and not recommended).

9. What are the main weaknesses of the Rail Fence Cipher? How does it compare to modern cryptographic methods?

Easy to break with brute-force or pattern analysis.

Provides no substitution, just rearrangement.

No key or randomness involved.

Compared to modern cryptography (e.g., AES), it’s extremely insecure and unsuitable for any real application.

10. Can the Rail Fence Cipher be broken easily by an attacker? Justify your answer with reasoning.

Yes. Since the only unknown is the number of rails, an attacker can try all possibilities (brute-force) and quickly reconstruct the plaintext, especially for short messages.

**Assignment 4**

1. What is the Rail Fence Cipher, and how does it work?

The Rail Fence Cipher is a transposition cipher that encrypts messages by writing them in a zigzag pattern over multiple "rails" (rows) and reading the message line by line to produce the ciphertext.

2. Explain the encryption and decryption process using the zigzag pattern.

Encryption: Write the plaintext in a zigzag form down and up across a fixed number of rails. Read the rows line by line to form the ciphertext.

Decryption: Reconstruct the zigzag path and place the ciphertext characters row by row, then read them diagonally in the original zigzag pattern.

3. What are the main steps in encrypting and decrypting a message using the Rail Fence Cipher?

Encryption:

Choose the number of rails.

Write characters in a zigzag pattern.

Read rows line by line to get ciphertext.

Decryption:

Create a zigzag structure with placeholders.

Fill ciphertext row-wise.

Read in zigzag order to get the original text.

4. How does the number of rails affect the security of the Rail Fence Cipher?

The more rails used, the more scrambled the message becomes. However, too many rails can make the zigzag pattern predictable, and since the number of rails is a small range, brute-force is still easy.

5. What happens if the length of the plaintext is not a multiple of the number of rails?

It is not a problem. The zigzag pattern continues normally, and some rails may just have fewer characters than others.

6. Explain the concept of "zigzag" writing in the Rail Fence Cipher. How does it contribute to encryption?

The message is written diagonally down and up across rails, forming a zigzag. This rearranges the letters, disguising the structure of the original message through transposition, not substitution.

7. Can the Rail Fence Cipher be used for longer or more complex messages? Why or why not?

It can be used, but it’s not suitable for secure communication. Longer messages increase chances of pattern detection, and it's still easy to crack compared to modern encryption methods.

8. How would you adapt the Rail Fence Cipher to handle spaces and punctuation marks in the plaintext?

Either preserve spaces and punctuation during encryption/decryption, or

Remove them temporarily and reinsert them after decryption (not secure or reliable).

9. What are the main weaknesses of the Rail Fence Cipher? How does it compare to modern cryptographic methods?

Easy to break using brute-force or pattern analysis.

No actual encryption key; just reordering of characters.

No confusion or diffusion, unlike modern ciphers like AES, which use multiple rounds and complex keys for security.

10. Can the Rail Fence Cipher be broken easily by an attacker? Justify your answer with reasoning.

Yes. Since the only unknown is the number of rails, an attacker can try all possibilities (usually very few), making brute-force attack trivial. It's not secure for any serious application.

11. What is the computational complexity of the Rail Fence Cipher encryption and decryption processes?

Both encryption and decryption run in O(n) time, where n is the length of the plaintext or ciphertext.

It involves a single pass through the text to assign characters to rails (encryption) and another to read them in zigzag (decryption).

**Assignment 5**

1. What is the One Time Pad (OTP) cipher, and how does it differ from other symmetric ciphers like the Caesar Cipher or Vigenère Cipher?

The One Time Pad is a symmetric encryption technique that uses a truly random key that is as long as the message and used only once. Unlike Caesar or Vigenère ciphers, which use repeating patterns or fixed shifts, OTP provides perfect secrecy when implemented correctly.

2. Explain the concept and how OTP uses a random key to achieve unbreakable encryption.

OTP works by combining the plaintext with a random key using an XOR operation (or modular addition). The key must be:

Random

Equal in length to the message

Used only once

If all these conditions are met, the ciphertext reveals no information about the plaintext without the key.

3. Why is the key in the OTP cipher required to be truly random, and how does this impact the security of the encryption?

A truly random key ensures that there’s no predictable pattern an attacker can exploit. If the key is not random, it introduces structure into the ciphertext, which can be analyzed and broken.

