

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Mật mã và An ninh mạng (TN) - CO3070

Báo cáo

BÀI THỰC HÀNH SỐ 1

Giảng viên hướng dẫn: ThS. Nguyễn Cao Đạt

Sinh viên thực hiện: 2014486 - Đậu Xuân Thành

TP. Hồ Chí Minh, 04/2024

Mục lục

1. Các hệ mã đối xứng truyền thống	3
1.1. Câu 1	3
1.2. Câu 2	4
1.3. Câu 3	5
1.4. Câu 4	6
1.5. Câu 5	6
1.6. Câu 6	6
1.7. Câu 7	7
2. Chuẩn mã hóa dữ liệu DES	9
2.1. Câu 1	9
2.2. Câu 2	9
2.2.1. Tính khoá con K1 được sử dụng cho vòng mã hoá đầu tiên	9
2.2.2. Tính L0, R0	10
2.2.3. Tính kết quả mở rộng R0: E[R0], với E là hàm mở rộng	10
2.2.4. Tính giá trị $A = E[R0] \oplus K1$	10
2.2.5. Chia 48-bit kết quả ở câu d và chia thành các nhóm 6 bit, thực hiện tính toán trên từng nhóm 6 bit thông qua S-box, ghi lại kết quả.	10
2.2.6. Nối các kết quả tính được ở câu e thành chuỗi kết quả 32-bit, ghi lại kết quả dưới dạng binary (B).	10
2.2.7. Tính giá trị P(B), với P là hàm hoán vị	10
2.2.8. Tính giá trị $R1 = P(B) \oplus L0$	10
2.2.9. Ghi lại kết quả ciphertext cho vòng thứ nhất	11
Tài liệu tham khảo	12

Danh mục hình ảnh

1. Các hệ mã đối xứng truyền thống

1.1. Câu 1

Xét bảng tần suất số lần xuất hiện các chữ cái trong văn bản tiếng Anh, ta thấy E có tần suất xuất hiện nhiều nhất (12, 31)

⇒ Chúng ta bắt đầu xét quy luật bắt đầu từ ký tự E.

Ta có thể lấy được khóa bằng cách lấy *hiệu* của vị trí trong bảng chữ cái ký tự E và các ký tự có tần suất xuất hiện nhiều trong bản mã.

$$\text{key} = \text{index}(E) - \text{index}(x)$$

Sau khi có *key*, ta thử từng *key* cho vào hàm giải mã và thử cho đến khi tìm được một chuỗi có nghĩa.

Tần suất của các ký tự còn lại trong bản mã:

```
Lab01 python scripts/cisear.py
> Frequence of chars:

N: 5
X: 5
S: 5
W: 5
J: 5
F: 5
I: 5
M: 4
K: 3
L: 2
B: 2
Y: 2
R: 2
T: 2
D: 2
G: 1
A: 1
Q: 1
```

Sử dụng Python để hiện thực hàm giải mã:

```
from collections import Counter

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

def decrypt(message, key):
    prt = key
    key = ord(key) - ord('E')
    translated = ""
    for letter in message:
        if letter in LETTERS:
            num = LETTERS.find(letter)
            gap = num - key
            if gap < 0:
                gap += len(LETTERS)
            elif gap > 0:
```

