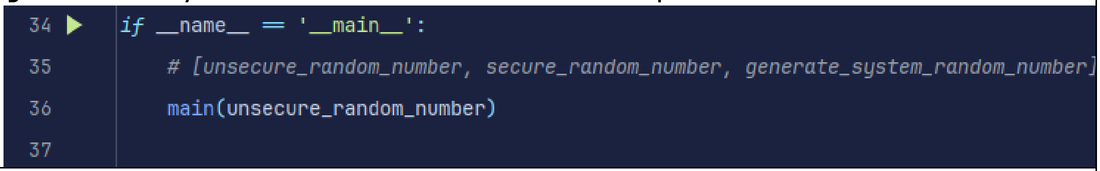
Crypto Engineering Quiz 5

1.

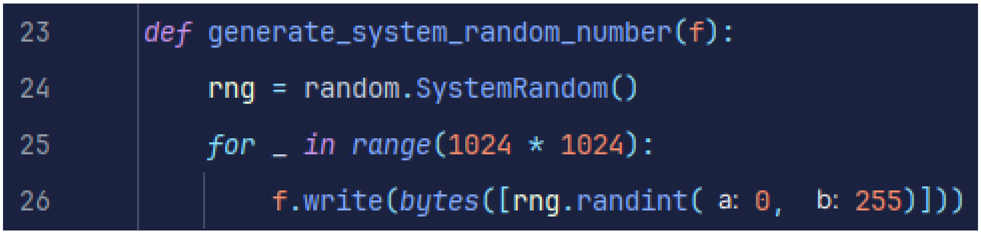
to run the code: **python RNG.py**
required library: random, pycrypto
to install: **pip install pycryptodome**

to change the generator:
plug in unsecure_random_number, secure_random_number, or generate_system_random_number in the parameter.



```
if __name__ == '__main__':
    # [unsecure_random_number, secure_random_number, generate_system_random_number]
    main(unsecure_random_number)
```

a.
The program generates one byte from system random per iteration, and run for 1024 * 1024 iterations.



```
def generate_system_random_number(f):
    rng = random.SystemRandom()
    for _ in range(1024 * 1024):
        f.write(bytes([rng.randint( a: 0,  b: 255)]))
```

Since system random is cryptographically secure, so it can pass the NIST STS test.

b.
NIST STS is consisted of 15 different tests, all the tests would calculate a score P-value and test if $P \geq 0.01$, then the sequence is accepted as a random.

1. Frequency test
Test if the total occurrence of 0s and 1s is close enough.

2. Frequency Test within a Block
Test if the occurrence of 0s and 1s in a M-bit block is enough close to $\left\lfloor \frac{M}{2} \right\rfloor$.

3. Runs Test
Test if the consecutive runs of identical bits in a sequence is close enough to $\frac{1}{2}$.

4. Test for the Longest Run of Ones in a Block
Test if the length of longest consecutive 1s in a M-bit block is close enough to some designated value.

5. Binary Matrix Rank Test
Split the sequence to some M*Q matrices and test if the ranks

of the matrices match the specific value.

6. Discrete Fourier Transform (Spectral) Test
Apply Discrete Fourier Transform to test if some periodic features appear frequently in the sequence.

7. Non-overlapping Template Matching Test
Test if some pre-defined non-periodic **non-overlapping** pattern occurs in the sequence too much.

8. Overlapping Template Matching Test
Test if some pre-defined non-periodic **overlapping** pattern occurs in the sequence too much.

9. Maurer’s “Universal Statistical” Test
Test if some block in the sequence occurs frequently and thus can be compressed without loss of information. If so, then the sequence is considered non-random.

10.Linear Complexity Test
Test if the LFSR that can produce the sequence is long enough and thus complex enough to be considered as random.

11.Serial Test
Test if all possible overlapping patterns’ occurrence has near uniform distribution, if so, then it can be considered random.

12.Approximate Entropy Test
Test if the occurrence of pattern of two consecutive length is uniform enough to be considered as random.

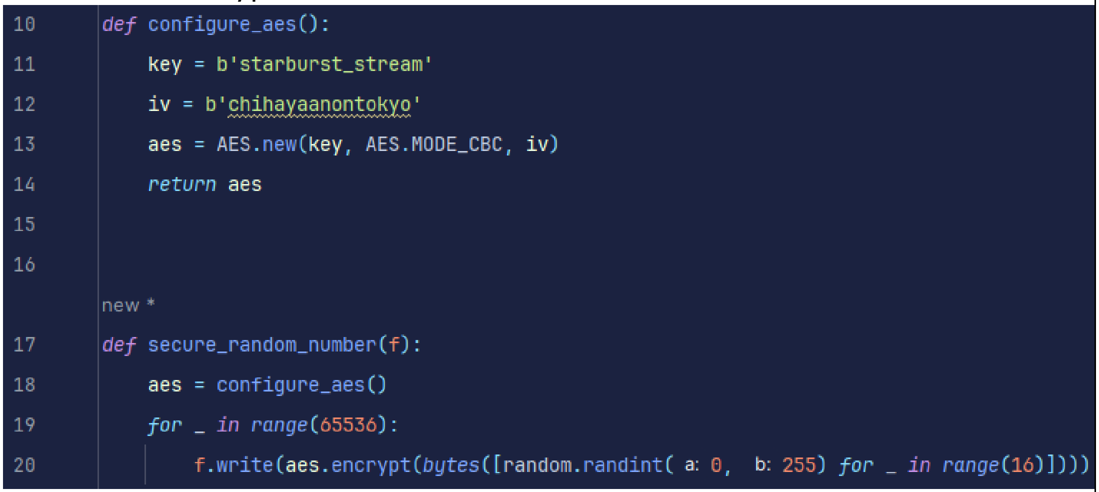
13.Cumulative Sums Test
Test if the Cumulative Sum of different length is close to the expected value of a random sequence.

14.Random Excursions Test
Test if state in $(\pm1, \pm2, \pm3, \pm4)$ during the Cumulative Sum random walk is visited too much that deviated from the expected value of random sequence.

15.Random Excursions Variant Test
Like the previous test, but it tests the states in $\pm[1,9]$.

c.
Take random randint as an example, it is built on the original random() function, and it is based on Mersenne Twister generator. So, it has a period of $2^{19937} - 1$ and thus

not cryptographically secure.
For modification, I would process the generated random number with AES encryption to meet the NIST STS tests.



```
def configure_aes():
    key = b'starburst_stream'
    iv = b'chihayaanontokyo'
    aes = AES.new(key, AES.MODE_CBC, iv)
    return aes


new *
def secure_random_number(f):
    aes = configure_aes()
    for _ in range(65536):
        f.write(aes.encrypt(bytes([random.randint( a: 0,  b: 255) for _ in range(16)])))
```

It has passed the tests and the binary file, and report is in the bonus folder.