Hybrid Cryptosystem using Rotor Machine and DES Encryption

Course: Cryptography and Network Security
Institution: UPRM
Members: jonathan.rodriguez72@upr.edu, jose.covas@upr.edu
Professor: Venkataramani Kumar
Deadline: November 21, 2025

Objective:

The objective of the project is to understand the significance of encryption, and effect of multi-layers of encryption on the overall security of the system. The other pertinent objectives pertaining to this project includes but not limited to:

1. Understand the operations of rotor machine, and data encryption standard (DES).
2. Practical implementations of rotor machine, DES, and a hybrid system that com bines both.
3. Compare the security advantages and potential weaknesses of using a hybrid cryp tographic system.
4. Gain practical experience in implementing encryption algorithms and managing encryption keys.

Overview of the project:

In this project you will be working on constructing a hybrid cryptosystem that comprises of a Rotor Machine, and DES. This two layer encryption will add on to the security of the information being transmitted. As shown in the Fig. 1 the hybrid cryptosystem comprises of two stages namely rotor machine, and DES. The information to be transmitted, referred to as **M**, is first encrypted using Rotor machine, this encrypted text (also known as **E1**) is once encrypted using DES, the resultant encrypted text is known as **E2**. **E2** is the double encrypted ciphertext. Similarly, the decryption process is performed in the reverse direction. **E2** is decrypted using DES, the result deciphered text is referred to as **D1** which is once again decrypted using rotor machine to get a final decrypted text **D2**. You need to test if the **M** and **D2** are same.

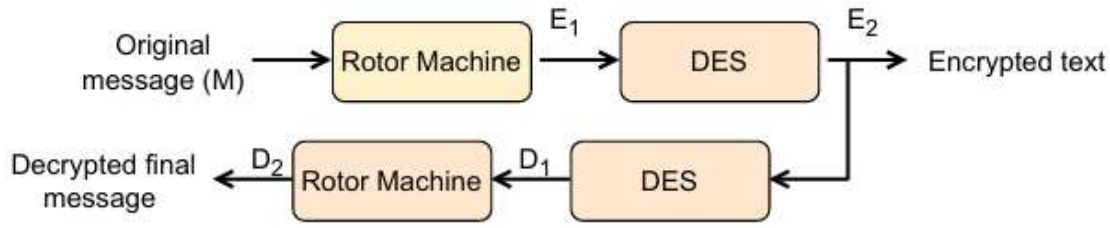


Figure 1: Hybrid Cryptosystem.

Notations used in this project

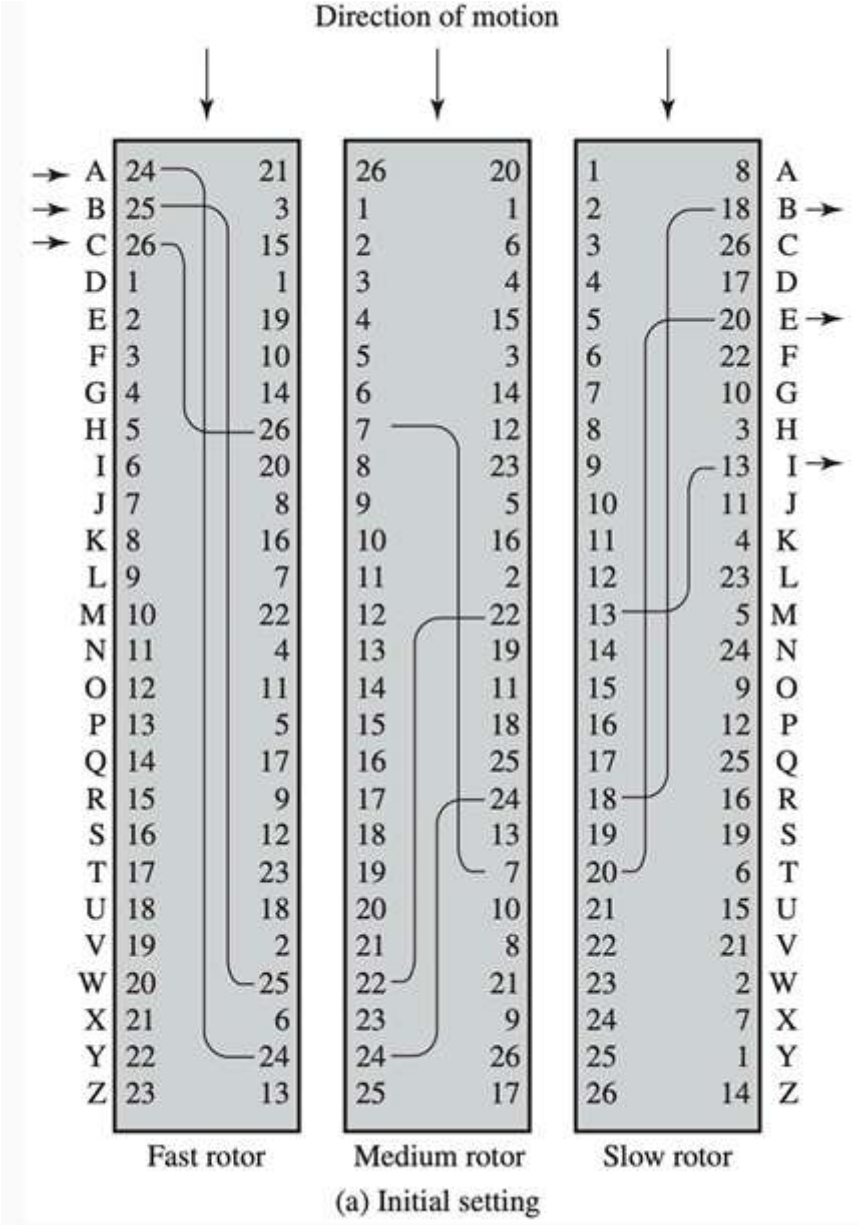
Notation	Description
M	Original message to be transmitted
E1	First encryption using rotor machine
E2	Second encryption using DES
D1	First decryption using DES
D2	Second decryption using rotor machine