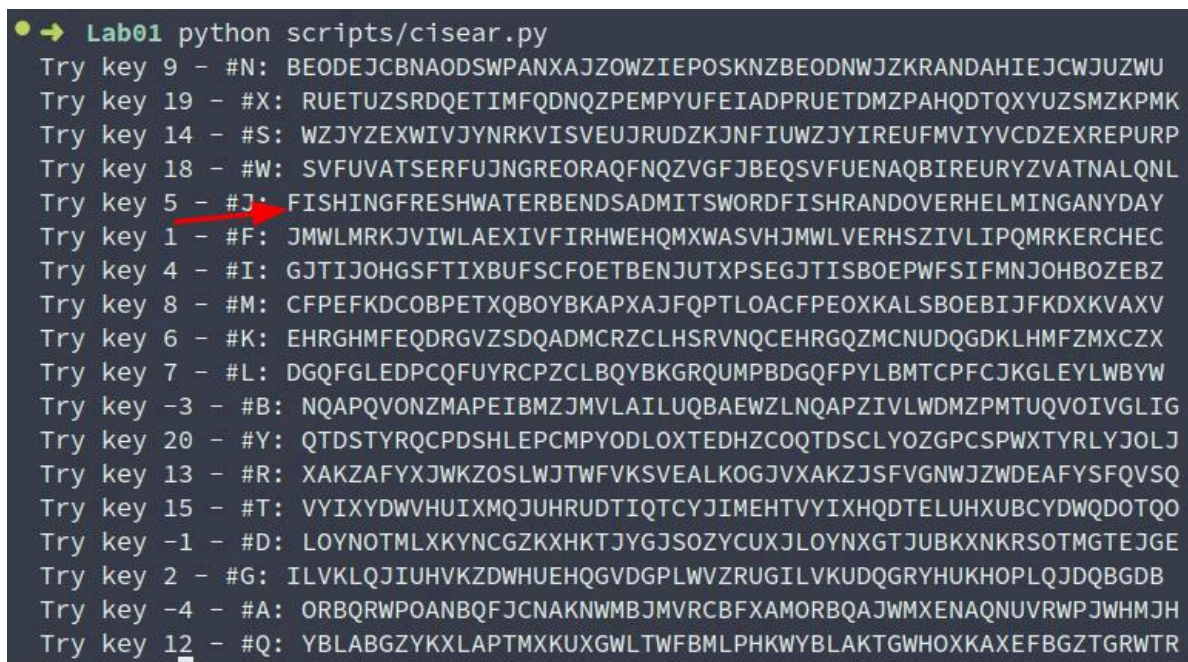
```

gap -= len(LETTERS)
try:
    translated = translated + LETTERS[gap]
except Exception as e:
    print(f"{e} - {gap}")
else:
    translated = translated + letter
print(f'Try key {key} - #{prt}: {translated}')

if __name__ == '__main__':
    message = "KNXMNLSLKWJXMBFYJWGJSIXFIRNYXBTWIKNXMWFSITAJWMJQRNSLFSIDFD"
    char_count = Counter(message)
    sorted_char_count = char_count.most_common()
    print(f"> Frequence of chars: \n")
    for char, count in sorted_char_count:
        print(f"{char}: {count}")
    message = message.upper()
    for i in sorted_char_count:
        decrypt(message, i[0])

```

Kết quả:



```

Lab01 python scripts/cisear.py
Try key 9 - #N: BEODEJCBNAODSWPANXAJZOWZIEPOSKNZBEODNWJZKRANDAHIEJCWJUZWU
Try key 19 - #X: RUETUZSRDQETIMFQDNQZPEMPYUFEIADPRUETDMZPAHQDTQXYUZSMZKPMK
Try key 14 - #S: WZJYZEXWIVJYNRKVISVEUJRUDZKJNFIUWZJYIREUFMVIYVCDZEXREPURP
Try key 18 - #W: SVFUVATSERFUJNGREORAQFNQZVGFJBEQSVFUENAQBIREURYZVATNALQNL
Try key 5 - #J: FISHINGFRESHWATERBENDSADMITSWORDFISHRANDOVERHELMINGANYDAY
Try key 1 - #F: JMWLMRKJVIWLAEXIVFIRHWEHQMXWASVHJMWLVERHSZIVLIPQMRKERCHEC
Try key 4 - #I: GJTIOHGSFTIXBUFSOETBENJUTXPSEGJTISBOEPWFSIFMNJOHBOZEBZ
Try key 8 - #M: CFPEFKDCOBPETXQBOYBKAPXAJFQPTLOACFPEOXKALSBOEBIJFKDXKVAXV
Try key 6 - #K: EHRGHMFEQDRGVZSDQADMCRZCLHSRVNQCEHRGQZMCNUDQGDKLHMFZMXCZX
Try key 7 - #L: DGQFGLDPCQFUYRCPZCLBQYBKGRQUMPBDGQFPYLBMTCPFCJKGLEYLWBYW
Try key -3 - #B: NQAPQVONZMAPEIBMZJMVLAILUQBAEWZLNQAPZIVLWDMZPMTUQVOIVGLIG
Try key 20 - #Y: QTDSTYRQCPDSHLEPCMPYODLOXTEDHZCOQTDSCLYOZGPCSPWXTYRLYJOLJ
Try key 13 - #R: XAKZAFYXJWKZOSLWJTWFKVSVEALKOGJVXAKZJSFVGNWJZWDEAFYSFQVSQ
Try key 15 - #T: VYIXYDWHUIXMQUHRUDTIQTCYJIMEHTVYIXHQDTELUXHUBCYDWQDOTQO
Try key -1 - #D: LOYNOTMLXKYNCGZXHKTYGJSOZYCUXJLOYNXGTJUBKXNKRSTMGTEJGE
Try key 2 - #G: ILVKLQJIIUHVKZDWHUEHQGVGDPLWVZRUGILVKUDQGRYHUKHOPLQJQDBGDB
Try key -4 - #A: ORBQRWPOANBQFJCNAKNWMBJMVRCBFXAMORBQAJWMXENANUVRWPJWHMJH
Try key 12 - #Q: YBLABGZYKXLAPTMXKUXGWLTFBMLPHKWYBLAKTGWHOXKAXEFBGZTGRWTR

