Cryptography Engineering Quiz 3 313553024 蘇柏叡

Problem 1.

Description (2.5 pts)

In traditional cryptography, SHA functions (like SHA-256 or SHA-3) are primarily used for hashing and digital signatures. However, SHA-3 (based on the Keccak sponge construction) is more flexible than traditional hash functions due to its sponge-based design. This unique structure allows SHA-3 to be adapted to other cryptographic purposes beyond hashing.

Requirements

1. **Comparison of SHA-256 and SHA-512/256:**
   - Compare the structure and security properties of SHA-256 and SHA-512-truncated-to-256-bits.
   - Do they provide the same level security?
   - Which one is better, and why? Provide a detailed explanation.
2. **Exploring SHA as an Encryption Function:**
   - Can SHA functions (especially SHA-3/SHAKE) be adapted for symmetric encryption?
   - What makes the sponge construction in SHA-3 suitable (or unsuitable) for encryption?
3. **(Extra) Implementation:**
   - Implement a simple encryption/decryption system using SHAKE128 or SHAKE256 in Python or Go:
     - Derive a key from your password or input.
     - Your implementation should generate a keystream for encryption purposes.
     - Develop your own method to transform plaintext into ciphertext using this stream.

1.1 Comparison of SHA-256 and SHA-512/256

a. Block Size:

i. SHA-256: 512 bits

ii. SHA-512/256: 1024 bits

b. Output Length:

i. SHA-256: 輸出為256 bits長度的雜湊值

ii. SHA-512/256: 雖然內部採用的是512 bits 的運算，不過因為輸出會將原始512個bits長度的輸出截斷至256 bits，因此最終輸出仍然為256 bits。

c. Security Properties:

針對碰撞攻擊，不論是SHA-256、SHA-512/256，兩者的碰撞攻擊的複雜度都是(2^{128})，所謂碰撞攻擊指的是在理想條件下，要找到兩筆不同訊息卻產生相同雜湊值需要約(2^{128})的嘗試。至於在Preimage Resistance(從雜湊值逆推出輸入)、Second-Preimage Resistance部分(給定某一訊息及其雜湊值，要再找到另一個不同訊息卻產生相同雜湊值)，同樣複雜度皆為(2^{256})。

d. Whether the Level Security of SHA-256 or SHA-512/256 is same or not:

由面對碰撞攻擊以及在Preimage Resistance、Second-Preimage Resistance這三部分來檢核，我們可發現雖然**SHA-512/256**內部採用的是512 bits的運算，不過由於輸出是截斷至256 bits，因此在上述的攻擊中，**SHA-512/256**其複雜度和**SHA-256**其實是一致的，因此我們可以將兩者視為具有相同的**Level Security**。

e. Which one is better?

SHA-256與SHA-512/256都提供了相當優異的安全屬性，但由於SHA-512/256擁有更大的內部狀態，所以可能在Security Margin方面更勝一籌。至於該如何選擇，還是需要考量系統的效能需求、相容性，以及所需的安全保證等因素。

1.2 Exploring SHA as an Encryption Function:

a. Can SHA functions be adapted for symmetric encryption?

SHA-3系列由於其海綿結構的靈活性，能夠生成可變長度的密鑰流，這使得它**能夠適應對稱加密**。至於在加密過程中，密鑰流與明文可以進行XOR操作，並且可以根據需要生成任意長度的密鑰流，從而提供更高的靈活性和擴展性。

b. What makes the sponge construction in SHA-3 suitable (or unsuitable) for encryption?

i. 優點與合適的原因：在安全性方面，海綿結構確保輸出的密鑰流是pseudo-random的，並且能夠有效抵抗常見的加密攻擊，如：Collision and Pre-Image Attacks。此外，SHA-3的內部狀態可以吸收秘密密鑰來生成密鑰流，這樣能夠將密鑰流與密鑰綁定，進一步增強加密過程的安全性。

ii. 較不合適的原因：海綿結構本身需要較大的state size（SHA-3的state size為1600 bits），這對資源有限的環境時可能會帶來效率上的問題，此外與專門為加密設計的AES演算法相比，海綿結構的計算複雜度較高，這可能導致加密和解密過程的速度較慢。

