Author: Robert Lang & Mathis Michel

Lab 4: Cryptography

Part 1

Source Code

Frequency Analyzer

It reads the text of a file that can be specified via argument 1.
Then it will iterate over every char in the text once (text is made uppercase). In the iteration a entry is created in a Dictionary for each char that exists. The Dictionary holds the number of occurrences. If an entry in the Dictionary already exists the number is increased otherwise assigned to 1.
In addition to this the length of the input text is stored.
The PrintResults method loops over the sorted key value pairs of the dictionary(sorted descending by value). It will calculate the frequency using the text length and print the information to the console.
The GetCsvData formats the dictionary as csv. The csv will be outputted to an additional file in the containing folder of the input file.

VigenereCipher

It will ask the user to choose between decrypt and encrypt. The user is then asked to input a path to a file containing the content that should be processed. Then the user needs to enter the key.
Then the encrypted or decryption happens. it will loop over the provided content. For each char it will derive the new value followingly: Get the current char of the key([i - nonLetterCount] % [key length]). In case the current char of the input is no letter it will just append it to the result and increase the nonLetterCounter. Otherwise if the current char is in uppercase the offset will be 'A' otherwise 'a'. Then the current key char offset is calculated by using upper case or lower case respectively of it minus the offset. If we want to decrypt this key char offset is negated. With the equation the encrypted / decrypted char or both upper and lowercase can be derived:

Let:
offset=['a' or 'A']
c = [current char]
k = [current key char offset]

$(((((c + k) - \text{offset}) \% 26) + 26) \% 26) + \text{offset}$

The derived char is then appended to the result and written to the console.
Note: Please see the attached file 'toDecrypt.txt' for the encrypted text.

