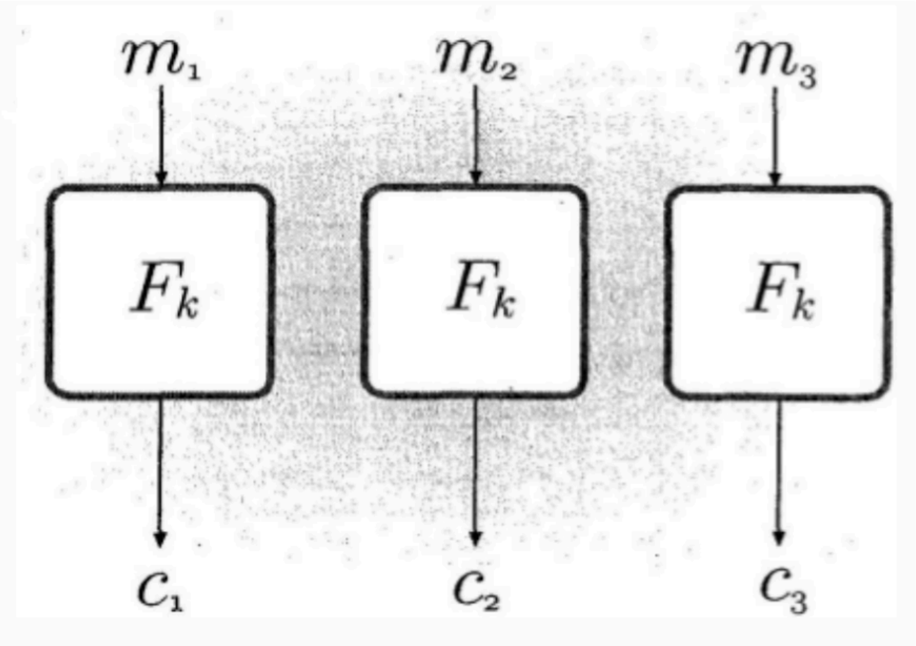
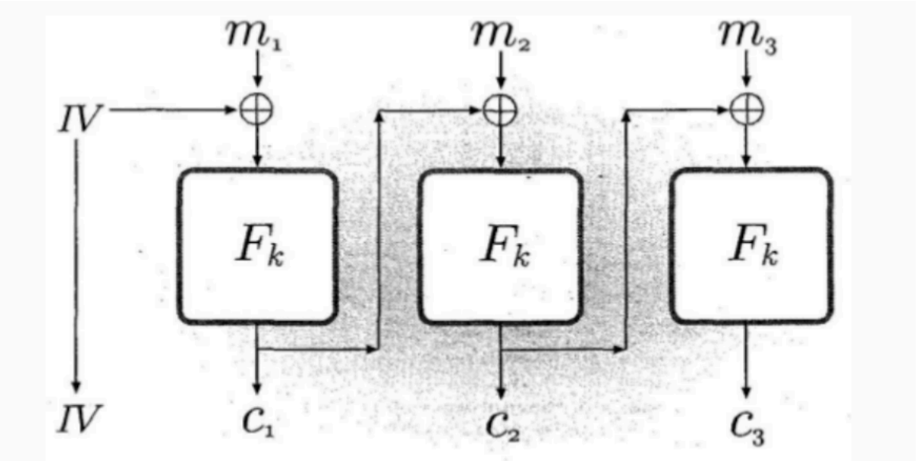
DES
Take message
Encrypt with key A
Then take output
Decrypt with key B
Then take output
Encrypt with key A
Put output as answer

AES
ECB
Easiest method
Take a line of plain text and put in part 4
Take key and put in part 4
Copy to output to notepad
Do for each line of plaintext
And put together as answer

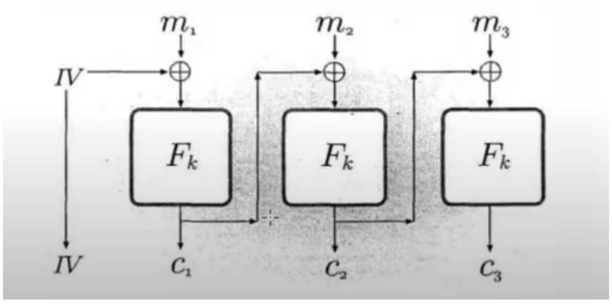
operation.pdf)