4. Discuss the importance of key randomness and how any pattern in the key makes the cipher insecure.

Any repetition or pattern in the key acts like a weakness, just like in the Vigenère cipher, where repeated keys make the cipher vulnerable to frequency analysis. In OTP, randomness is crucial to ensure that every possible plaintext is equally likely.

5. How does the XOR operation work in the OTP encryption and decryption process?

Encryption: Ciphertext = Plaintext XOR Key

Decryption: Plaintext = Ciphertext XOR Key

XOR is reversible, meaning the same operation is used for both encryption and decryption.

6. Explain how XOR is used for both encryption and decryption in OTP.

XOR has a unique property:

A XOR B = C and C XOR B = A

So, encrypting a message with a key and then applying the same key again using XOR retrieves the original message.

7. What would happen if the key used in the OTP encryption is reused?

Reusing the key opens the cipher to cracking through known-plaintext or ciphertext comparison attacks, as attackers can extract relationships between two messages. This breaks the core principle of one-time use.

8. Discuss the security risks of using the same key for multiple messages and why it’s called a “one-time” pad.

Using the same key for multiple messages lets attackers perform a "crib dragging" attack to guess parts of both messages. It’s called a “one-time” pad because each key must be used only once to maintain perfect secrecy.

9. Can the OTP cipher be broken using brute-force attacks?

Technically, no. Even with brute-force, the OTP ciphertext could decrypt to any possible plaintext of that length if the key is unknown. There's no way to verify correctness without knowing the original key.

10. Why is the OTP cipher considered unbreakable when the key is truly random and only used once?

Because there's no pattern to analyze, and every possible plaintext is equally likely. Mathematically, it offers perfect secrecy, proven by Claude Shannon, meaning no amount of ciphertext can reveal the original message without the key.

**Assignment 6**

1. What is RSA encryption, and how does it differ from symmetric encryption algorithms like AES?

RSA is an asymmetric encryption algorithm that uses a pair of keys: one public and one private. AES is a symmetric algorithm, where the same key is used for both encryption and decryption.

RSA is mainly used for secure key exchange, while AES is used for fast, bulk data encryption.

2. Explain the differences between symmetric and asymmetric encryption.

Feature Symmetric Encryption Asymmetric Encryption

Keys Single key (same for both) Two keys (public & private)

Speed Fast Slower

Security Key sharing risk More secure key distribution

Examples AES, DES RSA, ECC

3. What is the significance of the public and private keys in the RSA algorithm?

The public key is used to encrypt messages and is shared with everyone.

The private key is used to decrypt and is kept secret.

Together, they ensure confidentiality and authenticity.

4. Discuss how public and private keys are used for encryption and decryption.

Encryption: ciphertext = (plaintext^e) mod n (using public key)

Decryption: plaintext = (ciphertext^d) mod n (using private key)

This ensures that only the holder of the private key can decrypt the message encrypted with the public key.

5. How do you generate the public and private keys in RSA?

Steps:

Choose two large prime numbers p and q

Compute n = p × q

Compute Euler’s Totient Function: ϕ(n) = (p - 1)(q - 1)

Choose e such that 1 < e < ϕ(n) and gcd(e, ϕ(n)) = 1

Calculate d such that d × e ≡ 1 mod ϕ(n)

Public key: (e, n)

Private key: (d, n)

6. Explain the process of selecting prime numbers, calculating the totient function, and choosing the exponents.

Select two large prime numbers p and q

Compute ϕ(n) = (p-1)(q-1)

Choose e such that e and ϕ(n) are coprime

Calculate d using modular inverse: d = e⁻¹ mod ϕ(n)

This ensures the keys work properly for encryption/decryption.

7. What is the role of Euler's Totient Function ϕ(n) in RSA key generation?

ϕ(n) determines the number of integers less than n that are coprime to n. It is crucial for:

Finding e (public exponent)

Calculating d (private exponent)

So it governs the mathematical relationship between the keys.

8. Describe how the totient function is calculated and its importance in the key generation process.

If n = p × q (where p and q are primes),

then ϕ(n) = (p - 1)(q - 1)

This value helps:

Ensure that the chosen e is coprime to ϕ(n)

Compute the modular inverse d used in decryption

9. Why must the public exponent e be coprime with ϕ(n)?

If e and ϕ(n) are not coprime, the modular inverse d cannot be computed, which is required to generate the private key.

