Assessment-3

Title: Cryptography Analysis and Implementation

Submitted by:
GANTEGAMPU AKSHITHA
20BCB7065
Computer Science Engineering Spl. Business Systems
VIT-AP UNIVERSITY

Objective:

The goal of this project is to analyse cryptographic methods and implement them in a real-world environment.

Introduction:

Cryptographic algorithms are critical to the security and integrity of digital communications. We will look at three distinct types of cryptographic algorithms in this report: one symmetric encryption technique, one asymmetric encryption algorithm, and one hash function. Each algorithm's working principles, strengths, shortcomings, and real-world applications will be examined.

I. Advanced Encryption Standard (AES) Symmetric Encryption Algorithm

1. Explanation:The Advanced Encryption Standard (AES) is a symmetric encryption technique that is frequently used. It works with fixed-size data blocks, generally 128 bits, and employs a secret key for encryption and decryption. AES executes rounds, which are a series of substitution, permutation, and mixing operations.

The goal of this project is to examine cryptography systems.

2. Key Advantages and Strengths:

• Strong Security: AES has been carefully analysed by cryptographers all around the world and is deemed secure against known threats when properly implemented.

• Efficient: AES is very efficient and computationally quick, making it suited for systems with limited resources.

• Flexibility: AES allows key sizes of 128, 192, and 256 bits, allowing users to select the appropriate level of protection.

• Widespread Use: AES is the standard encryption algorithm used by governments, financial institutions, and organisations all around the world.

3. Known Weaknesses or Vulnerabilities:

• Side-Channel Attacks: AES is vulnerable to side-channel attacks, such as timing or power analysis assaults, in which an attacker utilises information that has been released during the encryption process.

• Key Management: AES is Any flaw in the key management method might lead to the compromise of encrypted data.

Examples from the Real World:

AES is widely used in a variety of applications, such as: • Secure communication protocols (e.g., SSL/TLS) for secure online browsing.

• Full disc encryption software (e.g., BitLocker, FileVault) to safeguard data stored on storage devices.

• Wireless networks (for example, Wi-Fi networks that use WPA2) to encrypt network communication.

Algorithm for Asymmetric Encryption: Rivest-Shamir-Adleman (RSA)

1. RSA is an asymmetric encryption method named after its creators. It employs a public-private key pair for encryption and decryption and is based on the mathematical principles of huge prime numbers. The public key is freely circulated, whereas the private key is kept confidential.

2. Key Advantages and Strengths:

• Strong Encryption: Based on the difficulty of factoring huge integers into prime factors, RSA provides safe encryption.

• Digital Signatures: RSReag the message integrity verification and sender authentication.

• Key Exchange: RSA allows two parties to exchange safe keys, providing secure communication routes.

• extensively Used: RSA is extensively supported and used in a variety of cryptographic libraries and systems.

3. Known Weaknesses or Vulnerabilities:

• Key Size: The key size employed by RSA is directly connected to its security. Longer key sizes are necessary to ensure security as computing power increases.

• time Attacks: Implementation flaws, such as discrepancies in time during encryption or decryption, can be exploited to retrieve the private key.

4. Real-World Applications: RSA is frequently used in the following scenarios:

• Use protocols such as OpenPGP or S/MIME to secure email transmission.

• Document signing or certificate authority that use digital signatures.

III. Hash Function: SHA-256 (256-bit Secure Hash Algorithm)

1. Explanation: SHA-256 is a cryptographic hash algorithm that takes any size input message and returns a 256-bit hash value. It computes the hash value via a sequence of logical operations, including bitwise operations and modular arithmetic. The resultant hash is unique to the input data, and even little changes in the input result in a considerably different hash.

2. Key Advantages and Strengths:

• Collision Resistance: SHA-256 is meant to produce a minimal likelihood of the same hash value for diverse inputs, resulting in good collision resistance.

• Deterministic Output: SHA-256 will always create the same hash output given the same input, allowing for verification and integrity tests.

• SHA-256 has a fixed size.

• Fixed Size: SHA-256 produces a fixed-size hash result regardless of input size, making it ideal for a wide range of applications.

• Wide Adoption: Because SHA-256 is extensively used and supported, it has become a mainstream choice for cryptographic hash algorithms.

3. Known Weaknesses or Vulnerabilities:

• Length Extension Attacks: SHA-256 is susceptible to length extension attacks, in which an attacker increases the hash value without knowing the original material. HMAC constructs or specialised variations such as SHA-3 can be used to mitigate this problem.

• Quantum Computing: In the future, quantum computers may be able to compromise the security of SHA-256 and other regularly used cryptographic algorithms. As prospective alternatives, post-quantum hash functions are being investigated.

4. Examples from the Real World:

The SHA-256 algorithm is often used in the following applications:

• Fixed Size: SHA-256 produces a fixed-size hash result regardless of input size, making it ideal for a wide range of applications.