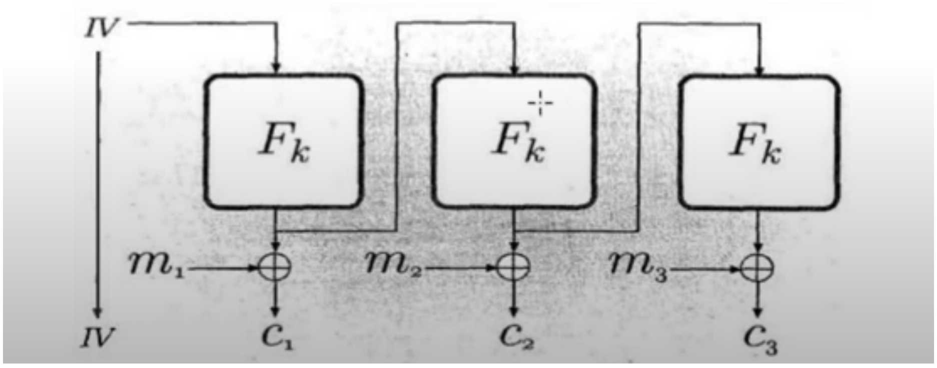
Electronic Code Book(ECB) mode



Cipher block changing
Take IV and put in notepad
Now take this IV and put in XOR along with a plaintext
Take XOR output along with key and put in part 4
Copy this output in notepad and this is the new IV
Take new IV put in XOR along with new plaintext
Put XOR output in part 4
Repeat for all plaintext

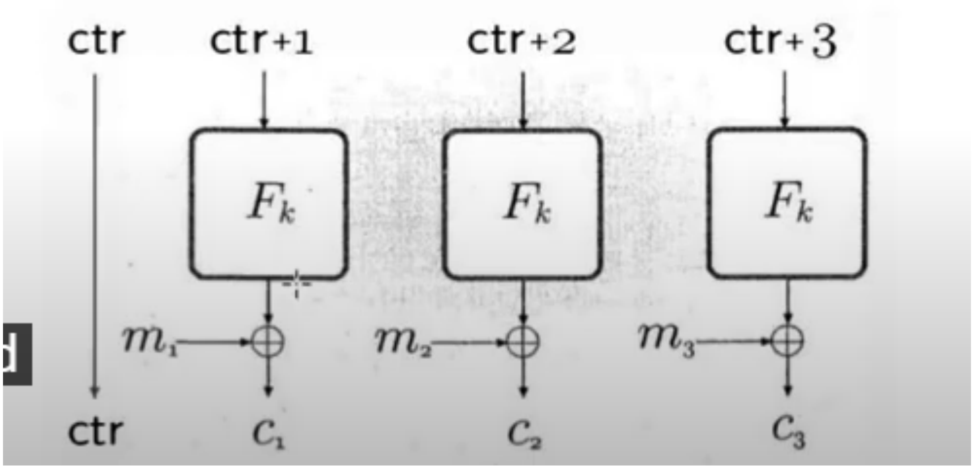


Output Feedback
Take IV put in notepad
Take this IV with key and put in part 4
Taje part 4 output and put in XOR along with a plaintext
Put XOR output in notepad
Now take above mentioned part 4 output and encrypt it with key in part 4
Take this new part 4 output and XOR it with new plaintext
Repeat

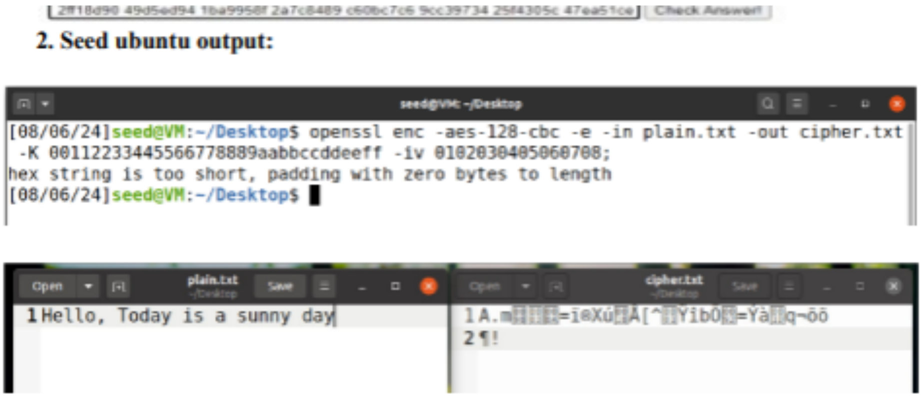


Counter Mode

First take ctr and put in notepad
Increment ctr
Put first ctr in part 4 with key
Put key and paintext in XOR and put answer in notepad
Take the new ctr and put in part 4 with original key
Take new part 4 answer with new plaintext and XOR it and put it in notepad
Take a new ctr again put in part 4, take part 4 output with new plaintext and put in XOR and put XOR output in notepad
Repeat for all plaintext



