Homework 3. Encryption

Document version 1.1 25 Aug 2021, 12:13

การบ้าน คราวนี้เป็นเรื่องเกี่ยวกับสตริง
โจทย์เป็นเรื่องการเข้ารหัสลับ โปรแกรมของคุณต้องรับอินพุตเป็นสตริงสองอัน
สตริงอันที่หนึ่งเป็นรหัสเรียกว่า key
สตรองอันที่สองเป็นข้อความที่คุณต้องการเข้ารหัสเรียกว่า plaintext
เมื่อผ่านโปรแกรมของคุณและพิมพ์ออกมาที่เอาท์พุตจากกายเป็นสตริงที่เข้ารหัสแล้ว

ยกตัวอย่างเช่นอินพุตคือ

```
key = "ABCDEFGHIJKLMNOP"
plaintext = "I LOVE PYTHON101"
```

จะได้เอาท์พุตคือ

```
4kZ;SR2U$;p2\<n\
```

ผมยังไม่ได้คิดวิธีถอดรหัสกลับมานะครับ

วิธีการของเราได้รับแรงบันดาลใจมาจากการเข้ารหัสมาตรฐานเออีเอส สามารถอ่านเพิ่มเติมได้ที่
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

ผมจะอธิบายวิธีการทำการบ้าน ควบคู่ไปกับการเขียนโปรแกรมทีละสเต็ป

ก่อนอื่นเราจะนิยามสตริงก่อน สตริงของเราในการบ้านนี้เป็นสตริงที่พิมพ์ออกมาดูได้ และเป็นภาษาอังกฤษ ซึ่งมีรหัสแอสกีระหว่าง 32 ถึง 90 ความยาวสตริงบังคับไว้ที่ 16 ตัวอักษร เทียบเท่ากับ รหัสยาว 128 บิต

คุณสามารถดูการเข้ารหัสแอสกีได้จากที่นี่
https://www.ascii-code.com/

นั่นก็คืออินพุตของเราต้องเป็นชุดของตัวอักษรดังต่อไปนี้

```
 !"#$%&'()*+,-./0123456789:;<=>@ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

รวมถึงช่องว่างด้วยหนึ่งตัว

วิธีการเข้ารหัสของเรามีสี่ขั้นตอน

addkey, subbyte, shiftrow, mixcolumn

และทำเช่นนี้สองรอบ
โค้ดชั้นนอกสุดจะมีหน้าตาดังนี้

```
# homework 3. Encryption

def encrypt(key,text):
    t1 = oneround(key,text)
    t2 = oneround(key,t1)
    return t2

def oneround(key,text):
    t1 = addkey(key,text)
    t2 = subbyte(t1)
    t3 = shiftrow(t2)
    t4 = mixcolumn(t3)
    return t4
```

ดูโปรแกรมไปเลยทีละขั้น

---

1) addkey คือเอา plaintext กับ key มาทีละตัวอักษร จับเอา ค่ารหัสแอสกีมา xor กัน เสร็จแล้วแปลงกลับมาให้เป็นตัวอักษรที่พิมพ์ได้ จะเอาค่าแอสกีของอักขระใช้ ord(c) เช่น ord("A") คือ 65

```
def addkey(k,t):
    s = doxor(k[0],t[0])
    s += doxor(k[1],t[1])
    s += doxor(k[2],t[2])
    . . .
    return s

def doxor(c1,c2):
    c3 = . . .
    c4 = makeprintable(c3)
    return c4

def makeprintable(c):
    c1 = ((c % 90) + 32 % 90)
    return chr(c1)
```

การทำให้ตัวอักษรพิมพ์ได้ (makeprintable()) ใช้วิธีบังคับให้อยู่ในช่วงรหัสที่กำหนด

```
key = "ABCDEFGHIJKLMNOP"
plaintext = "I LOVE PYTHON101"
```

ตัวอย่างเช่นเอา key กับ plaintext ข้างบนมาทำ addkey จะได้ เอาต์พุตนี้

```
s = addkey(key, plaintext)
print(s)

((/+3#-80>###EE'
```

---

2) subbyte (substitution byte) คือการแทนตัวอักษรของ plaintext ด้วยตัวอักษรอื่น

วิธีนี้เป็นการเข้ารหัสลับอันแรกสุดที่เราค้นพบในสมัยโบราณ อ่านได้ที่นี่
https://en.wikipedia.org/wiki/Caesar_cipher

การแปลงสตริงหนึ่งไปเป็นอีกสตริง วิธีทำใช้สตริงสองชุด ชุดหนึ่งเป็นตัวอักษรอินพุตอีกชุดหนึ่งเป็นตัวอักษรเอาพุต เหมือนที่ทำในโจทย์ในเกรดเดอร์

ผมกำหนด สตริงเพื่อทำการแทนตัวอักษรให้แล้ว โดยทำเป็นฟังชันให้สร้างสตริงนั้น (ยังเรียนไม่ถึง เอาไปใช้ก่อน) เพื่อหลีกเลี่ยงการพิมพ์ตัวอักษรแปลกแปลกเข้าไป

