St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2025-2026
Class: BE-IT A/B, Semester: VII

Subject: Secure Application Development Lab

Student Name: Charis Fernandes Student Roll No: 26

Experiment – 9 : Study and Implement Symmetric and Asymmetric Encryption Algorithm

Aim: To study and Implement symmetric and asymmetric encryption algorithm
a. Symmetric Key Cryptography (Data Encryption Standard)
b. Asymmetric Key Cryptography (Diffie Hellman)

Objectives: After study of this experiment, the student will be able to

- understand what is cryptography
- understand the concept of symmetric and asymmetric cryptography
- understand where encryption/decryption standards are used

Lab objective mapped: ITL703.6 To **apply** secure coding for cryptography

Prerequisite: Basic knowledge of symmetric and asymmetric key cryptography.

Requirements: C/C++/JAVA/PYTHON

Pre-Experiment Theory:

(a) **Symmetric Key Cryptography**

The Data Encryption Standard (DES) is a symmetric key block cipher algorithm used for data encryption and decryption. It was widely used in the past but is now considered outdated and insecure due to its relatively short key length (56 bits). Nonetheless, understanding how DES works can provide insight into the historical development of encryption algorithms.

Key components of the DES algorithm:

Key Generation:

- The original DES uses a 56-bit key, but only 64 bits are used for the key schedule, with the remaining 8 bits being used for error checking and parity.
- The 56-bit key undergoes a permutation and shifting process to produce 16 subkeys, each of 48 bits in length.

Encryption Process:

- DES operates on 64-bit blocks of plaintext and ciphertext.
- The initial step involves permuting the 64-bit plaintext using a fixed permutation table (initial permutation, IP).
- The 64-bit block is then divided into two halves, each 32 bits in size: the left half (L0) and the right half (R0).

- The core of the DES algorithm consists of 16 rounds, where each round uses the corresponding subkey generated during the key schedule.
- In each round, the right half of the data from the previous round (Ri-1) is expanded to 48 bits using an expansion permutation (E-bit selection).
- The expanded right half is then XORed with the current subkey (Ki) to introduce key mixing.
- The XOR result is passed through a series of eight substitution boxes (S-boxes) to perform nonlinear transformations.
- After S-box substitution, a permutation (P-box) is applied to the result.
- The output is then XORed with the left half of the data from the previous round (Li-1).
- The left and right halves are swapped, and the next round begins.

Decryption Process:

- The decryption process in DES is the same as encryption, but the subkeys are used in reverse order.

Final Permutation:

- After completing all 16 rounds, the two halves are combined and undergo a final permutation (inverse initial permutation, IP^-1) to produce the final 64-bit ciphertext.

(b) **Asymmetric Key Cryptography**

Diffie Hellman key exchange is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

Algorithm:

Alice and Bob, two users who wish to establish secure communications. We can assume that Alice and Bob know nothing about each other but are in contact.

1. Communicating in the clear, Alice and Bob agree on two large positive integers, p and g, where p is a prime number and g is a primitive root mod p.
2. Alice randomly chooses another large positive integer, XA, which is smaller than p. XA will serve as Alice's private key.
3. Bob similarly chooses his own private key, XB.
4. Alice computes her public key, YA, using the formula YA = (g^XA) mod p.
5. Bob similarly computes his public key, YB, using the formula YB = (g^XB) mod p.
6. Alice and Bob exchange public keys over the insecure circuit.
7. Alice computes the shared secret key, k, using the formula k = (YB ^XA) mod p.
8. Bob computes the same shared secret key, k, using the formula k = (YA ^XB) mod p.
9. Alice and Bob communicate using the symmetric algorithm of their choice and the shared secret key, k, which was never transmitted over the insecure circuit.

Procedure:

Insert following codes with output

a. Symmetric Key Cryptography (Data Encryption Standard)

```
def xor_encrypt_decrypt(key, data):
 return ''.join(chr(ord(c) ^ key) for c in data)
def generate_key(length):
 return (ord('K') % 256)
def main():
 key = generate_key(256)
 message = input("Enter a statement: ")
 encrypted_message = xor_encrypt_decrypt(key, message)
 print("Encrypted message:", ''.join(f"{ord(c):02x}" for c in encrypted_message))
 decrypted_message = xor_encrypt_decrypt(key, encrypted_message)
 print("Decrypted message:", decrypted_message)
if __name__ == "__main__":
 main()
```

```
Enter a statement: ello mate charis
Encrypted message: 2e2727246b262a3f2e6b28232a392238
Decrypted message: ello mate charis
```

Demonstrates a simple XOR encryption and decryption using a fixed key derived from the character 'K'. It takes user input, encrypts it by XORing each character with the key, prints the encrypted message as hexadecimal, and then decrypts it using the same key.

b. Asymmetric Key Cryptography (Diffie Hellman)

```
def power(a, b, p):
    if b == 1:
        return a
    else:
        return pow(a, b) % p
def xor_encrypt_decrypt(key, data):
 return ''.join(chr(ord(c) ^ key) for c in data)
def generate_key(shared_secret):
 return shared_secret % 256
def main():
 P = int(input("Enter the prime number P: "))
 G = int(input("Enter the primitive root G for P: "))
 a = int(input("Enter Sender's private key a: "))
 x = power(G, a, P)
 b = int(input("Enter Receiver's private key b: "))
 y = power(G, b, P)
 ka = power(y, a, P)
 kb = power(x, b, P)
 key = generate_key(ka)
 message = input("Enter a message to encrypt: ")
 encrypted_message = xor_encrypt_decrypt(key, message)
 print("Encrypted message:", encrypted_message)
 decrypted_message = xor_encrypt_decrypt(key, encrypted_message)
 print("Decrypted message:", decrypted_message)
if __name__ == "__main__":
 main()
```

```
Enter the prime number P: 9
Enter the primitive root G for P: 8
Enter Sender's private key a: 5
Enter Receiver's private key b: 6
Enter a message to encrypt: townhall
Encrypted message: unvoi`mm
Decrypted message: townhall
```

Shows a Diffie-Hellman key exchange to create a shared secret, which is then used as the key for XOR encryption/decryption.

Post-Experiments Exercise:

Extended Theory: Nil

Post Experimental Exercise:

Questions:

- Compare symmetric and asymmetric encryption algorithms.
- Discuss the security of DES in the modern context, considering the advancements in cryptanalysis techniques and computing power.
- Discuss real-world applications of the Diffie-Hellman algorithm in various cryptographic protocols, such as secure key exchange for SSL/TLS in web browsers or VPN connections.

Conclusion:

- Write what was performed in the experiment.
- Write the significance of the topic studied in the experiment.

References:

http://cse29-iiith.vlabs.ac.in/
