密碼工程 Quiz5

學號:112550090 姓名:曾士珍

Problem1

a) Write a Python/C++ program to generate 1M bytes of cryptographically secure random numbers.

匯入 `secrets` 模組，使用 `secrets.token_bytes()` 函數生成 1 百萬位元組的密碼安全隨機資料，並將其寫入檔案

```python
import secrets
with open("random.bin", "wb") as file:
    file.write(secrets.token_bytes(1048576))
```

b) Run the NIST SP 800-22 statistical test on your 1M bytes of binary cryptographically secure random numbers and analyze the test results to identify any deviations from the expected statistical properties of random numbers.

檢測方式說明：

1.Frequency: Checks whether the frequency of occurrence of each bit or symbol in the data is uniform, which is useful for detecting obvious biases in encryption algorithms.

2.Block Frequency: Divides the data into multiple blocks and then performs frequency tests on each block to ensure the uniformity of the entire data stream.

3.Cumulative Sums: Computes partial sums or cumulative sum sequences of the data and then checks whether these sums fall within the expected range.

4.Runs: Detects the number of consecutive identical or different bits in the data to identify patterns.

5.Longest Run: Finds the longest consecutive run in the data and checks whether its length falls within the expected range.

6.Rank: Applies matrix transformations to the data and calculates its rank to detect

linear correlations in the data.

7.FFT: Applies the fast Fourier transform to analyze the spectral characteristics of the data to detect any regularities or patterns.

8.Non-Overlapping Template: Searches for specific templates or sequences in the data to assess the randomness of the data.

9.Overlapping Template: Similar to non-overlapping template testing, but allows templates to overlap, providing increased detection sensitivity.

10.Universal: A comprehensive testing method that combines multiple statistical tests to comprehensively evaluate the randomness of the data.

11.Approximate Entropy: Calculates the similarity of patterns of consecutive bits in the data to assess the degree of disorder in the data.

12.Random Excursions: Analyzes random excursions (sequences consisting of +1 and -1 symbols) in the data to detect any non-random behavior.

13.Random Excursions Variant: Similar to random excursions testing but uses different statistical methods.

14.Serial: Detects the correlation between consecutive bits or symbols in the data to assess its randomness.

15.Linear Complexity: Calculates the linear complexity of the data to assess its randomness and complexity.

c) Extra credit: Find out a non-cryptographically secure random number generator, such as random(), to demonstrate its lack of safety. Then, propose modifications to enhance its security to generate cryptographically secure random numbers that meet the highest standards of security and reliability.

使用 random 產生亂數

```
import random
random_bytes = bytearray(random.randint(0, 255) for _ in range
    (1048576))
with open("random.bin", "wb") as file:
    file.write(bytes(random_bytes))
```

檢測結果(有六筆不通過)

```
1  0  0  0  0  0  0  0  0  0    ----    0/1    NonOverlappingTemplate
1  0  0  0  0  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  1  0  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
1  0  0  0  0  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
1  0  0  0  0  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  0  0  1  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  1  0  0  0    ----    1/1    NonOverlappingTemplate
1  0  0  0  0  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  1  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  0  1  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  0  1  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  0  0  1  0    ----    1/1    NonOverlappingTemplate
0  0  1  0  0  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  1  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  0  1  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  0  1  0  0  0  0    ----    1/1    NonOverlappingTemplate
0  0  0  0  1  0  0  0  0  0    ----    1/1    NonOverlappingTemplate
1  0  0  0  0  0  0  0  0  0    ----    0/1    NonOverlappingTemplate

 1  0  0  0  0  0  0  0  0  0    ----    0/1    RandomExcursionsVariant
 1  0  0  0  0  0  0  0  0  0    ----    0/1    RandomExcursionsVariant
 1  0  0  0  0  0  0  0  0  0    ----    0/1    RandomExcursionsVariant
```

使用 random 模組來生成密碼安全性的隨機數可能會有一些限制，因為它主要是為非加密用途而設計的。改善的方法可以結合多個不同的隨機性來源，改用 random 模組和操作系統提供的 os.urandom() 函數相加來產生結果。

使用 os.urandom() + random

```python
import random
import os
random_bytes = bytearray(random.randint(0, 255) for _ in range
    (1048576))
# Add randomness from OS sources
random_bytes += bytearray(os.urandom(1048576))
print(random_bytes)
with open("improve.bin", "wb") as file:
    file.write(bytes(random_bytes))
```

檢測結果(全部通過)

詳見附檔 improve_random_finalAnalysisReport.txt