```

Kết luận:

- Vậy, bản mã được giải với $key = 5$, thu được kết quả là:
FISHINGFRESHWATERBENDSADMITSWORDFISHRANDOVERHELMINGANYDAY
- Điểm yếu của giải thuật Caesar: Dễ dàng bị phương pháp vét cạn giải mã.

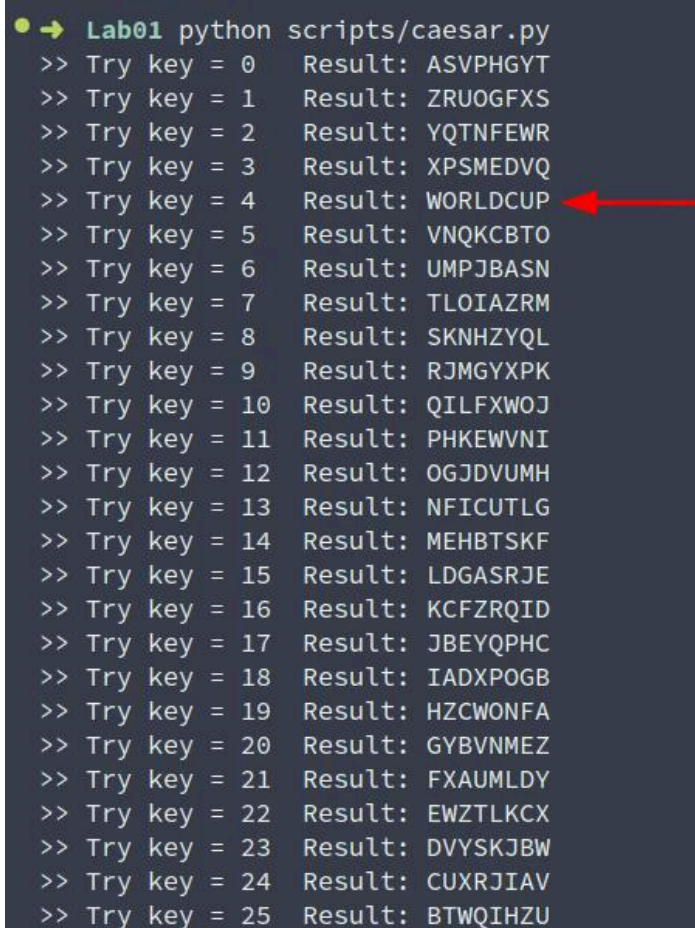
1.2. Câu 2

Chúng ta tiến hành vét cạn để tìm khóa K , khóa K được vét cạn từ $0 \rightarrow 25$. Lần lượt cho các ký tự trong ciphertext trừ đi khóa K để tìm được vị trí của plaintext.

Sử dụng Python

```
def decrypt_replace(message, key):
    translated = ""
    for symbol in message:
        if symbol in LETTERS:
            num = LETTERS.find(symbol)
            num = num - key
            if num < 0:
                num = num + len(LETTERS)
            translated = translated + LETTERS[num]
        else:
            translated = translated + symbol
    print(f">>> Try key = {key}\t Result: {translated}")
if __name__ == '__main__':
    message = 'asvphgyt'
    for i in range(26):
        decrypt_replace(message.upper(), i)
```

