Lab 1 Seguridad y Criptografía

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.

```
EKE83+2vsUQAzKDmgvut1JYPS8VKfIYhKHAJVmW9imY6r4XcZgqiCGQp3dCkPxttKjN7
FV7FLCFRd8XJXcqgAuxuilQZg7RfX7qjcKfssiB6oeK8zHkk85fX45HvRmEYf2vn1XOQ
azW4zwzcLzd1j3rHOEiV71ibDC4/DDgnayeb7bGeInTDpK6Ml+itpMg5CsAGxDTOSAHb
DSRxrRZ6Lb8E6+2AzkL461tq3vmnHcKkabEfwUgQWvOwewpAh1AeBP8n04+R+mmugR6E
NIosXZGSfYISO0nwf/nd3lBEBEQYS51hwrJ05GijNILpruWKyyVeN21jZhhkRDOndX20
qo3BzHJiDEIIUlWhJGgqyS0mpKuTBFxg0L6GqNa6sLxOeM3XJa1TxK64SNMNIdy9NfT4
7eTTaE9nUh6Isk5enfoSRcwdpxUxLMSNRXeWqTRorrhl5T1f8OrlMWoHBMsKXXgG9Vnv
Y44nayOlRZ6ZCPuQ+SvzYmSMo6klRRLp7RmrmaCgTCIifKr9ziRadwY4TGniHw60ldeL
kGl4ZFSekukjuk+Y1eOlyAeS5+x84eP/L5hYr/rCVyFOmHEW3nln9+MfoCV8rVA+FJ+0
xJlTv9qsatLKz6E2ss6vYQCBdsWCVHWJEkAkJ3w9iNyGxD1gqM1gkt9FslzrOfbZzcOL
Eqzm9bJ1BTlhe+gkhylDDrBo0JEyJieRcUb3IpFSqQZcDDT4ifslf9XrF15yoBJ1vRmb
RVYNF0mXuGe926yJ9EjliWRJMmuSQ9bTtL3mx6TnZh6uMjOKtx7DaUcaQ5QQDkbGNLSi
GVCGd2EUykkayeRv81Am3MN48DyDfBNcSj19orrvA+UbDFQHAxxZ7Ntt8tk/8ST8f5Mz
ChckvILBaFiBZ8oTdvpkhwa5/+KBKIPUg84x5YRv1L/RVIt+zS1DfjdJCFWFlT9Sug4Q
8AgpxaU5B2VdMe1zAgxH7D8Q7+7WZxd+55XxqQDVk5PUXTuZdFdm2c5FW/illkrbzYnQ
FTroTCmgrlENQtYURgTriaX2V2rVdKjGcXdB2uByD6XyzKxRAaswncBuLc3HDAT185UZ
SI3Y4N8B0Adv2VKr7aPirFw4iYQS2D2Nb2VPnY+MSVjFzvo8m/CDlrBvJUQMph+8uoRf
8FrfiQ5kSyUqyxdYdA+djiTNPVZUczUBgT8H3dSsPLSxGrAETrOuJw92UExmn13bg/9T
RscsmVvo0IPmW7XmoKZF1BYLTQSw8e0E8uHOSN0hUWuftj2GRGNgyFfBi2dP9UNxpBMg
fpVOZa6URVI70GQLERY/6OVKlrXqQ8TZEqKP1rmnv3birOFNOgM5IXejDQCY7OQV/rEH
p3S8FsH50VmU4PRzVxbvX1RQrO13oUcbGQo+5x6jGpbMCcyEShZfelO5cT1bKt+bCNNy
oDTrhYp17WWeRVEzsTzppv5Fr8gi6HnZQ7A4P4TneD349cWpcJMwSnwOScVN2YTQkYXH
udEqJ6MLRjUusZzDxR2qhO7C4LBtLPq9/b9Ghp5A55dt+sSIU/WrQnqXqM1T2/ekW5aj
eumpcSsfrI1W16T/iwmQyxgKFasFJmy5LBGUYGpgQt6SWYxvS10PJFzVz7Vo0MTq7aD8
foN2/DVTSipBwLEjQDEpFTQlRdbCNR4Qpd5xZufLwPznc5nD56phy3uHMOUhGnvTb1/E
HSsB/k26WQujKObXsfY1799SWdP5kU9ou43KOUlJ1UncgjW4bEnnetyoGp/d5Y7KbzYG
1w/HOLbyMQQKnNGcSWcCW4HZIP13ZigPD2hs5p93HxEcCOuZC2YlGLBMyKbJRVCbf2ci
e2HTxKUZ2zFeSD8C4/y6sdUbv5roGgLnuE8/KaJiGGdiYJSpiXA60A6x4Lk7CpgcSnhD
HlX0/PqgWpA5hld+aVCPV0kHMA2jFd1j3HTpV5AqLBUXMkp0q83vDHLm36SnYi+DdsEY
b2iOv63HOfJvqb8JOI98Q9LVnUXGAqbp1/Ek34XsBQ0ePaWNv/+Z4ovOI8psr2T0aLzs
CyRHDeFyizKVYSOig844Z0zfl4TzwxP0X46HwLjVUL2G92DvgphNJiA+yBh2AX2Fs/9c
vjMdQHDStb1cHctsWHH83VqeLJldX6KRmtZU3OWAizJvTC1R5KI55CDCKVKAh4J6lWYk
WA7QYUhGog6EkGxS8NluVnGa36J08STpAakvNIRAL4OjuLAqP8emwn2LtN1IbyDxKzOc
FFEYa6Q0gu0HDcTmRgg1el5GwTTdlGFdnAHmwW88pEe5j38Uno9/Pb2p4k+D8wZGYRSE
BH1Ky12ipKwMBYoGqImfBAr+A18irpPyZRzEMnKM2nac1U9qlL69D6Rmb1/kqRhUfWnc
IV4YgdyBFjvRw3o2mnWFCudDLX14z/vW2efJsI+DNPDIURng8ZhA4rmGA5NwEHl6QWLW
lOawQEfvzdJgDVscN7pFwG2pbQX+6ezxAF9jYT8rIaBzMu9z0sQZFNnK2K/xicqKL9Qc
acmRJdT8Z5S/6ToQr1aHFonOSV0tDfmlY+EKmjku/LcIECdfhhKPzO1s/3g9wA==
```

(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 when given the sequence of numbers

1579573374, 2047559810, 1573901180, 696743789, 3065020298,
2179045855, 2765740378, 687998646, 4048999215, 964510569.

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]`.