Lab 1 Seguridad y Criptografía (updated 2022-06-23)

Course code:	MC1003
Due:	June 27, 2022 at 23:59 (Central Standard Time)
Total points:	25 points

1. **Breaking a XOR Shift Cipher** *(3 points)*

   During the lecture you learned how to break the XOR shift cipher in theory. In this exercise you will break such an encrypted message yourself.

   Because raw bytes are difficult to put in a PDF like this one, we encoded the bytes using `base64` encoding. Decoding the `base64` string to raw bytes can be done using your favorite programming language or CyberChef. An example of how to decode a `base64` string is shown in listing 1.

   Listing 1: Decode a `base64` string into bytes using Python3.

   ```
   import base64

   ct = base64.b64decode(base64_ciphertext)
   ```

   The `base64` encoded ciphertext for this exercise is shown in listing 2.

   Listing 2: The `base64` encoded ciphertext.

   ```
   KxRIRjMSEgMUFQkIRhIOA0YKBxEfAxRGEQcVRgdGCwcIRgkARgdGFBMBAQMCRgUJEwgS
   AwgHCAUDRhIOBxJGEQcVRggDEAMURgoPAQ4SAwJGBB9GB0YVCw8KA11GBQkKAkpGFQUH
   CBIfRgcIAkYDCwQHFBQHFRUDAkYPCEYCDxUFCRMUFQNdRgQHBQ0RBxQCRg8IRhUDCBIP
   CwMIEl1GCgMHCEpGCgkIAUpGAhMVEh9KRgIUAwcUH0YHCAJGHwMSRhUJCwM0CRFGCgkQ
   BwQKA0g=
   ```

   Write a program that creates 255 candidate plaintexts by decrypting the ciphertext using every possible key. Compute the statistical distance between a candidate plaintext and the statistical distribution of an English plaintext to automatically identify the plaintext.

   Hand in your program, the recovered plaintext and the decryption key.

2. **Breaking the XOR Vigenère Cipher** *(7 points)*

   During the lecture you learned how to break the XOR Vigenère cipher in theory. In this exercise you will break such an encrypted message yourself.

   The `base64` encoded ciphertext for this exercise is shown in listing 3.

   Listing 3: The `base64` encoded ciphertext.