Kết quả:



```
Lab01 python scripts/caesar.py
>> Try key = 0    Result: ASVPHGYT
>> Try key = 1    Result: ZRUOGFXS
>> Try key = 2    Result: YQTNFEWR
>> Try key = 3    Result: XPSMEDVQ
>> Try key = 4    Result: WORLD CUP
>> Try key = 5    Result: VNQKCBTO
>> Try key = 6    Result: UMPJBASN
>> Try key = 7    Result: TLOIAZRM
>> Try key = 8    Result: SKNHZYQL
>> Try key = 9    Result: RJMGYXPK
>> Try key = 10   Result: QILFXWOJ
>> Try key = 11   Result: PHKEWVNI
>> Try key = 12   Result: OGJDVUMH
>> Try key = 13   Result: NFICUTLG
>> Try key = 14   Result: MEHBTSKF
>> Try key = 15   Result: LDGASRJE
>> Try key = 16   Result: KCFZRQID
>> Try key = 17   Result: JBEYQPHC
>> Try key = 18   Result: IADXPOGB
>> Try key = 19   Result: HZCWONFA
>> Try key = 20   Result: GYBVNMEZ
>> Try key = 21   Result: FXAUMLDY
>> Try key = 22   Result: EWZTLKCX
>> Try key = 23   Result: DVYSKJBW
>> Try key = 24   Result: CUXRJIAV
>> Try key = 25   Result: BTWQIHZU
```

Kết luận:

- Khóa **K** = 4
- Thông điệp: *WORLD CUP*

1.3. Câu 3

Dựa vào bảng tần suất ở câu 1, ta thấy tần suất xuất hiện của **E**, **T** theo thứ tự là cao nhất và cao thứ 2. Vậy ta chọn **B** đại diện cho **E**, **U** đại diện cho **T**. Ta có hệ phương trình:

$$\begin{cases} (E \rightarrow B)(4 \times a + b) \bmod 26 = 1 \\ (T \rightarrow U)(19 \times a + b) \bmod 26 = 20 \end{cases} \Rightarrow \begin{cases} a = 3 \\ b = 15 \end{cases}$$

1.4. Câu 4

2 vấn đề chính của OTP:

1. Khoá phải có độ dài bằng thông điệp: Trong OTP, khoá được sử dụng một lần duy nhất và phải có cùng độ dài với thông điệp cần mã hóa. Điều này đòi hỏi việc tạo, quản lý và phân phát khoá rất khó khăn khi thông điệp cần mã hóa có kích thước lớn hoặc cần gửi qua mạng. Nếu khoá ngắn hơn thông điệp, hoặc nếu cùng một khoá được sử dụng nhiều lần, thì mật mã OTP có thể bị tấn công và không còn an toàn nữa.
2. Bảo mật khoá: Khoá trong OTP phải được duy trì với mức độ bảo mật cao và không được tiết lộ cho bất kỳ ai trừ người nhận. Việc bảo mật khoá là quan trọng vì nếu một kẻ tấn công có thể truy cập hoặc đoán được khoá, thì an toàn của mật mã OTP sẽ bị đe dọa. Nếu khoá bị lộ, thông điệp có thể bị dễ dàng giải mã và thông tin sẽ không còn bí mật nữa.

Những vấn đề này khiến cho việc triển khai và sử dụng mật mã OTP trở nên khó khăn trong thực tế. Tuy OTP có tính bảo mật tuyệt đối khi được thực hiện đúng cách, nhưng việc duy trì khoá và đảm bảo khoá không bị rò rỉ là một nhiệm vụ khó khăn đối với các hệ thống thực tế.

1.5. Câu 5

M = Must see you over Cadogan West. Coming at once.

Mã hóa:

- Loại bỏ dấu câu, chuyển các ký tự viết hoa thành viết thường, chia M thành các cặp ký tự:

M = 'mu' 'st' 'se' 'ey' 'ou' 'ov' 'er' 'ca' 'do' 'ga' 'nw' 'es' 'tc' 'om' 'in' 'ga' 'to' 'nc' 'ez' ← Thêm z vào cuối ký tự 'e' đứng 1 mình.

- Áp dụng giải thuật:

```
ey -> gz
ou -> pn
ov -> nw
er -> lg
ca -> tg
do -> tu
ga -> er
nw -> ov
es -> ld
tc -> db
om -> uh
in -> fp
ga -> er
to -> hw
nc -> qs
ez -> de
```

⇒ Kết quả: C = gzpnnwlggtuerovlldbuhfperhwqsde.