```
subinput = createsubinput()
suboutput = createsuboutput()

print(subinput)
print(suboutput)
```

จะได้สตริง

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

จะเห็นว่าการแทนของเราคือการเลื่อนตัวสูงขึ้นเช่นจาก A เป็น B
ลองใช้ subbyte ดู

print(subbyte(plaintext)) จะได้

```
J!MPWF!QZUIPO212
```

---

3) shiftrow คือการเลื่อนสตริงไปทางขวาโดยเราจะเลื่อนไปทางขวาสองครั้งการเลื่อนเป็นแบบวงกลม คือตัวที่หลุดไปทางขวาจะโผล่มาเข้าข้างหน้าทางซ้าย ดังตัวอย่างต่อไปนี้

```
def shiftrow(t):
    t1 = shiftonce(t)
    t2 = shiftonce(t1)
    return t2

def shiftonce(t):
    . . .
    return t1
```

ลองใช้ดู

print(shiftrow(plaintext)) จะได้

```
01I LOVE PYTHON1
```

---

4) mixcolumn ขั้นตอนสุดท้ายก็คือเอาตัวอักขระมาคูณกับ fix polynomial ในที่นี้เราใช้คูณค่าแอสกี กับตัวอักขระโดยฝังตัวสัมประสิทธิ์การคูณเข้าไปในโค้ดเลย

```
# fixpoly = "2311123111233112"

def mixcolumn(t):
    s = mulc(t[0], "2")
    s += mulc(t[1], "3")
    s += mulc(t[2], "1")
    . . .
    return s

def mulc(c1,c2):
    . . .
    c4 = makeprintable(c3)
    return c4
```

ลองใช้ดู

```
print(mixcolumn(plaintext))

R,B!j>,RIb e2],4
```

ผมให้โค้ดที่เป็นโครงไว้ข้างล่างแล้วนะครับ
งานของคุณคือการเติมโปรแกรมบรรทัดที่เป็น... ให้ครบถ้วน
โปรแกรมของคุณรันโดยฟังชัน main() รับอินพุต เป็นสตริง key และ plaintext แล้วพิมพ์ ข้อความที่เข้ารหัสแล้วออกมา

คุณสามารถทดสอบทีละฟังก์ชันย่อยได้ตามตัวอย่างที่ได้อธิบายทีละขั้นนั้น คุณลองเล่นโดยเปลี่ยน สัมประสิทธิ์โน การทำ mixcolumn และเปลี่ยน ตารางแทน ใน subbyte ได้ การทำรหัสลับที่แข็งแรงเป็นศาสตร์และศิลปะแบบหนึ่ง

ผมยังไม่ได้คิดวิธีถอดรหัสกลับมานะครับ

ขอให้สนุกกับการทำรหัสลับนะครับ

```
============ fragment of code ==========

# encryption  inspired by AES (Advanced Encryption Standard)
# Prabhas Chongstitvatana

key = "ABCDEFGHIJKLMNOP"
plaintext = "I LOVE PYTHON101"

def encrypt(key,text):
    t1 = oneround(key,text)
    t2 = oneround(key,t1)
    return t2

def oneround(key,text):
    t1 = addkey(key,text)
    t2 = subbyte(t1)
    t3 = shiftrow(t2)
    t4 = mixcolumn(t3)
    return t4

# xor key with text, keep it in printable range
def addkey(k,t):
    s = doxor(k[0],t[0])
    s += doxor(k[1],t[1])
    . . .
    return s

# doxor  xor ascii code of two characters c1, c2
def doxor(c1,c2):
    . . .
    c4 = makeprintable(c3)
    return c4

def makeprintable(c):
    c1 = ((c % 90) + 32 % 90)
    return chr(c1)

# substitute characters in t with pattern in subinput and suboutput
def subbyte(t):
    s = findreplace(t[0])
    s += findreplace(t[1])
    s += findreplace(t[2])
    . . .
    return s

# findreplace use subinput and suboutput

def findreplace(c):
    . . .
    return c2
```

```
# these two functions use "for", which we will learn later, just use
# it now
def createsubinput():
    s = ""
    for i in range(32,91):
        s += chr(i)
    return s

def createsuboutput():
    s = ""
    for i in range(33,91):
        s += chr(i)
    s += chr(32)
    return s

subinput = createsubinput()
suboutput = createsuboutout()

# rotate string one character to the right
def shiftonce(t):
    . . .
    return t1

def shiftrow(t):
    t1 = shiftonce(t)
    t2 = shiftonce(t1)
    return t2

# coefficients use in multiply with fix polynomial in mixcolumn
# fixpoly = "2311123111233112"
# we just hardcode it into the function
def mixcolumn(t):
    s = mulc(t[0], "2")
    s += mulc(t[1], "3")
    s += mulc(t[2], "1")
    . . .
    return s

def mulc(c1,c2):
    . . .
    c4 = makeprintable(c3)
    return c4

def test():
    s = ""
    s += "1"
    s += "2"
    print(s)
    txt = "WHATEVER01234567"
    txt2 = subbyte(txt)
    print(plaintext)
    txt4 = oneround(key,plaintext)
```

```
    print(txt4)
    txt5 = encrypt(key,plaintext)
    print(txt5)

def main():
    key  = input . . .
    plaintext = input . . .
    ciphertext = encrypt(key, plaintext)
    print(ciphertext)

main()



####  End  ######
```