   ```
   EjxYTxMKdwYaSBwcGTwfCBxHOwsGLRYdGwsBZSw8GhsaCncIEwdDVQM6AQ1OEzwJEWwJ
   FRoBUjIgIFQOFRg+B1QLHRoEJRoHCUchGEUlC1QzBBY2IypTHFIUPgcQRE8XBjoGDgYT
   dAcQOEUVVAMeKi43VAAUWTsIGgwDGgYxAEkZDiAARToEFxULBmUtPAAcXlk2BRhIAxob
   PhoHCUcyBxc7BAYQRQYqYTcdHAIWJAAaD08aEnUSSQoSORhFOwoGAA1SMTY8VAsdFTsI
   BhtPFBoxUwhODzUEA2BFEhsXUiMoNQAWUg0/BgEbDhsQe1MnARB0CQtsBB0GFR03NXMZ
   GgENdwYXCxoFDXUSSRgGJxxFIAoAVAoUZS0yGgteWSQGVAsOGxo6B0kdEzUGAWwXHRMN
   BmUoPVQOUjo+HQ1PHFUHPRwZHg46D0UoDAcAFxsmNWhUDQcNdw8dBg4ZGCxTCE4FPQ9F
   ```

```
OBcVFxFSMiAgVA0dDDABAERPFBoxUxsHADwcRSULVBYEES5hPBJPBhA5EFQlDgcdNB1O
HUc2CQYnRQ0VFxZkYRoaHAYYOR0YEUNVNzwHEE4vNQQJbBIVB0UUMC0/VAAUWTYZBAQG
FhU7BxpOATsaRSoJDR0LFWUDIRUBBhY5STwBAxkHclMPBxUnHEUtDAYXFxMjNX1UOx1Z
EAgQGw0MUyZTAwEeeEgEIQoaExYGZTU7FRtSGyIHFwBPAhUmUyEPFTsEAWwxHBsIAjYu
PVhPExd3BhgMTzoGMhIHBx01HAwjC1QYBBZpYSQcAFIONhpUAwEaAztTCBwIIQYBbBEb
AwtSJDJzFU8RETYZVB8HGlQ2HBwCA3QMCmwEFhsQBmUgPQ0bGhA5DlQLDhkYPB0OTgE7
GkUuFxUdCwFrYRIHTxMXdwgdGh8aBiFTAB1HOgcRbAkVHQFSKjQnVAYcWTZJEAkWWVQd
EhsBCzBIAiMRVBYQATxhJB0bGlknCB0MTxQCPBIdARUnSAQiAVQHCh0rYSQVHFIJPgUb
HAYbE3USSQ0VNQ4RbBIdAA0dMDVzFQYWQncIGgxPGxshUwYACy1IJz4EGgAKHGUJOhgD
AVkxBhgDHFlUNwYdTgg4DEUtEx0VER03Mn9UHBMOdwAaSCcUBjofDUJHNUhHLgwGEEgf
JC9xVAAUWTkGVBsCFBg5UwgMDjgBETVLVDULFmUyPFQbGxcuSTkJHRwVO1QaTkUiARYl
ChpWRQUkMnMVTxQYNB1PSAUABByFTCB1HdgoMK0UTHRceZ2EfAQwLXiRJLgcATlQ0HQ1C
RzUbRTsMABxFEyktcxYGFVkUAAARTxQSMxIAHBR4SAQiRT0aBAciNCEVGxsWOUkHAAAA
GDFTGhoGJhxFJRFUGwMUa2EdGxheWTgHVAkDGVQmBgoGRzUOAy0MBgdFCyo0cxUDBRgu
GlQOBhsQdRJJTBE9Gww4CgZUChRlKTwaAABbbEkVBgtVGztTHQYOJ0gCPgQaEEUWJDhz
Mw4WCjUQVAsAABgxHU4aRyAADCIOVBsDUiQvKhYAFgB3DxsaTwEcNAdJBwokBxc4BBoA
RQIqMidUDQcNdyQVGgYUGntTKAADeEgEP0UdAEUFKjQ/EE8dGjQcBkgGG1QUBg4bFCBE
RS0LDVQBEzxhJBsaHh13DRtETxQHdQcBDxN0ARZsBFQHBhoqLj9UGRMaNh0dBwFVGTod
HQZJ
```

(a) (1 point) Write one or more functions that take a `base64` encoded string, convert this into raw bytes, and split these raw bytes into chunks of length n.

(b) (1 point) Write a function to compute the Hamming distance between two chunks. Note that while the chunk length may vary, two chunks are always of the same size possibly except for the last chunk.

(c) (3 points) Recover the (most likely) key length that was used to encrypt the plaintext with by iterating over different key lengths and seeing which key length yields the lowest Hamming distance between all chunks.

(d) (2 points) Write a function to split the ciphertext in n parts, where each part is encrypted using the same XOR shift key. Use your solution from question 1 to recover the key for each of the ciphertext parts.

Hand in your program and the decryption key for the Vigenère cipher.

3. **Importance of Randomness** *(15 points)*

The Middle Square Weyl Sequence Random Number Generator (RNG) [Wid17], is claimed to be suitable for cryptographic purposes. Alice and Bob—just starting to learn about cryptography—decided to use this RNG to encrypt the messages they exchange. When Alice wants to send a message to Bob, she first converts her plaintext to bytes, to obtain a buffer of n bytes. Next, she generates $\lceil \frac{n}{4} \rceil$ random numbers using the Middle Square Weyl Sequence based on a key agreed with Bob. She converts these numbers to a *key stream* of bytes using big endian encoding. Now to encrypt the plaintext buffer, she XORs the buffer and the random bytes to obtain the ciphertext.

