Final Year B.Tech. (CSE) – VII [2024-25]

6CS451: Cryptography and Network Security Lab (C&NS Lab)

Date: 26/08/2024

Assignment 5

PRN: 21510042 **Name:** Omkar Rajesh Auti

---

1. **Apply DES algorithm for practical applications**

Ans:
The Data Encryption Standard (DES) is a symmetric-key algorithm for the encryption of digital data. Although DES is now considered insecure for many applications due to its small key size, it is still an important algorithm for understanding the basics of cryptography.

Practical Application of DES Algorithm

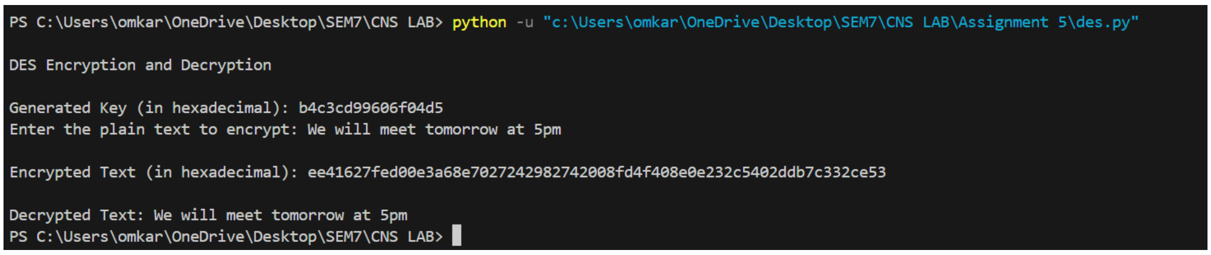
To apply the DES algorithm in a practical application, we can use the **pycryptodome** library in Python, which provides an implementation of DES. Below is an example that demonstrates how to use DES to encrypt and decrypt a message.

Python Code:

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


def des_encrypt(plain_text, key):
    """
    Encrypt the plain text using DES algorithm.

    Parameters:
    plain_text (str): The text to be encrypted.
    key (bytes): The encryption key (must be 8 bytes long).

    Returns:
    bytes: The encrypted cipher text.
    """
    cipher = DES.new(key, DES.MODE_ECB)
```

```python
    padded_text = pad(plain_text.encode(), DES.block_size)
    encrypted_text = cipher.encrypt(padded_text)
    return encrypted_text


def des_decrypt(cipher_text, key):
    """
    Decrypt the cipher text using DES algorithm.

    Parameters:
    cipher_text (bytes): The encrypted text to be decrypted.
    key (bytes): The decryption key (must be 8 bytes long).

    Returns:
    str: The decrypted plain text.
    """
    cipher = DES.new(key, DES.MODE_ECB)
    decrypted_text = unpad(cipher.decrypt(cipher_text), DES.block_size)
    return decrypted_text.decode()


def main():
    """
    The main function to run the program.
    """
    print("\nDES Encryption and Decryption")

    # Generate a random 8-byte key for DES
    key = get_random_bytes(8)
    print(f"\nGenerated Key (in hexadecimal): {key.hex()}")

    # Input plaintext
    plain_text = input("Enter the plain text to encrypt: ")

    # Encrypt the plaintext
    encrypted_text = des_encrypt(plain_text, key)
    print(f"\nEncrypted Text (in hexadecimal): {encrypted_text.hex()}")

    # Decrypt the ciphertext
    decrypted_text = des_decrypt(encrypted_text, key)
    print(f"\nDecrypted Text: {decrypted_text}")


if __name__ == "__main__":
    main()
```

Output:



```
PS C:\Users\omkar\OneDrive\Desktop\SEM7\CNS LAB> python -u "c:\Users\omkar\OneDrive\Desktop\SEM7\CNS LAB\Assignment 5\des.py"

DES Encryption and Decryption

Generated Key (in hexadecimal): b4c3cd99606f04d5
Enter the plain text to encrypt: We will meet tomorrow at 5pm

Encrypted Text (in hexadecimal): ee41627fed00e3a68e7027242982742008fd4f408e0e232c5402ddb7c332ce53

Decrypted Text: We will meet tomorrow at 5pm
PS C:\Users\omkar\OneDrive\Desktop\SEM7\CNS LAB>
```

Practical Applications:

- **File Encryption:** DES can be used to encrypt sensitive files before storing them in insecure locations.
- **Secure Communication:** DES ensures that messages sent over a network are unreadable to unauthorized parties.
- **Password Storage:** Encrypting passwords before storing them in databases (though modern standards recommend stronger algorithms like AES).

While DES itself is outdated and not recommended for secure applications, understanding how it works is crucial for grasping more advanced encryption algorithms like AES.