1.3 SHAKE128 Implementation:

在problem1中main.py，當使用者輸入密碼和明文後，程式會根據密碼生成密鑰流並進行加密。如圖，用戶輸入密碼為313553024、明文為NYCU CS is the best.並使用SHAKE128根據密碼生成一個密鑰流，並使用該密鑰流對明文做XOR，產生f2596a3b74e2c5ba179f8dfc03c5859f20c01e4ee3563a5a1688e47f250048d2ba密文。這個密文是加密後的結果，此外由於XOR操作具備對稱性，在使用相同的密鑰流對密文進行解密操作後，程式可以恢復出原始的明文，以本例而言，當解密過程完成後，程式會顯示解密後的明文NYCU CS is the best.，證明加密與解密過程是正確的，並且能夠順利地恢復原始的資訊。

```
plesase enter your password: 313553024
please enter your plaintext: NYCU is the best.
原文: NYCU is the best.
密文 (16 進制): f2596a3b74e2c5ba179f8dfc03c5859f20c01e4ee3563a5a1688e47f250048d2ba
解密後: NYCU is the best.
```

Problem 2.

Description (3 pts)

The Index of Coincidence (IC) is a fundamental tool in cryptanalysis. It helps analysts determine whether a text is in a natural language, encrypted with a simple substitution cipher, or highly randomized.

The IC is computed using the following formula:

$$IC = \frac{\sum f_i(f_i - 1)}{N(N-1)}$$

Requirements

1. **Understanding IC:**
   - Provide a brief explanation of f_i and N.
2. **Program Implementation:**
   - Write a program in Python or Go to compute the Index of Coincidence for a given ciphertext.
   - Analyze the IC value and determine the probable language of the text.

   Ciphertext:

   ```
   WKHUH DUHWZ RZDBV RIFRQ VWUXF WLQJD VRIWZ DUHGH VLJQR QHZDB
   LVWRP DNHLW VRVLP SOHWK DWWKH UHDUH REYLR XVOBQ RGHIL FLHQF
   LHVDQ GWKHR WKHUZ DBLVW RPDNH LWVRF RPSOL FDWHG WKDWW KHUHD
   UHQRR EYLRX VGHIL FLHQF LHVWK HILUV WPHWK RGLVI DUPRU HGLII LFXOW
   ```

   (hint: ignore the spaces)
3. **IC Value Comparison:**
   - Compare the IC values of English, Chinese, and randomly generated text.
   - Explain the differences and their significance in cryptanalysis.
4. **(Extra) Decryption Challenge:**
   - Attempt to decrypt the ciphertext.
   - Describe your decryption process and methodology.

2.1 Understanding IC:

a. fi:第i種字母在文本中出現的次數。

b. N:文本中所有字母的總數量。

c. 公式解釋: 所有兩兩組合的數量作為計算IC公式的分母，至於分子部分則是所有兩兩字母相同的組合數，換言之，若是某個字母本身出現頻率較高，如:英文的母音，則該字母對於整體IC的貢獻就會越高。

2.2 Understanding IC:

a. 2.2程式碼和2.4程式碼為寫於problem2資料夾內的main.py中，詳細撰寫程式碼的邏輯與方法將會一併於2.4進行說明，本題僅會針對IC value以及可能的語言進行回答。

b. 經過計算後，所得之IC value為0.0681，經查找後發現此值與英文的IC value近乎相同，故我們可推斷此文字使用之語言為英文。

```
Index of Coincidence (IC) = 0.0681
```

2.3 IC Value Comparison:

English: 0.066 - 0.068、Chinese: 0.0136、Random text: 0.0385，而Random text則是假定為英文。首先，造成三者差異的主要原因在於分布不一樣，以Random text而言，我們是將26個字母設定為均勻分布，因此透過公式可以發現其值大約就是1/26，就是0.0385，至於英文則是因為其部分字母的頻率較高，如E、A、O、I...等頻率較高，因此會有提升IC value的情況。至於中文文本，雖然在文獻中找不太到相關數值，不過我們利用政治大學統計的口語語料庫的數據資料[1]進行計算得其值約為**23,756,786,106/322,823,406**，約為0.0136，然而我們需要知道的是實際文本IC值可能會更低，因為口語語料庫可能是使用較常見的字。但我們仍可知道，中文的IC value會如此低的原因是因為中文字大約至少10萬字，而雖然有部分字如:的、是...使用頻率較頻繁，但因為分母過大，導致其IC value是相對偏低的。另外，若是使用中文的隨機文本，則值將會是約10^{-5}左右，因此由數學角度而言，只要是語言中有常見的字，則其IC value就會是高於使用該語言的隨機文本的IC value。