Eve intercepted a ciphertext, shown in Listing 4, from Alice to Bob and wants to decrypt this message. From the plaintext metadata sent along the encrypted message, Eve knows that the plaintext is a PNG image.

In this exercise, we are going to recover the image Alice sent to Bob.

Listing 4: The `base64` encoded ciphertext of `image.png` sent from Alice to Bob.

```
zMU5Qis3lghENvYoW35ex2CFqIxD7unCLN/r0xgW9snrSJ7m/Nm8SRLozUBqHh4VrN5a
Q5V/7EqHURHDZnDvZdUqe+mOhK0H4lKYR3T/alHLWWvVAFAsEaLgrBLwweaz7fOwfupg
43JRd7WU8Q1eTeWp0hrM/S7PFs65LI6TLk2UM/gFKdHQbauZRoQjYEEI+57bucXAtpYP
RXhZy3aiez0PAldZ9rBT/JrRHR9WumarpZ2rJePTIVI+cyW5ufzrRQaKEjSP+kng1TY9
HaXKkfdspYYfjpFF2Ue2s+LRrakKWQETZ3w5STDSGVPxaLECwu+DqESIQWCgqzP3zsS8
A232w++0xdeJaa0aWXV8G0LKSxFPHCuMgVLipQQSMf6plk07dBgieGtpvmeldp6nbVgB
/8l+8T9Q6aUb/KS+o7Od3AXNIoSWLtaXq/edp5UYjh9LWOuPQ5AWzSo1gJ058rxbUscI
ZI1y8qDdcHCFuWS9SACKNQpsHf/FmIMQVcq8CJwZlHqT3/AoeDOXBL7lteheYNiLQRZk
y+wj2QsnihKgXVYerWPChmRifYTCrdQ8qgWrJdzyDS69yv3Eyt3DsyHwi5H2ujtSh6eC
zLnmKp/bhmvi5fvaLIoG4tHGscPx52jdULTCk2xTwHUBMF/3kz4i14exvTpFpLfhyNBs
5NQ0wKOJU78+vjsBTZFmpYmfCDBAg3yccqG8I4FQ3mtQqLlbawynWM8E8iFGbNjqKzYq
bFjDE4Nk51SniHPsLqgH1EPC68VIMh8DEK5Yg+DZ60SMQQTRenRJJVSXUKUZ9is+elRI
hD11q9XNFScwt4sMOt5c89TUbqNMsl/5evxq7V3E5jw7ARhClymCIgu6SMdesf8WdhQN
lxyAkwuQ339c2zbf9zFt0h2YDfvH/1UPJ/p1BH54qqB3DzUpbNG5TsO6nmpspyHSvBI+
8WPYHkccIK7N3zNtgc2rxnOJG6nTYprOQ85rgeb9YjSXIbimgZRE2NSBH9tLounSoXh7
G4LSRoMIF1oPKaChRJoOBqdN5FbT8Yf95UZgpDuz42NMpcD6e/xHE/69RdUkTTxpn0rn
GgYcfR0p4xoWj8OK1gYlQr2rY2R7gK8VfdlY3ip4y67h0Tz2iD0dmk28vNXviQHqnLTZ
kUG9TLnF43g0dcGUylt9lp6sV4uA6sr1WkAWdnWR3zUKBaHzxt0BLAl9WFA6cqhBoSRJ
BvACANRp+fd3Sl5sWzn8K9R4YuanwekU6yBbU6aL5E1poNgTFJgl+/YlIGvFzuMKGZOk
dIu25gD8c0oO6CbhMp9x7TF7w1yNUY2cCvPOfKbyl4hK3dyXwGmAlad/ooNEt0VA64zo
Bo8+8/vGyBV6JMa710PdtLIaHrEvwEIXCX5rZjAukFCpOgTpgrp2TCzSJIwlIhxXvfNd
NcqLnDzDWaUur8vBzsw87kynVAuYiHTmFQhyCmbLxWhDRHkGeK2y6qxybjplmFhnCNsq
foO4VbuT0nfCVkfavmUqOoE/xtar48xZmqBECxGPWwX0uFbeUxSv5trNMlZKYAql0rBs
eHE4fknpEUJG34oYjj5z69LrfSIh7gVh1D+pUfPiaK0duoJSDnC0lhJ3hHyJa6WcYhzF
90tcjZp+4a2HDPMdzzHobX7uWoBjPOh6TpXlLueaCFHPKVylkl+G0RdJTOsMH9nqOUYl
RYvx4H2IDnrjszkHLfWXLwEPbQPrRAFdgKb9CVd0biRUOkx+2DQgVT1A50nkCFT0VLSp
onOkM1K83vP/Ysrfu1sTEAdLI1PdtULLMzo+3iduXRdhDIk3JNRSVASrm0azX1tRtZMb
1wl7zEcFYWGR4Kd+VY3jZCqaYJytEPgfFIIcJ2twop7y/a461jCNbG47uPl1zGKXNRG5
0YnBKZST3KoMCBJYmD50bfghagqKdxkLp1jQIOMNeswJrNgkEk3Hqrtu5n1UEAM2qBE0
43wSSa8gv4Q55rUICDborJgwyNmXwt1HliX8r/gKWMML3BUarKON+Q7s24IXYQq73vTl
cvp4oaZfarlo/dA8ni/a6oo6tR81xKLA/KMey4yvrsJNd772k9fp9wgxHVEUKSsUQzMk
RqkDajDYlI1EMeMjU6C8f9vzNfjQqp6SK9feeLZOZCczW1nOgtWU+Nq5j7ah/R/Ox/L0
soJ+g3ij+TMOGQQGnUMISxl/uTs2uNwVNnTSWE7raDzjFEV+VdTakHYIUCPl8qskc+x6
5UuGcSuTPtZbCEd2tZnUGSqXT32If40BnNprUKHW6N0Us4XeXWb5hRUaHwxj/86291eG
9WTIRdZ4SPaUT5fKWLbKjDQzX842RHaWAvYoN/8EUVqLD6uNLAop0YcHm5sh7VmrvBquo
oP3yZhDEHd67KkNeM8bBXPcehXjaMZ1620t5kvkHFvkLeDNDQQUcvIS+2mHT16LPZdyD
SzYVbvdTQeHYXRgbEFIMEMyhOkkUWY1IhzZACuq7F+r85yjVBFxC40TGdHtAlq7pJxGR
rjggo4hrKBSLza99W6Le70yPgBQV0Tdm6GziIXeQJ2CSUiP3PFlly/MEvjyOuA==
```