Coprimality ensures a valid key pair for encryption and decryption.

10. Explain the significance of selecting a public exponent e that is coprime to ϕ(n).

Choosing e such that gcd(e, ϕ(n)) = 1 ensures that:

Encryption is possible

A unique decryption key d exists

Common values for e are small primes like 65537, which make encryption efficient while maintaining security.

**Assignment 7**

1. What is a Digital Signature, and why is it important?

A digital signature is a cryptographic technique used to validate the authenticity and integrity of a message, document, or software.

It is important because it ensures that:

The message is from a verified sender.

The content has not been altered.

The sender cannot deny sending the message (non-repudiation).

2. Explain the role of digital signatures in ensuring authenticity, integrity, and non-repudiation.

Authenticity: Verifies the sender’s identity using their public key.

Integrity: Confirms that the message has not been modified using hash verification.

Non-repudiation: Since only the sender has the private key, they cannot deny creating the signature.

3. How does the Digital Signature Algorithm (DSA) work?

DSA works in three steps:

Key Generation: Creates a public-private key pair.

Signing: Sender hashes the message and uses their private key to sign the hash.

Verification: Receiver uses the public key to verify the signature against the hash of the received message.

4. Discuss the key components of DSA, including key generation, signing, and verification.

Key Generation: Generate parameters (prime numbers and base), then create public and private keys.

Signing: Hash the message, generate a signature using a random number and the private key.

Verification: Use the public key to verify that the signature matches the hash of the message.

5. What is the significance of hashing in the digital signature process?

Hashing converts the message into a fixed-size digest.

Signatures are generated from the hash, not the entire message, to:

Improve speed and efficiency.

Detect even small changes in the message.

Ensure that signing works on messages of any length.

6. Explain how hashing ensures data integrity and reduces computational overhead.

Integrity: Any change in the original message changes the hash value, so tampering is detectable.

Efficiency: Signing a short hash instead of the full message is faster and less resource-intensive.

7. Why are private and public keys used in DSA, and how do they differ?

The private key is used to sign the message (kept secret).

The public key is used to verify the signature (shared publicly).

This asymmetry ensures that only the rightful owner can create a valid signature.

8. Describe the role of each key and the principle of asymmetric encryption.

Private Key: Signs the message → proves authorship.

Public Key: Verifies the signature → confirms origin.

Asymmetric encryption allows secure communication without sharing secret keys.

9. What happens if a message is altered after being signed?

If altered:

The hash of the altered message won't match the original hash.

Verification fails, indicating tampering or corruption.

10. Explain the verification process and how it detects tampering.

Verification steps:

Receiver hashes the received message.

Uses sender's public key to verify the signature.

If hashes match, the message is authentic and intact.

If not, tampering is detected immediately.

11. Why do digital signatures depend on the security of the private key?

The private key must remain secret because:

Anyone with it can forge signatures.

If leaked, signatures can be falsified, compromising authenticity and non-repudiation.

**Assignment 8**

1. What is the primary purpose of the Diffie–Hellman Key Exchange?

The primary purpose is to securely exchange a shared secret key between two parties over an insecure channel without transmitting the key directly.

2. Discuss how the protocol enables secure key exchange over an insecure channel.

Diffie–Hellman uses:

Public values (𝑝 and 𝑔) and private values (𝑎 and 𝑏).

Each party sends a computed public value.

Both compute the same shared secret independently using their private value and the other’s public value.

Even if intercepted, the private key cannot be derived easily due to the discrete logarithm problem.

3. What are the public and private components in the Diffie–Hellman protocol?

Public: Prime number 𝑝 and base 𝑔, and computed values (e.g., A = g^a mod p).

Private: Each party’s secret key (𝑎 or 𝑏), which is never shared.

4. Explain the roles of 𝑝, 𝑔, and the private keys.

𝑝 (prime number): Defines the finite cyclic group.

𝑔 (generator/primitive root): Used to generate values within the group.

Private keys (𝑎, 𝑏): Random secret values used to compute the shared secret securely.

5. Why is the Diffie–Hellman protocol considered secure?

Because even if an attacker intercepts all the public information, computing the private key or shared secret is practically impossible due to the computational hardness of the discrete logarithm problem.