2.4 Decryption Challenge:

a. 解密結果:

```
PS C:\Users\蘇柏叡\Desktop\313553024\problem2> python main.py
Index of Coincidence (IC) = 0.0681

Decrypted Plaintext:
THEREARETWOWAYSOFCONSTRUCTINGASOFTWAREDESIGNONEWAYISTOMAKEITSOSIMPLETHATTHEREAREOBVIOUSLYNODEFICIENCIESANDTHEOTHE
RWAYISTOMAKEITSOCOMPLICATEDTHATTHEREARENOOBVIOUSDEFICIENCIESTHEFIRSTMETHODISFARMOREDIFFICULT
```

經過空格分開並且查找原文後可以發現是下面這段話(標點符號為自己加上去的):

THERE ARE TWO WAYS OF CONSTRUCTING SOFTWARE:
ONE IS TO MAKE IT SO SIMPLE THAT THERE ARE OBVIOUSLY NO DEFICIENCIES;
THE OTHER IS TO MAKE IT SO COMPLEX THAT THERE ARE NO OBVIOUS DEFICIENCIES.

b. 解題流程:

i. 先濾除空格並且計算字母頻率，接著開始計算IC Value
ii. 求得IC Value後因為0.068我們將其判定為英文語言
iii. 接著使用Caesar cipher的概念進行解密並且設立shift金鑰達到推估向前、向後位移量，並且輸出後檢視輸出的文本是否有符合英文文本的文法、單字合理性...等。
iv. 經嘗試後，發現當shift = 3（向前位移3)時，輸出最為正確。

Problem 3.

Description (4.5 pts)

You are given a ciphertext (download from e3) encrypted using the **Vigenère Cipher** with an unknown key. Your task is to develop a program in **Python** or **Go** to break the encryption automatically.

A frequency table is provided. You can get freqMap in the "Reference" page (Last page).

Requirements

1. **Key Length Estimation:**
   - Use Index of Coincidence (IC) analysis to determine the most probable key length.

   (hint 1: The IC value of English is approximately 0.068.)
   (hint 2: The key length is between 1 and 8, i.e., $1 <$ key length < 8.)
2. **Key Recovery:**
   - Apply the Chi-Square test to identify the key using English letter frequency analysis.
   - Use the following Chi-Square formula:

$$X^2 = \sum \frac{(O - E)^2}{E}$$

   (hint: Try all possible key candidates and select the one with the lowest Chi-Square value.)
3. **Decrypt the ciphertext:** Use the recovered key to decrypt the text and specify which article the plaintext is.
   (You don't need to submit or take the whole screenshot of the plaintext, just give the title of the target article.)

a. 解密結果:

```
PS C:\Users\蘇柏叡\Desktop\313553024\problem3> python main.py
估計的金鑰長度: 8
恢復的金鑰 (tentative): NYCUNYCU
```

解密後的文章我們發現是Declaration of Independence: A Transcription，也就是美國獨立宣言文章。

b. 解題流程:

首先，已知key length在1-8之間，因此我們需要先計算各種key length情況下的IC Value，並選擇IC Value最接近0.068的密鑰長度，此例為8(**但其實是4，因為NYCUNYCU是重複的，所以可以直接取值4**)，接著我們將密文視為由多個長度為8(或4)的子序列組成。

接著，我們會利用卡方檢定並且會先使用了某個字母作為密鑰字母，將密文轉換為明文後再去計算轉換後文本的字母頻率分佈(且和字母頻率表做比較)，**此步驟會選擇卡方值最小的位移量，作為該子序列的密鑰字母，並且獲得恢復的金鑰。**

最後，我們使用恢復的金鑰對密文進行Vigenère解密。這會將八碼的密鑰一直重複直到其長度與密文相同。其中會對密文中的每個字母，根據密鑰中的對應字母，進行解密操作，再將解密後的字母組合成明文，直到所有密文被解密成功。而最終我們獲得了明文可以發現其內容就是美國獨立宣言。

Reference

[1] https://spokentaiwanmandarin.nccu.edu.tw/character-frequency.html