Evaluation Rubric

Criteria	Points
Task-1 (Implement rotor machine)	20
Task-2 (Implementing DES)	20
Task-3 (Implement hybrid Cryptographic system)	20
Discussion Questions	30
Bonus	10
Total	**100**

Task-1: Implement rotor machine- In this task a classic rotor machine should be implemented subject to the following conditions:

- Rotor machine should have 3 rotors
- Develop a code that simulates the working of rotor machine in such a way that each rotor should rotate after character is encrypted.
- The developed system should include provisions for decryption process too.
- Test your rotor machine design using the following messages: **'HELLO'**, **'HOPE'**, and **'NEW YEAR'**. Document your results in the format shown in Table 2.



(a) Initial setting

Table 2: Documenting the results of rotor machine

Initial Information	Encrypted Text	Decrypted Text
HOPE		
HELLO		
NEW YEAR		

In [1]:
```
import pandas as pd

from rotor_machine import RotorMachine
from des_encryption import DESEncryption
from hybrid_cryptosystem import HybridCryptosystem

M1 = "HOPE"
```

```
M2 = "HELLO"
M3 = "NEW YEAR"


import time
import psutil


def get_time_wall() -> float:
    """Returns real time in milliseconds
    Returns:
        float: wall-clock time (ms)
    """
    return time.perf_counter() * 1000.0


def get_cpu_time() -> float:
    """Returns CPU time in milliseconds
    Returns:
        float: CPU time (ms)
    """
    return time.process_time() * 1000.0


def get_cpu_utilization(simulation_time: float) -> float:
    """Returns CPU utilization in percentage
    Args:
        simulation_time (float): elapsed wall-clock time
    Returns:
        float: CPU utilization (%)
    """
    if simulation_time > 0:
        return psutil.cpu_percent(interval=0.1)
    return 0.0


def performance_metrics(
    class_object: RotorMachine | DESEncryption | HybridCryptosystem, text
):
    simulation_time = get_time_wall()
    encrypted_text = class_object.encrypt(text)
    decrypted_text = class_object.decrypt(encrypted_text)
    simulation_time = get_time_wall() - simulation_time
    cpu_utilization = get_cpu_utilization(simulation_time)

    return simulation_time, cpu_utilization
```

In [2]:

```
# Create rotor machine
rotor_machine = RotorMachine()

# Perform encryption
rotor_machine_E1 = rotor_machine.encrypt(M1)
rotor_machine_E2 = rotor_machine.encrypt(M2)
rotor_machine_E3 = rotor_machine.encrypt(M3)

# Perform decryption
```

```
rotor_machine_D1 = rotor_machine.decrypt(rotor_machine_E1)
rotor_machine_D2 = rotor_machine.decrypt(rotor_machine_E2)
rotor_machine_D3 = rotor_machine.decrypt(rotor_machine_E3)

# Print the results
pd.DataFrame(
    {
        "Initial Information": [M1, M2, M3],
        "Encrypted Text": [rotor_machine_E1, rotor_machine_E2, rotor_machine_E3],
        "Decrypted Text": [rotor_machine_D1, rotor_machine_D2, rotor_machine_D3],
    }
).style.set_caption("Table 2: Documenting the results of rotor machine").hide(
    axis="index"
)
```