1.6. Câu 6

M = spyarrivesonthursday key1 = XUAN, key2 = THANH

Mã hóa

- Key 1: XUAN

4	3	1	2
X	U	A	N
s	p	y	a
r	r	i	v
e	s	o	n
t	h	u	r
s	d	a	y

Kết quả: C1 = yiouaavnryprshdsrets

- Key 2: THANH

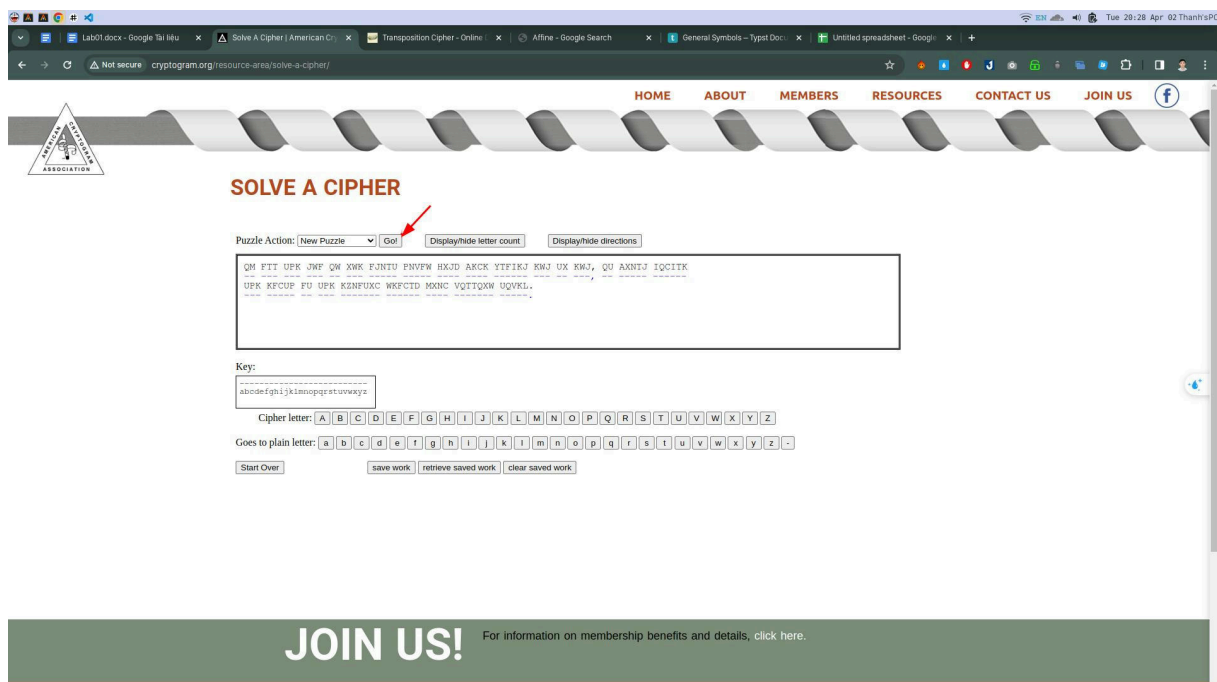
5	2	1	4	3
T	H	A	N	H
y	i	o	u	a
a	v	n	r	y
p	r	s	h	d
s	r	e	t	s

