MITE 433: Cybersequrity Lab

# Secret-Key Encryption Lab Manual (Linux + OpenSSL)

Manual: Secret‑Key Encryption (Lab)   
  
Lab Setup  
---------  
1. Ensure prerequisites  
 - A Linux environment (Ubuntu, Debian, Fedora, etc.)  
 - openssl installed (check via `openssl version`)  
 - Terminal (shell) access  
  
2. Create a working directory & sample plaintext  
 mkdir ~/secret\_key\_lab  
 cd ~/secret\_key\_lab

$ nano plaintext.txt

Welcome to IIT DU Lab.

ls -l plaintext.txt

Caesar Cipher (Shift Cipher)

Create a plaintext file:

echo "HELLO WORLD" > plaintext.txt

cat plaintext.txt

Output

HELLO WORLD

Encrypt with shift (example: shift 3):

tr 'A-Z' 'D-ZA-C' < plaintext.txt > ciphertext.txt

cat ciphertext.txt

**Output:**

KHOOR ZRUOG

Decrypt back:

tr 'D-ZA-C' 'A-Z' < ciphertext.txt > decrypted.txt

cat decrypted.txt

**Exercise:** Try shift values of 5 and 13 (ROT13).

Lab 2: Playfair Cipher

**Step 1: Generate Playfair key square (e.g., KEYWORD)**

Remove duplicates: KEYWORDABCFGHIJLMNPQSTUVXZ

Construct 5×5 matrix (J merged with I):

K E Y W O

R D A B C

F G H I L

M N P Q S

T U V X Z

Save this in a text file:

cat > key\_matrix.txt << 'EOF'

K E Y W O

R D A B C

F G H I L

M N P Q S

T U V X Z

EOF

Step 2: Prepare plaintext

echo "HELLOCRYPTO" | tr -d ' ' | tr 'J' 'I' > plaintext.txt

cat plaintext.txt

Output

HELLOCRYPTO

Step 3: Split into digraphs (pairs)

fold -w2 plaintext.txt

Output  
HE

LL

OC

RY

PT

O

### Step 1: Create Key Matrix

We will store the 5×5 matrix in a file using **heredoc**:

cat > key\_matrix.txt << 'EOF'

K E Y W O

R D A B C

F G H I L

M N P Q S

T U V X Z

EOF

cat key\_matrix.txt

**Output:**

K E Y W O

R D A B C

F G H I L

M N P Q S

T U V X Z

### Step 2: Prepare Ciphertext

Create a ciphertext file:

cat > ciphertext.txt << 'EOF'

BMODZBXDNABEKUDMUIXMMOUVIF

EOF

cat ciphertext.txt

**Output:**

BMODZBXDNABEKUDMUIXMMOUVIF

### Step 3: Split into Digraphs

fold -w2 ciphertext.txt > digraphs.txt

cat digraphs.txt

**Output:**

BM

OD

ZB

XD

NA

BE

KU

DM

UI

XM

MO

UV

IF

### Step 4: Playfair Decryption Script (bash + awk)

Create a script playfair\_decrypt.sh:

cat > playfair\_decrypt.sh << 'EOF'

#!/bin/bash

matrix\_file="key\_matrix.txt"

digraphs\_file="digraphs.txt"

# Read matrix into array

mapfile -t matrix < <(awk '{for(i=1;i<=NF;i++) print $i}' $matrix\_file)

# Function to find row,col in matrix

find\_pos() {

local ch=$1

for r in {0..4}; do

for c in {0..4}; do

idx=$((r\*5 + c))

if [ "${matrix[$idx]}" = "$ch" ]; then

echo "$r $c"

return

fi

done

done

}

# Decrypt a digraph

decrypt\_pair() {

local a=$1

local b=$2

read r1 c1 <<< $(find\_pos $a)

read r2 c2 <<< $(find\_pos $b)

if [ $r1 -eq $r2 ]; then

# Same row: shift left

c1=$(((c1+4)%5))

c2=$(((c2+4)%5))

elif [ $c1 -eq $c2 ]; then

# Same column: shift up

r1=$(((r1+4)%5))

r2=$(((r2+4)%5))

else

# Rectangle: swap columns

tmp=$c1

c1=$c2

c2=$tmp

fi

idx1=$((r1\*5 + c1))

idx2=$((r2\*5 + c2))

printf "%s%s" "${matrix[$idx1]}" "${matrix[$idx2]}"

}

# Process all digraphs

while read -r line; do

a=${line:0:1}

b=${line:1:1}

decrypt\_pair $a $b

done < $digraphs\_file

echo

EOF

Make the script executable:

chmod +x playfair\_decrypt.sh

### Step 5: Run Decryption

./playfair\_decrypt.sh

**Expected output:**

HELLOCRYPTOX

Remove the padding X if added during encryption.

### Notes: The script works **entirely in Linux terminal** with bash + awk. It can be extended to read **plaintext from a file** and write output to a file:

./playfair\_decrypt.sh > decrypted.txt

cat decrypted.txt

# For Modern Crypto Algorithm

Step 1: Key & IV generation (explicit approach)

In many secret key labs, the instructor shows how to generate a random encryption key and initialization vector (IV) first.  
------------------------------------------------  
# Generate a 256-bit (32 bytes) key in hex

openssl rand -hex 32 > key.hex

# Generate a 128-bit (16 bytes) IV in hex

openssl rand -hex 16 > iv.hex

# Check sizes

wc -c key.hex iv.hex

* key.hex now holds 64 hex characters (32 bytes)
* iv.hex has 32 hex characters (16 bytes)

Step 2: Encrypt the plaintext with OpenSSL (AES‑CBC mode)  
----------------------------------------------------------  
Using the key and IV from above:

openssl enc -aes-256-cbc \

-in plaintext.txt \

-out ciphertext.bin \

-K $(cat key.hex) \

-iv $(cat iv.hex)

**Notes:**

-aes-256-cbc is a block cipher mode (AES with 256-bit key, Cipher Block Chaining).

-K expects the key in hex.

-iv expects IV in hex.

ciphertext.bin is raw binary ciphertext.

Step 3: Decryption  
-------------------  
To decrypt using the same key/IV:

openssl enc -d -aes-256-cbc \

-in ciphertext.bin \

-out decrypted.txt \

-K $(cat key.hex) \

-iv $(cat iv.hex)

If you had used base64:

openssl base64 -d -in ciphertext.b64 | \

openssl enc -d -aes-256-cbc \

-K $(cat key.hex) \

-iv $(cat iv.hex) \

-out decrypted.txt

Check that decrypted.txt matches plaintext.txt

diff plaintext.txt decrypted.txt && echo "Decryption OK"

Step 4: (Alternate) Password-based encryption  
----------------------------------------------  
Encrypt:

openssl enc -aes-256-cbc -salt -pbkdf2 -iter 100000 -in plaintext.txt -out ciphertext\_pass.enc

Decrypt:

openssl enc -d -aes-256-cbc -salt -pbkdf2 -iter 100000 -in ciphertext\_pass.enc -out decrypted\_pass.txt

Check:

cat decrypted\_pass.txt

**Clean up:** After lab, optionally delete sensitive files securely (e.g. shred key.hex iv.hex ).