SEED UBUNTU OUTPUT


2ff18d90 49d5ed94 1ba9958f 2a7c8489 c60bc7c6 9cc39734 25f4305c 47ea51ce

**2. Seed ubuntu output:**

```
[08/06/24]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher.txt
 -K 00112233445566778889aabbccddeeff -iv 0102030405060708;
hex string is too short, padding with zero bytes to length
[08/06/24]seed@VM:~/Desktop$ 
```

```
Hello, Today is a sunny day
```

```
A.m▯▯▯=ï®Xú▯Ä[^▯Yìb0▯=Yà▯q¬õõ
¶!
```

HMACC

# Experiment no. 3

**Aim:** Cryptographic Hash Functions and Applications (HMAC).

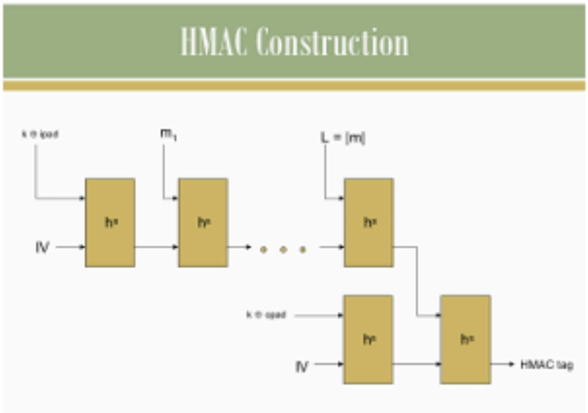
**Theory:**

What is HMAC?

HMAC algorithm stands for Hashed or Hash-based Message Authentication Code. It is a result of work done on developing a MAC derived from cryptographic hash functions. HMAC is a great resistance to cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC and thus is more secure than any other authentication code. RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security. The FIPS 198 NIST standard has also been issued by HMAC.

Objectives of HMAC

- As the Hash Function, HMAC is also aimed to be one way, i.e., easy to generate output from input but complex the other way around.
- It aims at being less affected by collisions than the hash functions.
- HMAC reuses algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- HMAC tries to handle the Keys in a more simple manner.



HMAC Construction

Steps for HMAC

1. Familiarize yourself with the working of SHA-1. Though we would be using a dummy hash in the sequel for simplicity, in general, you could be using SHA-1 instead

2. Select a plaintext for which the HMAC tag is to be computed.(by clicking on NextPalintext Button)

3. For simplicity fix l=8 which is default,but it should be l < (length of plaintext)/4.

4. Select an Initialization Vector, IV of length l.by clicking on "Next IV" button)

5. Use the ipad and opad as described in theory part to compute the ciphertext with the help of the hash function provided to you.

6. Divide generated plaintext 'm' into say 'k' chunks of 8 bits and kth chunk will have bits less than 8,to make it 8-bits by padding zeros at end