Histograms

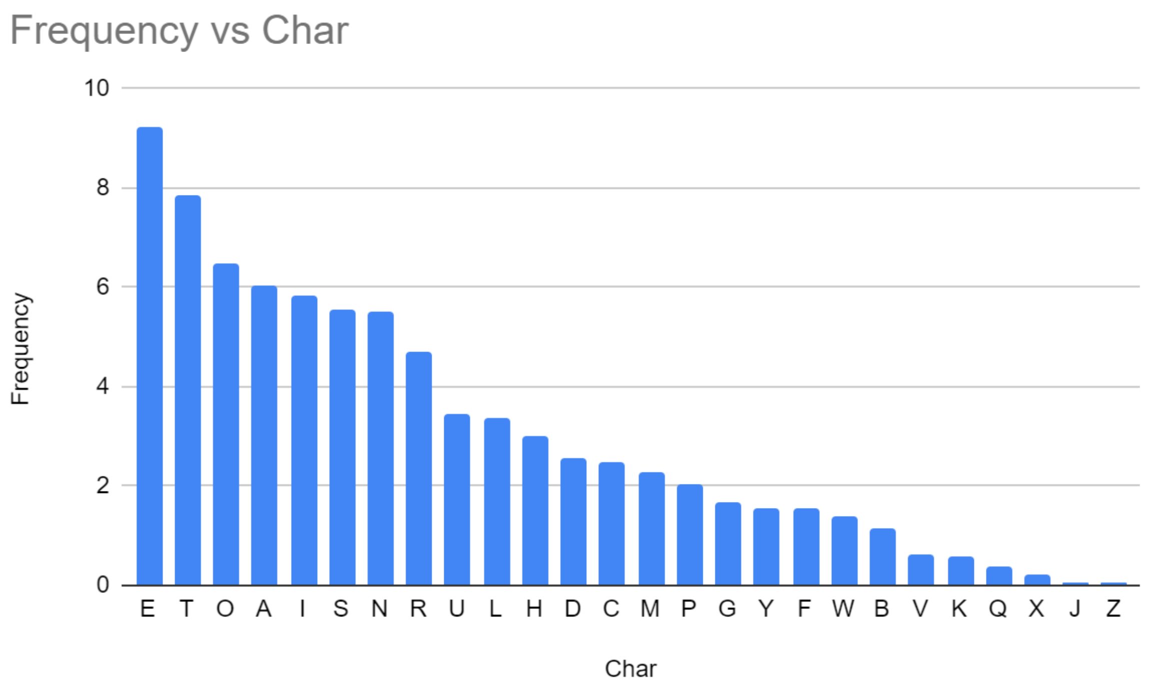


Figure 1: Plain Text

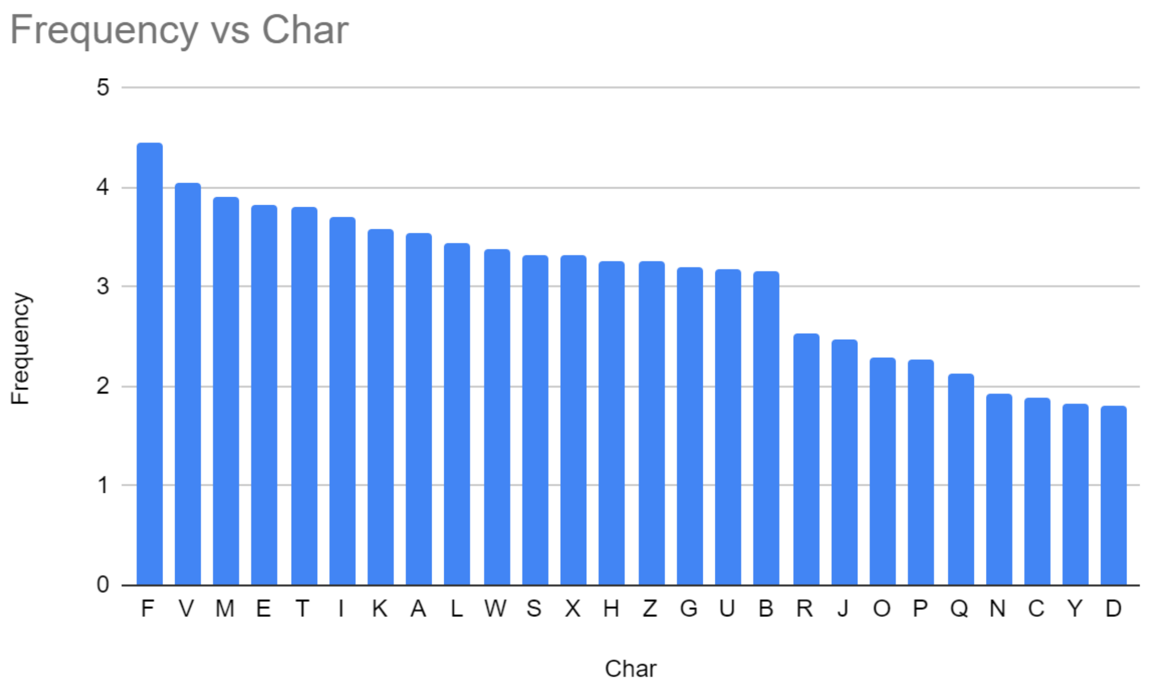


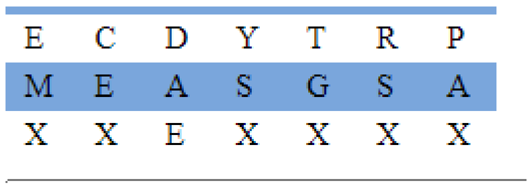
Figure 2: Vigenere Cipher

Comparing the two Charts

You can see a more natural distribution in the chart of the plain text. Since “E” is the most common letter used in the English language it’s obvious that it is our most frequent letter. After we encrypt the plain text you can see a wider spread between the distribution of the letters. It is not that obvious which of them the letter “E” is. You might guess it's “F” now but since it’s not a major difference you can’t tell for sure. Now it helps, if you can analyze the encrypted text with the key length. This way you can most likely reverse the shift of the letters and figure out which letter it is.