Out[2]:

Table 2: Documenting the results of rotor machine

Initial Information	Encrypted Text	Decrypted Text
HOPE	8 2r	HOPE
HELLO	8r□□	HELLO
NEW YEAR	Mr1F□r;B	NEW YEAR

Task-2: Implementing DES- In this task DES scheme should be implemented.

- Test the operation of the developed DES scheme using the same three original messages given in Task-1, and document the results once again similar to Table 2.

Table 3: Documenting the results of DES

Initial Information	Encrypted Text	Decrypted Text
HOPE		
HELLO		
NEW YEAR		

In [3]:

```
# Create DES object
des = DESEncryption()

# Perform encryption
des_E1 = des.encrypt(M1)
des_E2 = des.encrypt(M2)
des_E3 = des.encrypt(M3)

# Perform decryption
des_D1 = des.decrypt(des_E1)
des_D2 = des.decrypt(des_E2)
des_D3 = des.decrypt(des_E3)

# Print the results
pd.DataFrame(
```

```
    {
        "Initial Information": [M1, M2, M3],
        "Encrypted Text": [des_E1, des_E2, des_E3],
        "Decrypted Text": [des_D1, des_D2, des_D3],
    }
).style.set_caption("Table 3: Documenting the results of DES").hide(axis="index")
```

Out[3]:

Table 3: Documenting the results of DES

Initial Information	Encrypted Text	Decrypted Text
HOPE	mÿ]é□T »	HOPE
HELLO	%Æ□{îÚ□Ò	HELLO
NEW YEAR	·Äq□🡅 ¾¤	NEW YEAR

Task-3: Implement hybrid Cryptographic system- In this task two layer encryption will be done. First, M will be passed through rotor machine to get E1, which then should be passed through DES to get E2. In the similar manner decryption process should be done. Test the developed hybrid system on the following messages, and document the results as shown in Table 4

Table 4: Documenting the results of hybrid cryptosystem

| Original Information (M)|E1| E2| D1| D2| |----------|---|---|---|---| | HOW ARE YOU|||| | HAPPY NEW YEAR|||| | WELCOME TO PUERTO RICO||||

In [4]:

```
# Implementing the hybrid cryptographic system

# Original Information (M)
M1 = "HOW ARE YOU"
M2 = "HAPPY NEW YEAR"
M3 = "WELCOME TO PUERTO RICO"

hybrid_cryptosystem = HybridCryptosystem()

# Encryption using rotor machine E1 and Encryption using DES E2
# Decryption using DES D2 and Decryption using rotor machine D1

# Encryption and decryption of M1
E21 = hybrid_cryptosystem.encrypt(M1)
E11 = hybrid_cryptosystem.get_E1()
D21 = hybrid_cryptosystem.decrypt(E21)
D11 = hybrid_cryptosystem.get_D1()

# Encryption and decryption of M2
E22 = hybrid_cryptosystem.encrypt(M2)
E12 = hybrid_cryptosystem.get_E1()
D22 = hybrid_cryptosystem.decrypt(E22)
D12 = hybrid_cryptosystem.get_D1()

# Encryption and decryption of M3
E23 = hybrid_cryptosystem.encrypt(M3)
```

```
E13 = hybrid_cryptosystem.get_E1()
D23 = hybrid_cryptosystem.decrypt(E23)
D13 = hybrid_cryptosystem.get_D1()

# Print the results
pd.DataFrame(
    {
        "Original Information (M)": [
            M1,
            M2,
            M3,
        ],
        "E1": [E11, E12, E13],
        "E2": [E21, E22, E23],
        "D1": [D11, D12, D13],
        "D2": [D21, D22, D23],
    }
).style.set_caption("Table 4: Documenting the results of DES").hide(axis="index")
```

Out[4]:

Table 4: Documenting the results of DES

Original Information (M)	E1	E2	D1	D2
HOW ARE YOU	.p)gMrfg0p|	Å÷ÞSÓGññ'z□*µ³□Ù	.p)gMrfg0p|	HOW ARE YOU
HAPPY NEW YEAR	.Mkk0g!f)g0fMr	?□□Õàðéµ Þ5¨»□"□□	.Mkk0g!f)g0fMr	HAPPY NEW YEAR
WELCOME TO PUERTO RICO	)f;cp$fgapgk|frapgrQcp	□tp□SsgåÇÍÙ□□~ aÊ4□pZ$¯	)f;cp$fgapgk|frapgrQcp	WELCOME TO PUERTO RICO

Discussion Questions: Based on the hybrid cryptosystem answer the following questions:

1. How does the initial rotor position affect the encryption process? Provide examples to demonstrate how different initial rotor settings produce different ciphertexts for the same plaintext.

- The first rotor is the fastest rotor. The rotor position determines the specific substitution alphabet used for the first plaintext character and then shifts the entire sequence of substitutions for the rest of the text. This initial configuration affects in the way that the same plaintext encrypted with different initial settings will produce entirely different ciphertexts. Depending on this first position will affect both of the other two rotor's positions, affecting the ciphertext.

In [5]:

```
pd.DataFrame(
    {
```

```
        "Plaintext": [
            "HELLO",
            "HELLO",
        ],
        "Initial Rotor Position": ["A", "B"],
        "Ciphertext Generated": ["IGOPS", "JHPQT"],
        "Encryption Steps to get to first letter": [
            "H -> I : The shift is only one",
            "H -> J : The shift is two with this setting",
        ],
    }
).style.set_caption(
    "Examples of different initial rotor positions and their corresponding cipherte
).hide(
    axis="index"
)
```

Out[5]:

Examples of different initial rotor positions and their corresponding ciphertexts

Plaintext	Initial Rotor Position	Ciphertext Generated	Encryption Steps to get to first letter
HELLO	A	IGOPS	H -> I : The shift is only one
HELLO	B	JHPQT	H -> J : The shift is two with this setting

2. Discuss the process of decryption in a rotor machine. How does the structure of the machine ensure that decryption with the same rotor settings as encryption will return the original message?

- The decryption in a rotor machine is essentially the reverse of the encryption process, performed by inputting the ciphertext back into the machine using the identical settings. Since the entire process is reciprocal, feeding the ciphertext through the machine with the same rotor settings undoes the substitution step-by-step, perfectly restoring the original plaintext.

3. Consider two scenarios: (1) Using the same initial rotor positions for multiple messages and (2) Using different initial rotor positions for each message. How do these scenarios impact the overall security of the rotor machine encryption?

- The first scenario eventually weakens the security of the encryption process since it will make it easier for an attacker to crack and identify the substitutions that are being made. The second scenario is way more secure since it makes sure there is a unique substitution sequence done each time, improving the overall security.

4. Discuss the key schedule in DES. How are the 56-bit keys generated for each round of the encryption process? What effect does this have on the strength of the encryption?

- The 56-bits come from an original 64-bit key that undergoes a process of initial permutation and the removal of 8 parity bits. After the removal of these bits a left circular shift is done on the 56-bit key for the same amount as the round number before going through the process of permuted choice 2. The first round's key goes through one left circular shift while the fourth round's key goes through four left circular shifts. This process introduces confusion and diffusion by ensuring each round uses a different subset and arrangement of key bits improving the overall strength of the encryption process.

5. DESuses 16 Feistel rounds. Explain why multiple rounds of encryption (in this case, 16) are necessary to create a secure cipher. What would happen if fewer rounds were used? Justify your response by considering few examples.

- Multiple rounds of encryption are necessary in DES to achieve sufficient confusion and diffusion. Each of the 16 rounds uses a different subkey to thoroughly mix the plaintext, forcing it to be affected by the entire key. Using fewer rounds would just decrease the amount of diffusion, making it more predictable and less secure. One example that would explain how it is less secure is if we use two rounds. The output of this encryption would be much easier to decrypt than the one with 16 since the relation from the plaintext to the ciphertext is much closer. This two round example could easily be attacked by a linear cryptoanalysis to quickly decipher the key. If we increase the rounds to 8, the encryption process is safer than the previous example but it is still more prone to be decrypted easier than the 16 round one.

6. What is the role of padding in DES encryption? Why is padding necessary, and what padding schemes can be used? What are the potential issues that could arise if padding is not handled correctly?