Kết quả: C2 = onseivrraydsurhtyaps

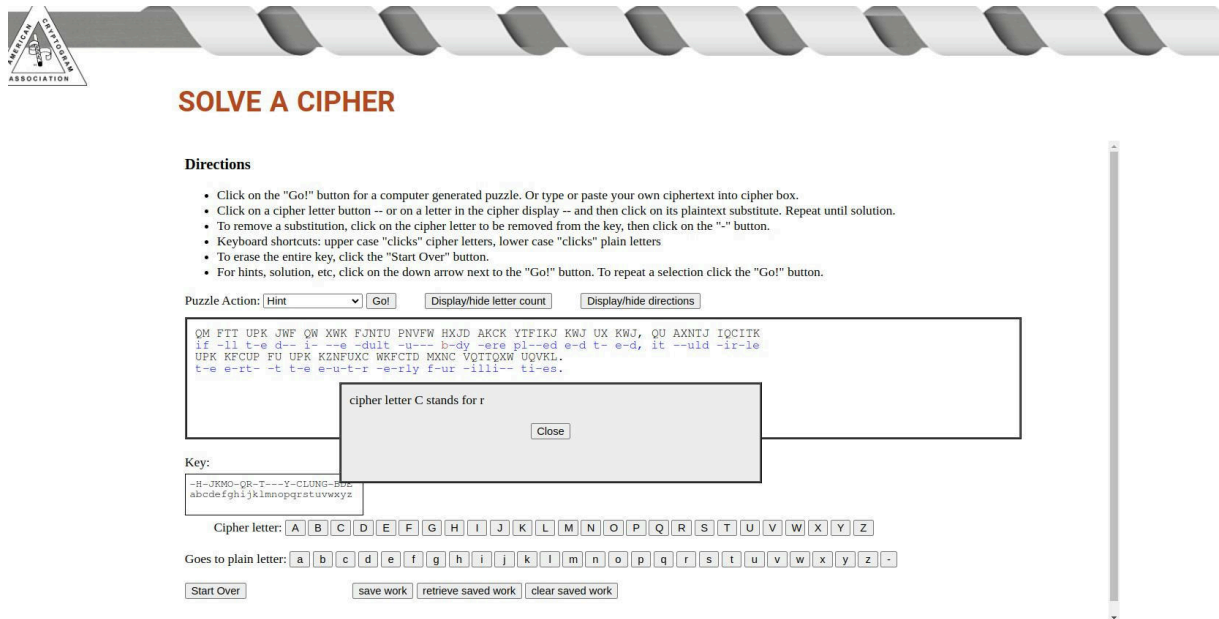
1.7. Câu 7

Cách chơi

- Click *New Puzzle* → *GO!*: Để tạo game mới.



- Click *Hint* : Để xem các hint.



SOLVE A CIPHER

Directions

- Click on the "Go!" button for a computer generated puzzle. Or type or paste your own ciphertext into cipher box.
- Click on a cipher letter button -- or on a letter in the cipher display -- and then click on its plaintext substitute. Repeat until solution.
- To remove a substitution, click on the cipher letter to be removed from the key, then click on the "-" button.
- Keyboard shortcuts: upper case "clicks" cipher letters, lower case "clicks" plain letters
- To erase the entire key, click the "Start Over" button.
- For hints, solution, etc, click on the down arrow next to the "Go!" button. To repeat a selection click the "Go!" button.

Puzzle Action: **Hint** | Go! | Display/hide letter count | Display/hide directions

QM FIT UPK JWF QW XWK FJNTU PNVEW HXJD AKCK YTFIKJ KWI UX KWI, QU AXNTJ IQCIK
if -ll t-e d-- i- --e -dult -u-- b-dy -ere pl--ed e-d t- e-d, it --uld -ir-le
UPK KFCUP FU UPK KZNFUXC WKFCID MXNC VQTIQXN UQVKL.
t-e e-tt- -t t-e e-u-t-r -e-ry f-ur -illi- ti-es.