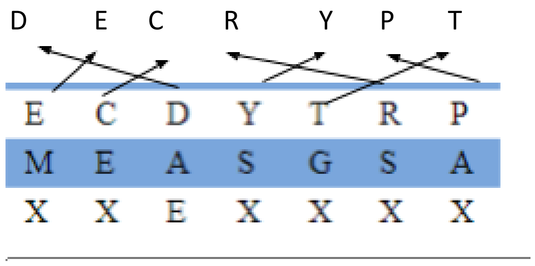
Part 2

The text has 924 characters and given the range >6 and <11, 7 is the only number that can be used to divide 924. So the block length may be 7. Let’s try to find a word in this block table:
The last 2 blocks are very easy to infer:



E	C	D	Y	T	R	P
M	E	A	S	G	S	A
X	X	E	X	X	X	X

Figure 3: Last 2 blocks
This could mean ‘decrypt a message’.
In order to make it to this words the column order needs to be adjusted:



D	E	C	R	Y	P	T
E	C	D	Y	T	R	P
M	E	A	S	G	S	A
X	X	E	X	X	X	X

Figure 4: Adjustment
The new column order is 2, 0, 1, 5, 3, 6, 4!

The resulting text is:
THEENIGMAMACHINEISANENCRYPTIONDEVICEDEVELOPEDANDUSEDTOPROTECTCOMMERCIALDIPLO
MATICANDMILITARYCOMMUNICATIONITWASEMPLOYEDEXTENSIVELYBYNAZIGERMANYDURINGTHES
ECONDWORLDWARINALLBRANCHESOFTHEGERMANMILITARYENIGMAHASANELECTROMECHANICALR
OTORMECHANISMTHATSCRAMBLESTHETWENTYSIXLETTERSOFTHEALPHABETINTYPICALUSEONEPERS
ONENTERSTEXTONTHEENIGMAKEYBOARDANDANOTHERPERSONWRITESDOWNWHICHOFTHETWENT
YSIXLIGHTSABOVETHEKEYBOARDLIGHTSUPATEACHKEYPRESSIFPLAINTEXTISENTEREDTHELITUPLETTER
SARETHEENCODEDCIPHERTEXTENTERINGCIPHERTEXTTRANSFORMSITBACKINTOREADABLEPLAINTEXT
THEROTORMECHANISMCHANGESTHEELECTRICALCONNECTIONSBETWEENTHEKEYSANDTHELIGHTSWI
THEACHKEYPRESSTHESECURITYOFTHESYSTEMDEPENDSONASETOFMACHINESETTINGSTHATWEREGEN
ERALLYCHANGEDDAILYDURINGTHEWARBASEDONSECRETKEYLISTSDISTRIBUTEDINADVANCEANDONO
THERSETTINGSTHATWERECHANGEDFOREACHMESSAGETHERECEIVINGSTATIONHASTOKNOWANDUSE
THEEXACTSETTINGSEMPLOYEDBYTHETRANSMITTINGSTATIONTOSUCCESSFULLYDECRYPTAMESSAGEX
XXXXX

