CRYPTOGRAPHY AND NETWORK SECURITY
LAB 10 : MESSAGE DIGEST 5 ALGORITHM

NAME : OM SUBRATO DEY

REGISTER NO.: 21BAI1876

Belonging to the family of hash functions, MD5 generates a 128 bit hash value. Here are some of its main advantages and disadvantages:

Advantages:

The hash value is produced by MD5 almost right away.

- It can be applied in a program effortlessly.

Small fingerprint - Another advantage of the algorithm, a 128-bit MD5 hash is easy to work with.

In order to produce the hash using MD5, no secret encryption key is necessary.

Disadvantages:

Although MD5 is a common choice, certainty has emerged about its collision vulnerability, demonstrating that it is possible for researchers to intentionally produce two different messages that yield the same hash.

That means the attacker can get inputs that hash to a given MD5 hash through a preimage attack … They are prone to preimage attacks since the construction of the hash is only 128 bits.

It is not capable enough - MD5 is no longer considered cryptographically sound hashing algorithm. MD5 ought to be discarded; on the other hand, hash functions derived from the SHA-256 family are favored.

- At risk of birthday attacks - The preceding hash size is insufficient when compared to others and, therefore, is susceptible to birthday attacks to discover collisions

Briefly, it is safe to say that, despite MD5 being computationally quick and simple to implement, it doesn't provide sufficient collision resistance and is unsatisfactory for present applications. It is primarily used today for data check summing but not for encryption or authentication..

CODE:

```
import hashlib
import os


# Function to generate an MD5 hash of a string
def md5_encrypt(data: str) -> str:
    """Generate MD5 hash of the given string."""
    md5_hasher = hashlib.md5()
    md5_hasher.update(data.encode('utf-8'))
    return md5_hasher.hexdigest()


# Simulate user registration (storing password as MD5 hash)
def register_user(username: str, password: str, users_db: dict) ->
None:
    """Registers a user by storing their username and hashed
password."""
    if username in users_db:
        print(f"User '{username}' already exists!")
        return


    # Hash the password
    hashed_password = md5_encrypt(password)


    # Simulate storing in the database
    users_db[username] = hashed_password
    print(f"User '{username}' registered successfully with hashed
password: {hashed_password}")


# Simulate user login (verifying password)
def login_user(username: str, password: str, users_db: dict) -> None:
    """Verifies a user's login by comparing the hashed password."""
    if username not in users_db:
```

```
        print(f"User '{username}' does not exist!")
        return

    # Hash the entered password to compare
    entered_password_hash = md5_encrypt(password)

    # Check if the hash matches the stored hash
    if entered_password_hash == users_db[username]:
        print(f"Login successful for user '{username}'!")
    else:
        print("Incorrect password.")

# Simulate the "database" to store user data
users_db = {}

# Registering users
print("Registering users...")
register_user("alice", "password123", users_db)
register_user("bob", "mySecretPass", users_db)
print()

# Attempting logins
print("Attempting logins...")
login_user("alice", "password123", users_db)  # Correct password
login_user("bob", "wrongPass", users_db)      # Incorrect password
login_user("charlie", "randomPass", users_db) # User doesn't exist

print("\nCurrent user database (hashed passwords):")
print(users_db)
```

OUTPUT:

21BAI1876 LAB 10.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
import hashlib
import os

# Function to generate an MD5 hash of a string
def md5_encrypt(data: str) -> str:
    """Generate MD5 hash of the given string."""
    md5_hasher = hashlib.md5()
    md5_hasher.update(data.encode('utf-8'))
    return md5_hasher.hexdigest()

# Simulate user registration (storing password as MD5 hash)
def register_user(username: str, password: str, users_db: dict) -> None:
    """Registers a user by storing their username and hashed password."""
    if username in users_db:
        print(f"User '{username}' already exists!")
        return

    # Hash the password
    hashed_password = md5_encrypt(password)

    # Simulate storing in the database
    users_db[username] = hashed_password
    print(f"User '{username}' registered successfully with hashed password: {hashed_password}")

# Simulate user login (verifying password)
def login_user(username: str, password: str, users_db: dict) -> None:
    """Verifies a user's login by comparing the hashed password."""
    if username not in users_db:
        print(f"User '{username}' does not exist!")
        return

    # Hash the entered password to compare
    entered_password_hash = md5_encrypt(password)

    # Check if the hash matches the stored hash
    if entered_password_hash == users_db[username]:
        print(f"Login successful for user '{username}'!")
    else:
        print("Incorrect password.")

# Simulate the "database" to store user data
users_db = {}

# Registering users
print("Registering users...")
register_user("alice", "password123", users_db)
register_user("bob", "mySecretPass", users_db)
print()

# Attempting logins
print("Attempting logins...")
login_user("alice", "password123", users_db)  # Correct password
login_user("bob", "wrongPass", users_db)      # Incorrect password
```

0s completed at 11:20 AM