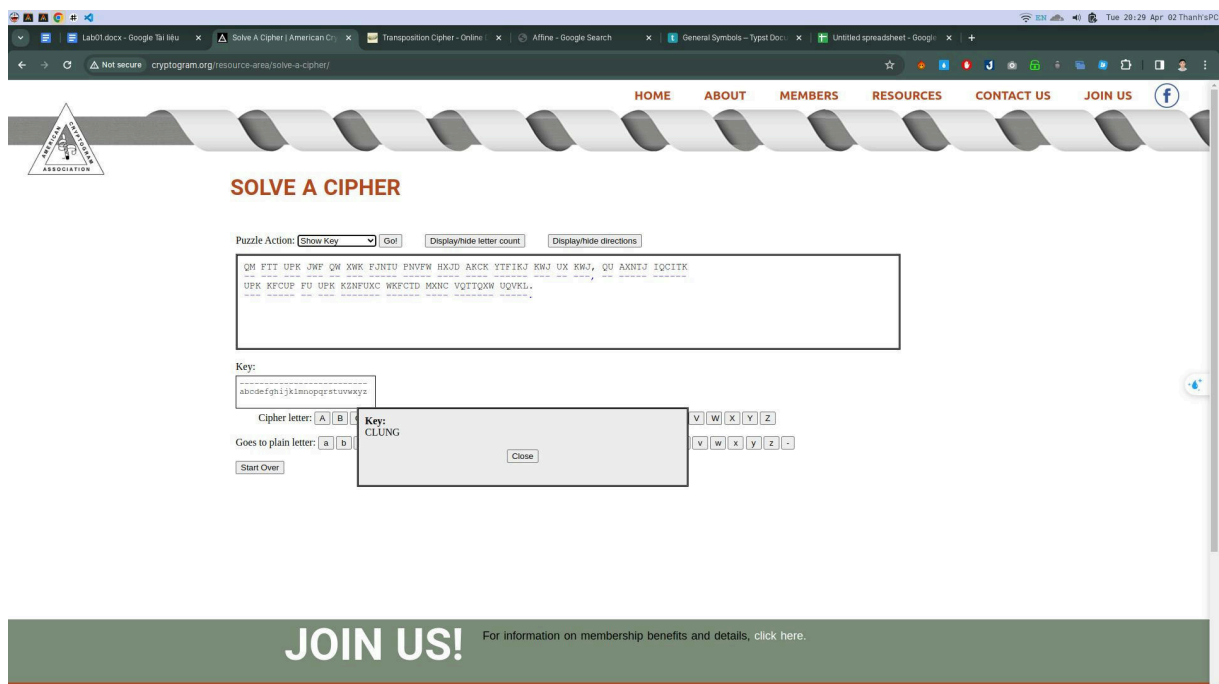
Key:
-H-JKMO-QR-T---Y-CLUNG-BZ
abcde fgh i j k l m n o p q r s t u v w x y z

Cipher letter: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Goes to plain letter: a b c d e f g h i j k l m n o p q r s t u v w x y z -

Start Over | save work | retrieve saved work | clear saved work

- Click *Show Key*: Để xem Key, Key = CLUNG



SOLVE A CIPHER

Puzzle Action: **Show Key** | Go! | Display/hide letter count | Display/hide directions

QM FIT UPK JWF QW XWK FJNTU PNVEW HXJD AKCK YTFIKJ KWI UX KWI, QU AXNTJ IQCIK
UPK KFCUP FU UPK KZNFUXC WKFCID MXNC VQTIQXN UQVKL.

Key:
abcde fgh i j k l m n o p q r s t u v w x y z

Cipher letter: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Goes to plain letter: a b c d e f g h i j k l m n o p q r s t u v w x y z -

Start Over | save work | retrieve saved work | clear saved work

JOIN US! For information on membership benefits and details, click here.

- Từ các hint có được ta sẽ map các letter ở "Cipher letter" với "Plain letter"

NOTE: Từ hint, có thể thấy "C stand for r" → cipher letter là C ứng với plain letter là r, từ đây ta có thể đoán nhanh được các ký tự còn lại trong Key (CLUNG):

- C → r
- L → s
- U → t
- N → u
- G → v

Các ký tự còn lại ta có thể dựa vào hint để map tiếp.

Và đây là kết quả sau khi map xong:

- Click on the "Go!" button for a computer generated puzzle. Or type or paste your own ciphertext into cipher box.
- Click on a cipher letter button -- or on a letter in the cipher display -- and then click on its plaintext substitute. Repeat until solution.
- To remove a substitution, click on the cipher letter to be removed from the key, then click on the "-" button.
- Keyboard shortcuts: upper case "clicks" cipher letters, lower case "clicks" plain letters
- To erase the entire key, click the "Start Over" button.
- For hints, solution, etc, click on the down arrow next to the "Go!" button. To repeat a selection click the "Go!" button.

Puzzle Action:

QM FTT UPK JWF QW XWK FJNTU PNVFW HXJD AKCK YIFIKJ KWJ UX KWJ, QU AXNTJ IQCITK
if all the dna in one adult human body were placed end to end, it would circle
UPK KFCUP FU UPK KZNFUXC WKECTD MXNC VQTTQXW UQVKL.
the earth at the equator nearly four million times.

Puzzle solved! . . . Key solved!