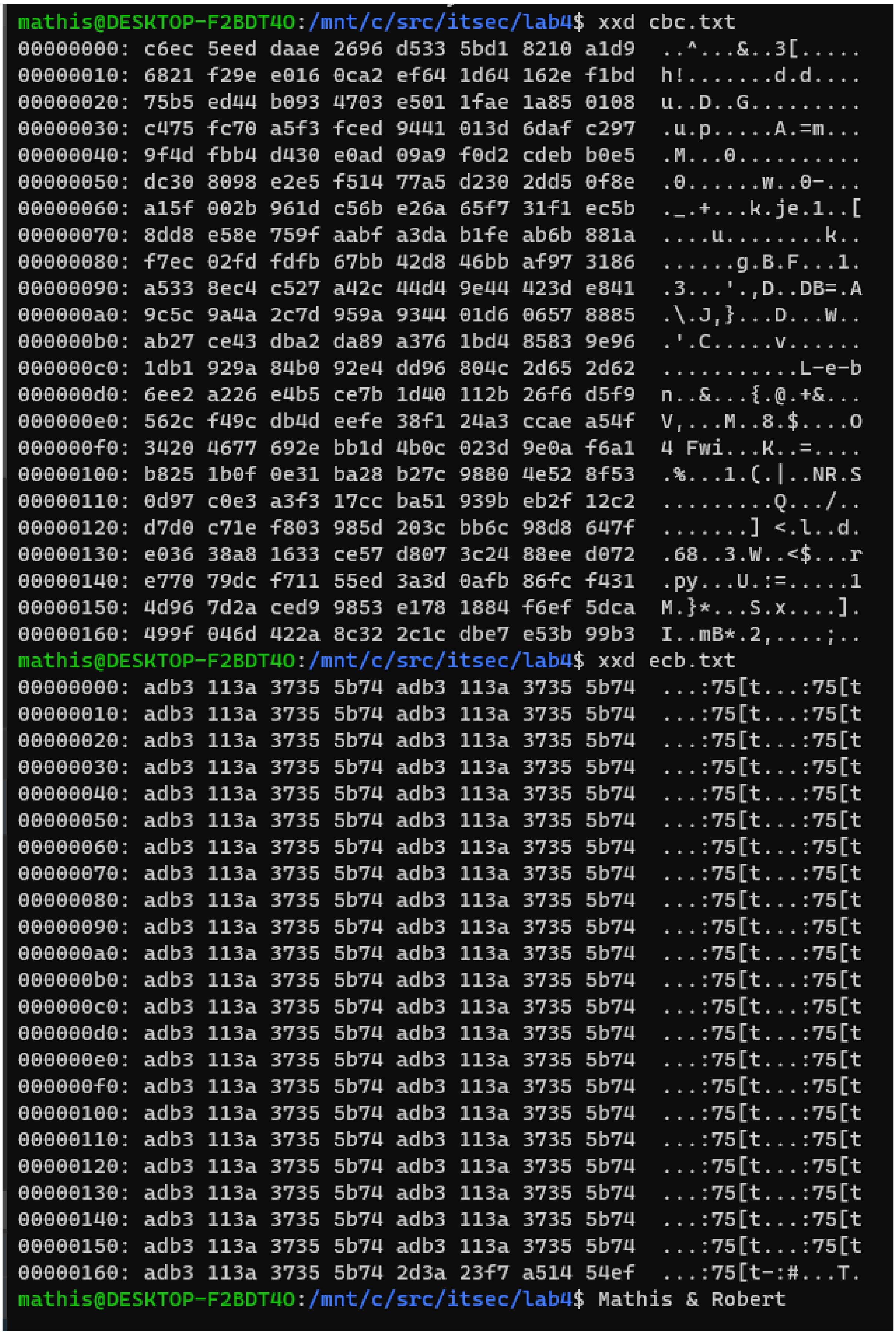
Part 3

AES

Plain:
Hallo wir sind Robert und Mathis:)!
Encrypted:
‰DNZ{:‹AEéà}¿ûŸVw"ðª#‰¾ÆYµa6%j}Á
UÃÌ6*4‡/
$µ¼à

