In [1]:
```
##import Libraries
```

In [2]:
```
import sqlite3
from cryptography.fernet import Fernet

# Function to generate an encryption key from a password
def generate_key_from_password(password):
    key = Fernet.generate_key()
    cipher_suite = Fernet(key)
    encrypted_password = cipher_suite.encrypt(password.encode()).decode()
    return key, encrypted_password

# Function to decrypt a password using the encryption key
def decrypt_password(key, encrypted_password):
    cipher_suite = Fernet(key)
    decrypted_password = cipher_suite.decrypt(encrypted_password.encode()).decode()
    return decrypted_password

# Initialize the database and encryption key
conn = sqlite3.connect("patient_records.db")
cursor = conn.cursor()

# Check if the "config" table exists, otherwise create it
cursor.execute('''
    CREATE TABLE IF NOT EXISTS config (
        id INTEGER PRIMARY KEY,
        encryption_key TEXT,
        encrypted_password TEXT
    )
''')
conn.commit()

# Check if the encryption key exists in the "config" table, otherwise create one
cursor.execute("SELECT encryption_key FROM config")
key_row = cursor.fetchone()

if key_row:
    encryption_key = key_row[0]
else:
    password = input("Create a password for data encryption: ")
    encryption_key, encrypted_password = generate_key_from_password(password)
    cursor.execute("INSERT INTO config (encryption_key, encrypted_password) VALUES (?, ?)", (encryption_key, en
    conn.commit()

cipher_suite = Fernet(encryption_key)
```

```
Create a password for data encryption: tolab
```

In [3]:
```
##Generate an encryption key and create a cipher suite for encryption and decryption:
encryption_key = Fernet.generate_key()
cipher_suite = Fernet(encryption_key)
```

In [4]:
```
##Connect to an SQLite database and create a table to store patient records:
conn = sqlite3.connect("patient_records.db")
cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS patients (
        id INTEGER PRIMARY KEY,
        name TEXT,
        diagnosis TEXT,
        medications TEXT
    )
''')
conn.commit()
```

In [5]:
```
##Create a function to encrypt and insert patient data into the database:

def encrypt_and_insert_patient_data(name, diagnosis, medications):
    data_to_encrypt = {
        "name": name,
        "diagnosis": diagnosis,
        "medications": medications
    }
    encrypted_data = cipher_suite.encrypt(str(data_to_encrypt).encode()).decode()

    cursor.execute('''
        INSERT INTO patients (name, diagnosis, medications)
        VALUES (?, ?, ?)
    ''', (name, diagnosis, encrypted_data))
    conn.commit()
```

In [6]:
```
##Create a function to search for patient records and decrypt them based on a search term:
def search_and_decrypt_patient_data(search_term):
    cursor.execute('SELECT id, name FROM patients WHERE name LIKE ?', ('%' + search_term + '%',))
```

```
    matching_records = cursor.fetchall()

    for record in matching_records:
        patient_id, patient_name = record
        cursor.execute('SELECT medications FROM patients WHERE id = ?', (patient_id,))
        encrypted_data = cursor.fetchone()[0]

        decrypted_data = cipher_suite.decrypt(encrypted_data.encode()).decode()

        print(f"Patient ID: {patient_id}, Patient Name: {patient_name}")
        print("Decrypted Patient Data:")
        print(decrypted_data)
        print("\n")
```

In [8]:
```
##Create a loop to accept user input for adding patient records and searching for records:
while True:
    print("Options:")
    print("1. Add Patient Record")
    print("2. Search and Decrypt Patient Records")
    print("3. Exit")

    choice = input("Select an option: ")

    if choice == "1":
        name = input("Enter patient name: ")
        diagnosis = input("Enter diagnosis: ")
        medications = input("Enter medications (comma-separated): ").split(",")

        encrypt_and_insert_patient_data(name, diagnosis, medications)
        print("Patient record added.\n")
    elif choice == "2":
        search_term = input("Enter search term: ")
        search_and_decrypt_patient_data(search_term)
    elif choice == "3":
        print("Exiting...")
        break
    else:
        print("Invalid choice.\n")
```

```
Options:
1. Add Patient Record
2. Search and Decrypt Patient Records
3. Exit
Select an option: 3
Exiting...
```

In [ ]:
```
##Close connection
conn.close()
```

In [ ]:
```

```