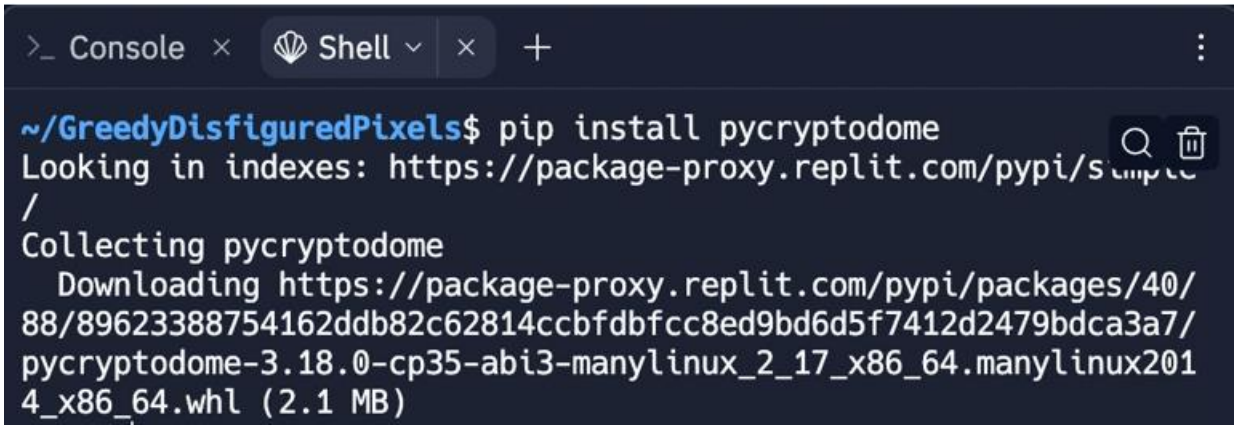
• used in conjunction with asymmetric encryption methods to generate digital signatures that provide digital document integrity and authenticity.

• Password Storage: SHA-256 is used to securely store user passwords in databases, along with other methods such as salting and stretching.

• Blockchain Technology: SHA-256 is widely used in blockchain networks such as Bitcoin to secure transaction data integrity and block validation.

Implementation: Python Implementation and Security Analysis of the AES Encryption Algorithm:

Step 1: Create the Environment



Step 2: Create the AES Key
Step 3: Using the AES technique, encrypt the contents from a file.
Step 4: Decrypt the encrypted data using the same key that was used for encryption.
Step 5: Encrypt a file and then decode the encrypted contents to ensure that our implementation is correct.

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

# Generate a random 256-bit key
key = get_random_bytes(32)

# Generate a random 128-bit IV
iv = get_random_bytes(16)

def encrypt_file(file_path, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    encrypted_data = b''

    with open(file_path, 'rb') as file:
        while True:
            chunk = file.read(16)  # Read 16 bytes at a time
            if len(chunk) == 0:
                break
            elif len(chunk) % 16 != 0:
                # Pad the chunk if its length is not a multiple of 16
                chunk += b' ' * (16 - (len(chunk) % 16))

            encrypted_chunk = cipher.encrypt(chunk)
            encrypted_data += encrypted_chunk

    return encrypted_data
```

```
def decrypt_data(encrypted_data, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = cipher.decrypt(encrypted_data)

    return decrypted_data

file_path = 'file_path.txt'

# Encrypt the file
encrypted_data = encrypt_file(file_path, key, iv)

# Decrypt the encrypted data
decrypted_data = decrypt_data(encrypted_data, key, iv)

# Print the decrypted data
print(decrypted_data.decode())
```

Threats or vulnerabilities to consider:
1. Key Management: It is critical to safeguard the secret key. To prevent unauthorised access, keep the key safe and use strict access restrictions.
2. Side-Channel Attacks: AES is susceptible to sidechannel attacks like as timing and power analysis. Countermeasures such as constant-time implementations can help to reduce these concerns.
3. Weak IVs: Using weak or predictable IVs might jeopardise AES security. It is important to generate random and distinct IVs for each encryption process.
Best Practises and Countermeasures:
1. Key Protection: Securely store the secret key, for example, with hardware security modules (HSMs) or protected key vaults. Use stringent access controls to limit key access to authorised individuals only.
2. IV Management: For each encryption operation, generate random and unique IVs. Never use the same IV with the same key more than once.
3. Implement Countermeasures: Use countermeasures against side-channel attacks, such as constant-time implementations or hardware with built-in timing attack protection.
Trade-offs and limitations:
1. Performance: AES encryption and decryption can be computationally costly, particularly for big files or significant data volumes, creating overhead and having an impact on resource-constrained applications.
2. Key Management: Encryption keys must be securely managed and protected, which necessitates careful design and execution of key distribution, storage, and revocation protocols.

3. Compatibility: Differences in AES implementations and support for varying key sizes across systems and platforms can make assuring compatibility difficult.

Finally, cryptography is critical for securing digital connections and safeguarding sensitive

= AES is a popular symmetric encryption technique that offers high security, efficiency.