DES

Plain:
ACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCU
RATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATE
ACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCU
RATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATE
ACCURATEACCURATEACCURATEACCURATEACCURATEACCURATEACCURATE
Encrypted in CBC mode:
Æì^íÚ®&–Õ3[Ñ,¡Ùh!òžà¢ïdd.ñ½uµíD°“Gå®…Äuüp¥óüí”A=m¯Â—ŸMû´Ô0à
©ðÒÍë°åÜ0€˜âåõw¥Ò0-ÕŽ¡_
Encrypted in ECB mode:
³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³
:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:
75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t³:75[t-:#÷¥Tï



```
mathis@DESKTOP-F2BDT4O:/mnt/c/src/itsec/lab4$ xxd cbc.txt
00000000: c6ec 5eed daae 2696 d533 5bd1 8210 a1d9  ..^...&..3[.....
00000010: 6821 f29e e016 0ca2 ef64 1d64 162e f1bd  h!.......d.d....
00000020: 75b5 ed44 b093 4703 e501 1fae 1a85 0108  u..D..G.........
00000030: c475 fc70 a5f3 fced 9441 013d 6daf c297  .u.p.....A.=m...
00000040: 9f4d fbb4 d430 e0ad 09a9 f0d2 cdeb b0e5  .M...0..........
00000050: dc30 8098 e2e5 f514 77a5 d230 2dd5 0f8e  .0......w..0-...
00000060: a15f 002b 961d c56b e26a 65f7 31f1 ec5b  ._.+...k.je.1..[
00000070: 8dd8 e58e 759f aabf a3da b1fe ab6b 881a  ....u........k..
00000080: f7ec 02fd fdfb 67bb 42d8 46bb af97 3186  ......g.B.F...1.
00000090: a533 8ec4 c527 a42c 44d4 9e44 423d e841  .3...'.,D..DB=.A
000000a0: 9c5c 9a4a 2c7d 959a 9344 01d6 0657 8885  .\.J,}...D...W..
000000b0: ab27 ce43 dba2 da89 a376 1bd4 8583 9e96  .'.C.....v......
000000c0: 1db1 929a 84b0 92e4 dd96 804c 2d65 2d62  ...........L-e-b
000000d0: 6ee2 a226 e4b5 ce7b 1d40 112b 26f6 d5f9  n..&...{.@.+&...
000000e0: 562c f49c db4d eefe 38f1 24a3 ccae a54f  V,...M..8.$....O
000000f0: 3420 4677 692e bb1d 4b0c 023d 9e0a f6a1  4 Fwi...K..=....
00000100: b825 1b0f 0e31 ba28 b27c 9880 4e52 8f53  .%...1.(.|..NR.S
00000110: 0d97 c0e3 a3f3 17cc ba51 939b eb2f 12c2  .........Q.../..
00000120: d7d0 c71e f803 985d 203c bb6c 98d8 647f  .......] <.l..d.
00000130: e036 38a8 1633 ce57 d807 3c24 88ee d072  .68..3.W..<$...r
00000140: e770 79dc f711 55ed 3a3d 0afb 86fc f431  .py...U.:=.....1
00000150: 4d96 7d2a ced9 9853 e178 1884 f6ef 5dca  M.}*...S.x....].
00000160: 499f 046d 422a 8c32 2c1c dbe7 e53b 99b3  I..mB*.2,....;..
mathis@DESKTOP-F2BDT4O:/mnt/c/src/itsec/lab4$ xxd ecb.txt
00000000: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000010: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000020: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000030: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000040: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000050: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000060: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000070: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000080: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000090: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
000000a0: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
000000b0: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
000000c0: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
000000d0: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
000000e0: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
000000f0: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000100: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000110: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000120: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000130: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000140: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000150: adb3 113a 3735 5b74 adb3 113a 3735 5b74  ...:75[t...:75[t
00000160: adb3 113a 3735 5b74 2d3a 23f7 a514 54ef  ...:75[t-:#...T.
mathis@DESKTOP-F2BDT4O:/mnt/c/src/itsec/lab4$ Mathis & Robert
```

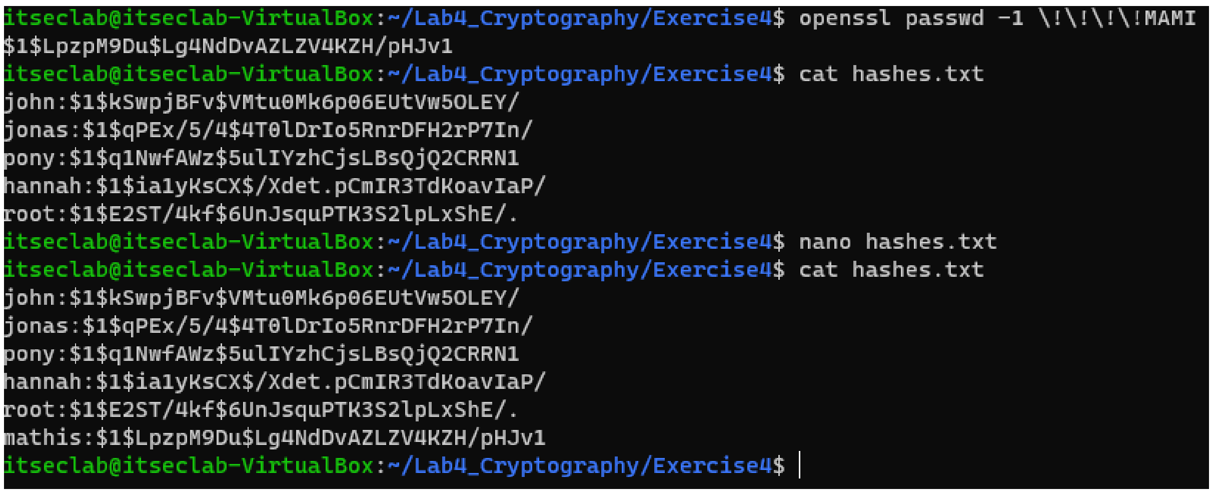
Figure 5: Encrypted HEX

CBC vs ECB

Which operation mode, do you think, is more secure and why?
We think that cbc is more secure because there is no repetition in the encrypted result. Whereas in ECB mode you can see that each block is the same, hence it is not depending on any other information than the block of input that is processed.