- The role of padding in the DES encryption is to make sure that every block encrypted has 64-bits. Padding is necessary to maintain the standard of 64 bit blocks. In our case our padding scheme used was that padding bits were applied using a numeric scheme where the number of missing bytes is repeated as 8-bit binary values to fill the block. If padding is not handled correctly it will lead to errors in the decryption process.

7. Consider the computational performance of the hybrid cryptosystem with standalone rotor machine, and DES using the evaluation metrics such as computational complexity, resource utilized, and the total time taken and document the results in Table 4. Analyze the results by explaining the benefits, and limitations of hybrid cryptographic systems.

- The hybrid cryptosystem combines the efficiency of the DES and rotor machine algorithms with the strong, non-repudiable key exchange of an asymmetric algorithm. One limitation that can be obtained from the results is that there is an increase in

computation time since it is performing two distinct cryptographic operations. Although its computation time is higher, the hybrid cryptographic system's resource utilization is the lowest from the three.

In [6]:
```
# Rotor Machine Performance Metrics
rotor_machine = RotorMachine()
rotor_machine_cc = "O(N) + O(1)"
rotor_machine_ct, rotor_machine_ru = performance_metrics(rotor_machine, M1)

# DES Performance Metrics
des = DESEncryption()
des_cc = "O(N) + O(1) + O(k)"
des_ct, des_ru = performance_metrics(des, M1)

# Hybrid Cryptosystem Performance Metrics
hybrid_cryptosystem = HybridCryptosystem()
hybrid_cryptosystem_cc = "O(N)+ O(1) + O(k)"
hybrid_cryptosystem_ct, hybrid_cryptosystem_ru = performance_metrics(
    hybrid_cryptosystem, M1
)

# Print the results
pd.DataFrame(
    {
        "Metric": [
            "Computational Complexity",
            "Computation time (s)",
            "Resource Utilized",
        ],
        "Rotor Machine": [rotor_machine_cc, rotor_machine_ct, rotor_machine_ru],
        "Des": [des_cc, des_ct, des_ru],
        "Hybrid Cryptosystem": [
            hybrid_cryptosystem_cc,
            hybrid_cryptosystem_ct,
            hybrid_cryptosystem_ru,
        ],
    }
).style.set_caption(
    "Table 5: Comparing the performances of three different cryptographic systems"
).hide(
    axis="index"
)
```

Out[6]:

Table 5: Comparing the performances of three different cryptographic systems

Metric	Rotor Machine	Des	Hybrid Cryptosystem
Computational Complexity	O(N) + O(1)	O(N) + O(1) + O(k)	O(N)+ O(1) + O(k)
Computation time (s)	0.159100	2.365200	3.697300
Resource Utilized	18.200000	3.300000	7.400000

8. Discuss the practicality of using a hybrid cryptosystem like Rotor Machine + DES in modern encryption applications. Could this approach be applied in real-world situations? If so, suggest few potential places where it can be applied.

- Using a hybrid cryptosystem like Rotor Machine + DES in modern encryption applications might not be the safest and most secure approach to take since these models can be considered a little outdated in relation to modern encryption applications. However, the approach of combining two encryption methods is very impactful since it helps make the process of encryption more secure. Combining other two more modern algorithms could be a very effective and more secure system. Doing some research, we found that the TLS and SSL protocols on secure web browsing use a hybrid approach. These protocols use asymmetric encryption to exchange a symmetric key. we also found that the hybrid approach is used in the process of virtual private networks, more commonly known as VPNs.

In [7]:
```
from tests import run_all_tests

run_all_tests()
```

```
Running tests...

Starting DES parser test...
Split into blocks test passed.
Parse test passed.
Deparse test passed.

DES parser test completed.
Passed DES parser test.

Starting DES bit converter test...
String to binary test passed.
Binary to string test passed.

DES bit converter test completed.
Passed DES bit converter test.

Starting DES permutation test...
Initial permutation test passed.
DES permutated choice 1 test passed.
DES permutated choice 2 test passed.
Expansion test passed.
P-box test passed.
Inverse initial permutation test passed.

DES permutation test completed.
Passed DES permutation test.

Starting DES test...
DES encryption test passed.

DES test completed.
Passed DES test.

Starting rotor machine test...
Default rotor machine test passed.
Custom rotor machine test passed.

Rotor machine test completed.
Passed rotor machine test.

Starting hybrid cryptosystem test...
Hybrid cryptosystem test passed.

Hybrid cryptosystem test completed.
Passed hybrid cryptosystem test.

All tests completed.

Test Results: Passed: 6, Failed: 0
```