Key:

F H I J K M O P Q R S T V W X Y Z C L U N G A B D E
a b c d e f g h i j k l m n o p q r s t u v w x y z

Cipher letter:

Goes to plain letter:

2. Chuẩn mã hóa dữ liệu DES

2.1. Câu 1

So sánh giữa mã hóa dòng và mã hóa khối

Mã hóa dòng	Mã hóa khối
Xử lý các thông điệp theo từng khối	Xử lý các thông điệp theo từng bit hay byte tại mỗi thời điểm.
Từng khối này sẽ được mã hóa hoặc giải mã.	Từng bit hay byte sẽ được mã hóa hoặc giải mã.
Phạm vi ứng dụng lớn hơn.	Phạm vi ứng dụng nhỏ
Thường chậm hơn so với mã hóa dòng do phải xử lý từng khối một.	Thường nhanh hơn với tốc độ ổn định, lý tưởng cho việc mã hóa dữ liệu trực tiếp trong thời gian thực
Có khả năng chống lại các tấn công trao đổi khối (block swapping) và một số tấn công khác	Có thể dễ bị tấn công nếu không có cơ chế kiểm soát lỗi cụ thể.

2.2. Câu 2

$M = 0123456789ABCDEF$

$K = 0123456789AB4486$

2.2.1. Tính khoá con K_1 được sử dụng cho vòng mã hoá đầu tiên

$M = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$ $K = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 0100\ 0100\ 1000\ 0110$ Sử dụng ma trận PC-1 để hoán vị K ta có K_+

$K_+ = 1011000\ 0010011\ 0000101\ 0100000\ 1010101\ 0110011\ 0000110\ 0000000$

$C_0 = 1011000\ 0010011\ 0000101\ 0100000$

$D_0 = 1010101\ 0110011\ 0000110\ 0000000$

$$C_1 = 0110000\ 0100110\ 0001010\ 1000001$$

$$D_1 = 0101010\ 1100110\ 0001100\ 0000001$$

Kết hợp C_1 và D_1 ta được:

$$C_1 D_1 = 0110000\ 0100110\ 0001010\ 1000001\ 0101010\ 1100110\ 0001100\ 0000001$$

Sử dụng ma trận PC-2, để hoán vị $C_1 D_1$ ta được K_1 :

$$\Rightarrow K_1 = 000000\ 110000\ 001000\ 000111\ 100110\ 110000\ 000110\ 100101\ (48\ bits)$$

2.2.2. Tính L_0, R_0

$$M = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$$

Sử dụng ma trận IP ta hoán vị M được IP:

$$IP = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111\ 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010\ (64\ bits)$$

Từ đó, ta có được L_0, R_0 bằng cách lấy 32bits đầu và cuối:

$$L_0 = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111$$

$$R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

2.2.3. Tính kết quả mở rộng $R_0: E[R_0]$, với E là hàm mở rộng

Ta có, $R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$.

Sử dụng ma trận E-Bit Selection, ta có thể tính $E(R_0)$:

$$\Rightarrow E(R_0) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101\ (48\ bits)$$

2.2.4. Tính giá trị $A = E[R_0] \oplus K_1$

$$A = 011110\ 010001\ 011101\ 010010\ 111000\ 010001\ 010011\ 110000$$

2.2.5. Chia 48-bit kết quả ở câu d và chia thành các nhóm 6 bit, thực hiện tính toán trên từng nhóm 6 bit thông qua S-box, ghi lại kết quả.

- $S1(B1) = 0111$
- $S2(B2) = 1100$
- $S3(B3) = 1111$
- $S4(B4) = 0010$
- $S5(B5) = 0110$
- $S6(B6) = 0110$
- $S7(B7) = 0011$
- $S8(B8) = 0000$

2.2.6. Nối các kết quả tính được ở câu e thành chuỗi kết quả 32-bit, ghi lại kết quả dưới dạng binary (B).

$$S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0111\ 1100\ 1111\ 0010\ 0110\ 0110\ 0011\ 0000$$

2.2.7. Tính giá trị $P(B)$, với P là hàm hoán vị

$$P(B) = 0000\ 0110\ 0110\ 1101\ 1000\ 0111\ 1001\ 1110$$

2.2.8. Tính giá trị $R1 = P(B) \oplus L0$

$$R1 = P(B) \oplus L0 = 1100\ 1010\ 0110\ 1101\ 0100\ 1011\ 0110\ 0001$$

2.2.9. Ghi lại kết quả ciphertext cho vòng thứ nhất

Cipher text of Round 1: F0AAF0AACA6D4B61



Tài liệu tham khảo