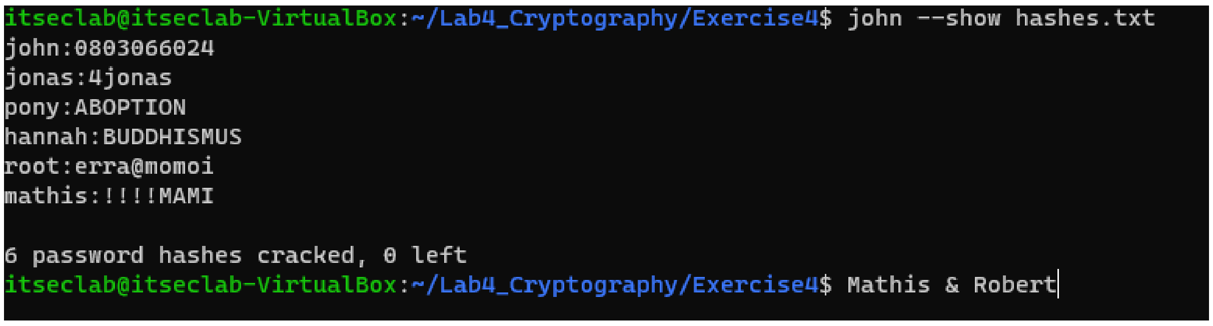
Part 4

My name is Mathis so I can choose the password '!!!!MAMI'.
By using openssl we can create the md5 hash and append it to the hashes file:



```
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$ openssl passwd -1 \!\!\!\!MAMI
$1$LpzpM9Du$Lg4NdDvAZLZV4KZH/pHJv1
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$ cat hashes.txt
john:$1$kSwpjBFv$VMtu0Mk6p06EUtVw5OLEY/
jonas:$1$qPEx/5/4$4T0lDrIo5RnrDFH2rP7In/
pony:$1$q1NwfAWz$5ulIYzhCjsLBsQjQ2CRRN1
hannah:$1$ia1yKsCX$/Xdet.pCmIR3TdKoavIaP/
root:$1$E2ST/4kf$6UnJsquPTK3S2lpLxShE/.
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$ nano hashes.txt
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$ cat hashes.txt
john:$1$kSwpjBFv$VMtu0Mk6p06EUtVw5OLEY/
jonas:$1$qPEx/5/4$4T0lDrIo5RnrDFH2rP7In/
pony:$1$q1NwfAWz$5ulIYzhCjsLBsQjQ2CRRN1
hannah:$1$ia1yKsCX$/Xdet.pCmIR3TdKoavIaP/
root:$1$E2ST/4kf$6UnJsquPTK3S2lpLxShE/.
mathis:$1$LpzpM9Du$Lg4NdDvAZLZV4KZH/pHJv1
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$
```

