Assignment: Cryptography Analysis and Implementation

NAME : JAYASREE N
REGISTER NUMBER: 20MIS0370

Symmetric Key Algorithm: Advanced Encryption Standard (AES)

Brief Explanation:

AES is a symmetric key algorithm that uses a block cipher to encrypt and decrypt data. It operates on fixed-size blocks of data and employs a symmetric key for both encryption and decryption. AES supports key sizes of 128, 192, and 256 bits, with the latter being the most secure. The algorithm consists of several rounds of substitution, permutation, and mixing operations to provide strong encryption.

Key Strengths and Advantages:

Security: AES is widely considered secure and has been extensively analyzed by cryptographic experts. It has withstood rigorous testing and scrutiny, making it a trusted choice for protecting sensitive information.
Efficiency: AES is computationally efficient, making it suitable for a variety of applications, including embedded systems and resource-constrained environments.
Key Length Options: AES supports different key lengths, allowing users to choose the level of security that aligns with their requirements.
Standardization: AES is a widely adopted standard, ensuring interoperability and compatibility across various platforms and systems.
Vulnerabilities or Weaknesses:
Side-Channel Attacks: AES implementations may be vulnerable to side-channel attacks such as timing attacks and power analysis attacks if not properly protected.
Key Management: As AES is a symmetric key algorithm, the challenge lies in securely exchanging and managing the secret key between communicating parties.
Quantum Computing: While AES is resistant to classical computing attacks, its security may be compromised in the future by the development of practical quantum computers capable of breaking traditional symmetric key algorithms.

Real-World Examples:

AES is extensively used in numerous applications, including:
Secure communications: AES is utilized in protocols such as SSL/TLS to secure data transmissions over the internet.

File and disk encryption: Many operating systems and encryption tools employ AES for encrypting files and disk partitions.
Wireless networks: AES is commonly used in securing Wi-Fi networks through protocols like WPA2.

Asymmetric Key Algorithm: RSA (Rivest-Shamir-Adleman)

Brief Explanation:

RSA is an asymmetric key algorithm widely used for encryption and digital signatures. It relies on the mathematical properties of large prime numbers and modular arithmetic. RSA involves the generation of a public-private key pair, where the public key is used for encryption, and the private key is kept secret for decryption or signing.

Key Strengths and Advantages:

Security: RSA is based on the difficulty of factoring large integers into their prime factors. If implemented with sufficiently large key sizes, RSA can provide strong security.
Digital Signatures: RSA is well-suited for digital signatures, allowing for the verification of message integrity and authenticity.
Key Exchange: RSA can be used for secure key exchange, enabling two parties to establish a shared secret key without transmitting it directly.
Vulnerabilities or Weaknesses:
Key Length: The security of RSA depends on the length of the keys used. As computing power increases, longer key lengths are required to maintain security.
Performance: RSA encryption and decryption operations are computationally expensive compared to symmetric key algorithms, making it less suitable for bulk encryption.
Padding and Implementation: Proper padding schemes must be used to prevent vulnerabilities such as padding oracle attacks. Implementation flaws can also introduce vulnerabilities.

Real-World Examples:

RSA finds applications in various scenarios, including:
Secure email communication: RSA is commonly used for encrypting email messages and protecting the confidentiality of sensitive information.
SSL/TLS: RSA is employed during the SSL/TLS handshake process to establish secure communication channels for web browsing.
Digital signatures: RSA is utilized in digital signature algorithms like RSA-PSS to provide integrity and non-repudiation in digital documents.

Hash Function: Secure Hash Algorithm 256 (SHA-256)

Brief Explanation:

SHA-256 is a widely used hash function that produces a fixed-size 256-bit hash value (digest) from an input message of any size. It belongs to the SHA-2 family of hash functions and is based on the Merkle-Damgård construction.

Key Strengths and Advantages:

Data Integrity: SHA-256 provides a strong guarantee of data integrity. Even a slight change in the input data will produce a significantly different hash value, making it useful for verifying data integrity and detecting tampering.
Deterministic and Efficient: SHA-256 produces the same hash output for the same input, making it suitable for various applications such as password storage or file integrity checking. It is computationally efficient and can process large amounts of data quickly.
Cryptographic Security: SHA-256 has undergone extensive analysis and is widely trusted in the cryptographic community. It does not reveal any information about the input from its hash output (pre-image resistance) and is computationally infeasible to find two different inputs with the same hash (collision resistance).

Known Vulnerabilities or Weaknesses:

Pre-Image Resistance: While finding the original input from a given hash (pre-image) is computationally infeasible, as computational power increases, it becomes more susceptible to brute-force or collision-based attacks.
Length Extension Attacks: SHA-256 and other hash functions in the Merkle-Damgård construction are vulnerable to length extension attacks, where an attacker can extend an existing hash without knowing the original message.
Quantum Computing: The development of large-scale quantum computers could potentially break the collision resistance property of SHA-256 and other hash functions based on similar principles.

Real-World Examples:

SHA-256 is widely used in various applications, including:

Cryptocurrency: Bitcoin and many other cryptocurrencies use SHA-256 in their proof-of-work algorithm to ensure the security and immutability of the blockchain.
Digital Signatures: SHA-256 is often used in conjunction with asymmetric algorithms like RSA or ECDSA to generate digital signatures, providing integrity and authenticity to digital documents or transactions.

Password Storage: SHA-256 is employed in password hashing algorithms like bcrypt or PBKDF2 to securely store passwords by hashing them, protecting against unauthorized access to user credentials.

Implement AES encryption and decryption:

Step -1 : The pycryptodome library, which provides a Python interface to the AES algorithm.

```
pip install pycryptodome
```

Step -2: Import the required modules in your Python code

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
```

Step -3: Generate a random 256-bit key

```
key = get_random_bytes(32)
```

Step -4: Create an AES cipher object with the generated key and the desired mode (e.g., ECB or CBC)

```
cipher = AES.new(key, AES.MODE_ECB)
```

Step -5: Define a function to encrypt the plaintext message

```
def encrypt(plaintext):
    ciphertext = cipher.encrypt(pad(plaintext.encode(), AES.block_size))
    return ciphertext
```

Step -6: Define a function to decrypt the ciphertext message

```
def decrypt(ciphertext):
    decrypted_text = unpad(cipher.decrypt(ciphertext), AES.block_size)
    return decrypted_text.decode()
```

Step -7: Test the implementation by encrypting and decrypting a sample message

```
plaintext_message = "Hello, AES!"
ciphertext_message = encrypt(plaintext_message)
decrypted_message = decrypt(ciphertext_message)

print("Plaintext message:", plaintext_message)
print("Ciphertext message:", ciphertext_message)
print("Decrypted message:", decrypted_message)
```