6. Discuss the difficulty of solving the discrete logarithm problem.

The discrete logarithm problem is:

Given

𝐴

=

𝑔

𝑎

m

o

d

𝑝

A=g

a

modp, finding

𝑎

a is computationally infeasible when 𝑝 is large.

This mathematical difficulty is what underpins the security of the protocol.

7. What is a primitive root modulo 𝑝 and why is it important in Diffie–Hellman?

A primitive root modulo 𝑝 is a number 𝑔 such that its powers modulo 𝑝 produce all integers from 1 to 𝑝−1.

Importance:

Ensures that all possible public keys can be generated.

Provides a large key space, enhancing security.

8. Explain the significance of 𝑔 and its selection.

𝑔 is the generator of the cyclic group.

Must be chosen carefully so that it is a primitive root modulo 𝑝.

A poor choice of 𝑔 can weaken the security of the shared secret.

9. Can the shared secret be transmitted directly in Diffie–Hellman? Why or why not?

No, because:

Direct transmission would expose it to attackers.

The protocol is designed so that both parties independently compute the shared secret using exchanged public values and private keys.

10. What would happen if the private keys 𝑎 and 𝑏 are not kept secret?

If private keys are leaked:

An attacker can compute the shared secret.

The entire confidentiality of communication is compromised.

The protocol's security relies completely on the secrecy of private keys.

**Assignment 9**

1. What are the common types of SQL Injection attacks?

Common types include:

Classic SQL Injection

Blind SQL Injection

Union-based SQL Injection

Error-based SQL Injection

Time-based Blind SQL Injection

2. Discuss types like Classic SQL Injection, Blind SQL Injection, and Union-based SQL Injection.

Classic SQL Injection: Attacker directly manipulates queries by injecting malicious SQL via input fields.

Blind SQL Injection: No visible error message; attacker infers information based on application behavior.

Union-based SQL Injection: Uses the UNION operator to retrieve data from other tables in the database.

3. How does the example query SELECT \* FROM users WHERE username = 'admin' AND password = '' OR '1'='1'; exploit a SQL Injection vulnerability?

The injected ' OR '1'='1' always evaluates to true.

It bypasses authentication because the WHERE clause becomes:

sql

Copy

Edit

username = 'admin' AND (password = '' OR '1'='1')

Which is true even with an empty password.

4. Analyze the query and explain how it bypasses authentication.

The attacker enters:

Username: admin

Password: ' OR '1'='1

This causes the query to ignore password verification, allowing unauthorized access.

5. What is the role of user input sanitization in preventing SQL Injection?

Sanitization ensures that user inputs:

Are free from SQL control characters like ', --, ;, etc.

Cannot modify the intended structure of the query.

Are validated and cleaned before being used in SQL statements.

6. Discuss how sanitizing inputs helps avoid injecting malicious SQL.

It blocks dangerous characters or patterns.

Converts inputs to safe formats.

Ensures queries remain unchanged regardless of what the user inputs.

7. What are parameterized queries, and how do they prevent SQL Injection?

Parameterized queries (aka prepared statements) separate SQL logic from user input.

Inputs are treated as data, not code.

Example in Java:

java

Copy

Edit

PreparedStatement stmt = conn.prepareStatement("SELECT \* FROM users WHERE username = ? AND password = ?");

stmt.setString(1, username);

stmt.setString(2, password);

8. Explain the concept and provide examples of using prepared statements.

Prepared statements use placeholders (?) and bind user inputs safely.

Example in PHP (PDO):

php

Copy

Edit

$stmt = $pdo->prepare("SELECT \* FROM users WHERE email = ? AND password = ?");

$stmt->execute([$email, $password]);

9. Why is error handling important in preventing SQL Injection?

Poor error handling may reveal internal database structure or query errors.

This helps attackers refine their injection attempts.

Custom, generic error messages prevent information leakage.

10. Discuss how exposing detailed errors can help attackers craft more effective SQL Injection attacks.

Detailed errors reveal:

Table/column names

Query structure

Data types

Attackers use this to build precise payloads for exploitation.

11. What is the difference between authentication bypass and data exfiltration in SQL Injection?

Authentication Bypass: Used to log in without valid credentials.

Data Exfiltration: Used to extract sensitive data (e.g., user info, credit card numbers) from the database.