



**MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE**

**NATIONAL TECHNICAL UNIVERSITY OF UKRAINE**

**" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"**

**Artem Volokyta**

**Lab Work 3**

**Implementation and Performance Analysis of Symmetric Encryption  
and Hashing Algorithms**

**kulubecioglu Mehmet**

**Variant Number 10**

**IM-14 FIOT**

**Kyiv**

**IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE**

**2025**

# Introduction

This report explores the implementation and performance analysis of symmetric encryption algorithms (DES and AES) and a cryptographic hash function (SHA-3). The goal of this project is to understand encryption and decryption processes, analyze security properties, and compare computational efficiency.

## Project Objectives:

1. Encrypt and decrypt data using the DES (Data Encryption Standard) algorithm.
2. Encrypt and decrypt data using the AES (Advanced Encryption Standard) algorithm.
3. Generate and analyze cryptographic hashes using SHA-3.
4. Compare the performance of DES, AES, and SHA-3 in terms of execution time.

## Technologies Used

The following technologies were utilized in this project:

- **Python 3:** Programming language for simulation execution
- **PyCryptodome:** Library for cryptographic operations
- **Google Colab:** Environment for running and testing the code

## Implementation Details

### DES Encryption and Decryption

The Data Encryption Standard (DES) is a symmetric-key algorithm that encrypts data in 64-bit blocks using a 56-bit key.

## DES Encryption Code:

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

# 8-byte key required for DES
key = b'b1234567'

cipher = DES.new(key, DES.MODE_CBC)

plaintext = b'This is a test'
padded_text = pad(plaintext, DES.block_size)

ciphertext = cipher.encrypt(padded_text)

print("Plaintext:", plaintext)
print("Padded Text:", padded_text)
print("Ciphertext:", ciphertext)
print("IV:", cipher.iv)

Plaintext: b'This is a test'
Padded Text: b'This is a test\x02\x02'
Ciphertext: b'g-Q\xc9IR7/\xb5(\xb2\x92\xc5\xaf\xd4$'
IV: b'\xa2\xfe\xdf\xc\xbd4\xa2'
```

## DES Decryption Code:

```
IV: b'\xa2\xfe\xdf\xc\xbd4\xa2'

[3] # Çözücü oluştur
dcipher = DES.new(key, DES.MODE_CBC, iv=cipher.iv)

# Şifreli metni çöz
decrypted_msg = unpad(dcipher.decrypt(ciphertext), DES.block_size)

print("Decrypted Message:", decrypted_msg)

Decrypted Message: b'This is a test'
```

## 4.2. SHA-3 Hashing

SHA-3 is a cryptographic hash function that generates a fixed-length digest, making it useful for data integrity verification.

### SHA-3 Hashing Code:

```
from Crypto.Hash import SHA3_256

# Hashleme nesnesi oluştur
hasher = SHA3_256.new()

# Hashlenecek veri
data = b'This is a test message'
hasher.update(data)

# Hash değerini al
hash_value = hasher.hexdigest()

print("Data:", data)
print("SHA3-256 Hash:", hash_value)

Data: b'This is a test message'
SHA3-256 Hash: dfce1be7affff1a4cd637356233d031b46aedf43176a734chdfha4f8c178921e
```

### 4.3. AES Encryption and Decryption

AES (Advanced Encryption Standard) is a block cipher that replaces DES and is considered more secure. It supports 128-bit, 192-bit, or 256-bit keys.

#### AES Encryption Code:

```
from Crypto.Cipher import AES

# 16 baytlık anahtar
key = b'1234567812345678'

# Şifrelenecek veri
data = b'This is a test message'

# AES şifreleyici (EAX modu)
cipher = AES.new(key, AES.MODE_EAX)

# Şifreleme işlemi
nonce = cipher.nonce
ciphertext, tag = cipher.encrypt_and_digest(data)

print("Ciphertext:", ciphertext)
print("Nonce:", nonce)
print("Tag:", tag)
```

Ciphertext: b'\xf0\x86\xfe\xad\x0c\x8\xff8<\x8c\x94\xe6 K'  
Nonce: b'\x17\x08\x84,\x3\x7f\x00\\\xf4\x940\xcd\x06\x92'  
Tag: b'n\x82\xcewKj\x13\xad\x81W\x1a{\x9dLD\xb3'

#### AES Decryption Code:

```
dcipher = AES.new(key, AES.MODE_EAX, nonce=nonce)

decrypted_msg = dcipher.decrypt(ciphertext)

print("Decrypted Message:", decrypted_msg)
```

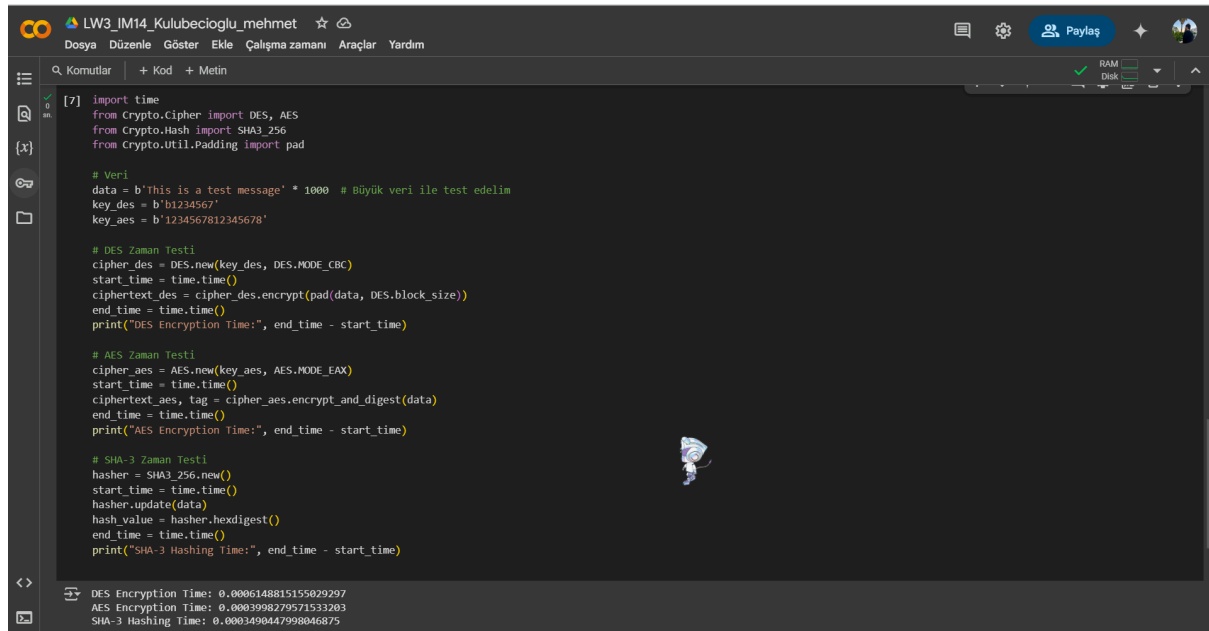
Decrypted Message: b'This is a test message'

AES uses a "nonce" to ensure security and uniqueness of encrypted messages.

## 5. Performance Analysis

A performance comparison was conducted to analyze the execution time of DES, AES, and SHA-3.

### Performance Testing Code:



```
[7] import time
from Crypto.Cipher import DES, AES
from Crypto.Hash import SHA3_256
from Crypto.Util.Padding import pad

# Veri
data = b'This is a test message' * 1000 # Büyük veri ile test edelim
key_des = b'b1234567'
key_aes = b'1234567812345678'

# DES Zaman Testi
cipher_des = DES.new(key_des, DES.MODE_CBC)
start_time = time.time()
ciphertext_des = cipher_des.encrypt(pad(data, DES.block_size))
end_time = time.time()
print("DES Encryption Time:", end_time - start_time)

# AES Zaman Testi
cipher_aes = AES.new(key_aes, AES.MODE_EAX)
start_time = time.time()
ciphertext_aes, tag = cipher_aes.encrypt_and_digest(data)
end_time = time.time()
print("AES Encryption Time:", end_time - start_time)

# SHA-3 Zaman Testi
hasher = SHA3_256.new()
start_time = time.time()
hasher.update(data)
hash_value = hasher.hexdigest()
end_time = time.time()
print("SHA-3 Hashing Time:", end_time - start_time)
```

DES Encryption Time: 0.0006148815155029297  
AES Encryption Time: 0.0003998279571533203  
SHA-3 Hashing Time: 0.0003490447998046875

### Findings:

SHA-3 was the fastest operation.

AES was faster than DES, making it more efficient and secure.

DES is outdated and slower than AES.

## 6. Conclusion and Recommendations

Through this project, different cryptographic techniques were implemented and compared.

### Key Takeaways:

AES is more secure and efficient than DES.

SHA-3 is the fastest operation and provides integrity verification.

Encryption speed varies depending on key size and block mode.

This project successfully demonstrated encryption, decryption, and hashing techniques in cybersecurity.