7. Compute z0="IV||(k XOR ipad)" manually where || impies concatenation and enter z0 in "Your text" field to get z1

8. Compute z1="z0||m1" manually where || impies concatenation and enter z1 in "Your text" field to get z2

9. Repeat above step and finally compute z(k+1)="zk||L" where L=|m|,make L 8-bits by padding zeros to left of it

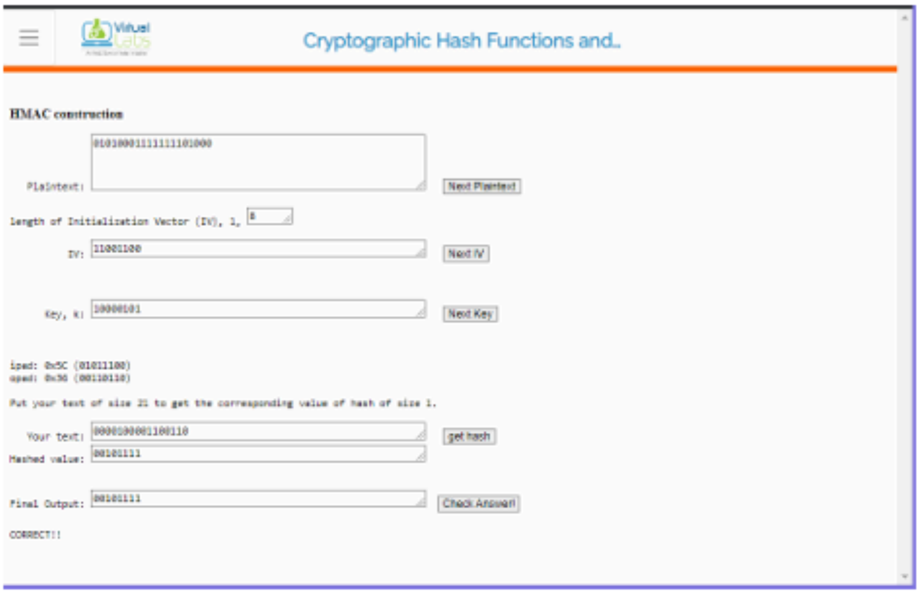
10. Compute p="IV||(k XOR opad)" manually where || impies concatenation and enter p in "Your text" field to get q

11. Compute r="q||z(k+1)" manually where || impies concatenation and enter 'r' in "Your text" field to get final HMAC tag 't'

12. Notice that z0,z1,z2,............zk,z(k+1),p,r are all of size '2l'(=16 in our case as l=8).

13. Write the final cipher text 't' in 'Final Output' field and check your answer

**Output:-**



**HMAC construction**

Plaintext: 01010001111111101000 [Next Plaintext]

length of Initialization Vector (IV), l, 8

IV: 11001100 [Next IV]

Key, k: 10000101 [Next Key]

ipad: 0x5C (01011100)
opad: 0x36 (00110110)

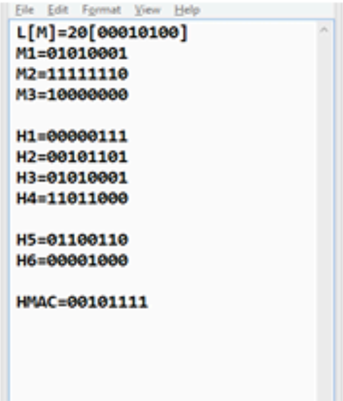
Put your text of size 21 to get the corresponding value of hash of size l.

Your text: 0000100001100110 [get hash]

Hashed value: 00101111

Final Output: 00101111 [Check Answer!]

CORRECT!!



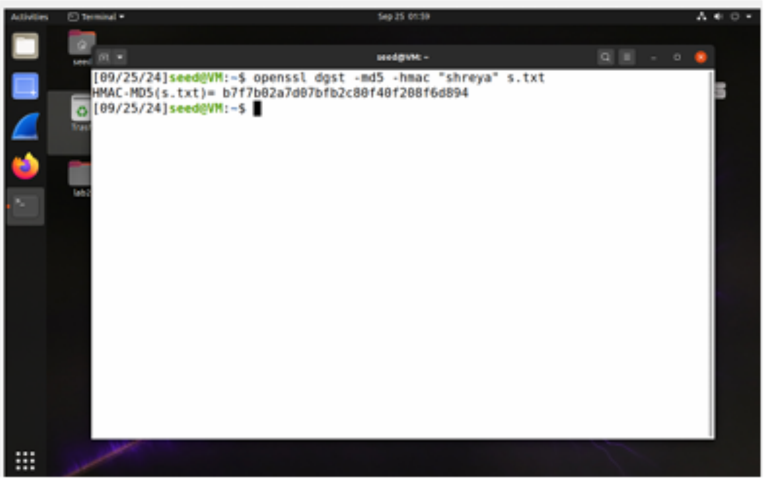
```
L[M]=20[00010100]
M1=01010001
M2=11111110
M3=10000000

H1=00000111
H2=00101101
H3=01010001
H4=11011000

H5=01100110
H6=00001000

HMAC=00101111
```

**<u>Seed Ubuntu</u> :-**



```
[09/25/24]seed@VM:~$ openssl dgst -md5 -hmac "shreya" s.txt
HMAC-MD5(s.txt)= b7f7b02a7d07bfb2c80f40f208f6d894
[09/25/24]seed@VM:~$ 
```

**Conclusion:** Hence, we understand, what is HMAC, how HMAC is constructed and how plantext is converted into cipher text.