(a) (10 points) Before we start trying to decrypt Alice's message, we will refute the claim that the Middle Square Weyl Sequence Random Number Generator generates cryptographically secure random numbers by breaking the RNG.
Predict the next three numbers given the sequence of numbers

$$487488736, 1142881612, 121804679, 3766260381, 3936115597,$$
$$3533079714, 514960507, 3453936634, 3546284790, \text{ and } 566317578,$$

and given that $s =$ `0xb5ad4eceda1ce2a9` was used as initial seed.
Hint: Have a look at `https://crypto.stackexchange.com/questions/62750/` if you have no idea on how to approach this.

(b) (1 point) To recover the plaintext from the ciphertext we will need a *crib*, a part of the plaintext that we can predict. What are typically the first 16 bytes of a PNG image file?

(c) (3 points) Recover the plaintext image by writing a computer program/script. What text is displayed in the image?

Remember to hand in your program as well.

(d) (1 point) The encryption of the image was done in a way that is very similar to the one-time pad. Still, we were able to break this cryptosystem. What is the difference between the broken cryptosystem and the one-time pad?

References

[Wid17] Bernard Widynski. *Middle Square Weyl Sequence RNG*. Tech. rep. Feb. 2020 (Apr. 2, 2017). arXiv: `1704.00358 [cs.CR]`.