Figure 5: Create MD5 and append it



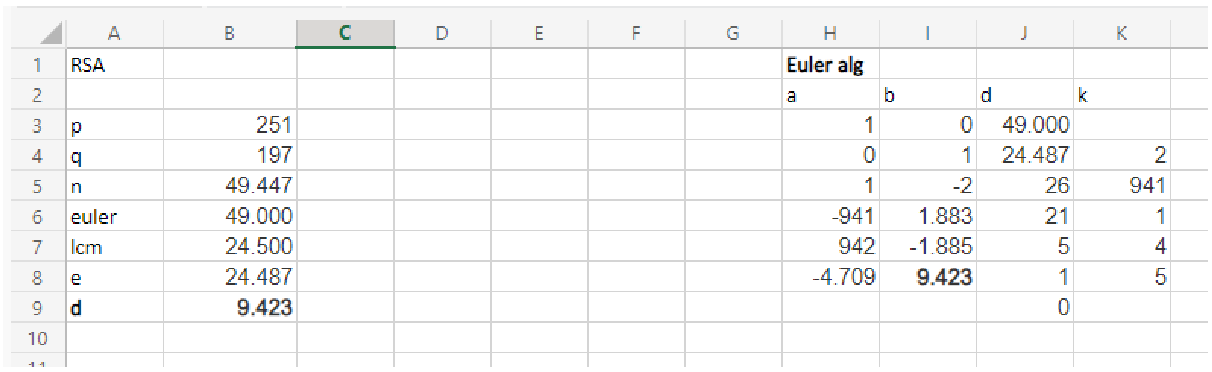
```
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$ john --show hashes.txt
john:0803066024
jonas:4jonas
pony:ABOPTION
hannah:BUDDHISMUS
root:erra@momoi
mathis:!!!!MAMI

6 password hashes cracked, 0 left
itseclab@itseclab-VirtualBox:~/Lab4_Cryptography/Exercise4$ Mathis & Robert
```

Figure 6: Cracked Passwords

Part 5

RSA calculate d

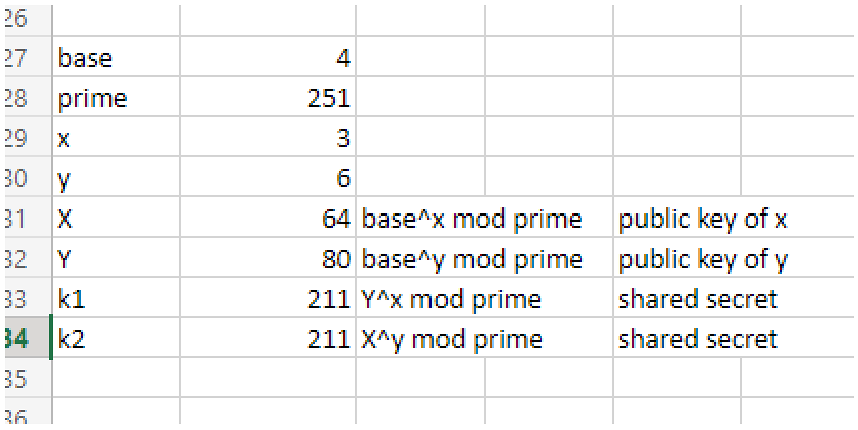


	A	B	C	D	E	F	G	H	I	J	K
1	RSA							**Euler alg**			
2								a	b	d	k
3	p	251						1	0	49.000	
4	q	197						0	1	24.487	2
5	n	49.447						1	-2	26	941
6	euler	49.000						-941	1.883	21	1
7	lcm	24.500						942	-1.885	5	4
8	e	24.487						-4.709	**9.423**	1	5
9	**d**	**9.423**								0	
10											

Figure 7: RSA calculate private key

Steps:

1. Generate primes p and q
2. calculate euler (p-1) * (q-1)
3. calculate lcm(p-1, q-1)
4. find e (coprime off lcm)
5. Use Euler algorithm on euler and e until reminder is 0
6. found d



26				
27	base	4		
28	prime	251		
29	x	3		
30	y	6		
31	X	64	base^x mod prime	public key of x
32	Y	80	base^y mod prime	public key of y
33	k1	211	Y^x mod prime	shared secret
34	k2	211	X^y mod prime	shared secret
35				

Figure 8: Diffie-Hellman

Steps:

1. Generate prime
2. choose x and y
3. calculate public keys X and Y
    a. base^x mod prime
    b. base^y mod prime
4. calculate shared secret s
    a. X^y mod prime
    b. Y^x mod prime

What is the Diffie-Hellman key exchange vulnerable to?

The Diffie-Hellman key exchange is vulnerable to a man-in-the-middle attack. If this is the case, someone can just sit between the persons communicating and intercept their messages. The *mitm* will substitute the key sent with its own and will therefore be able to de- and encrypt the messages. This will allow reading and modifying the messages.

What measures can be taken to prevent this type of attack?

The D.-H. key exchange does not identify the participants. So the people communicating should solve this problem by adding another protocol, which includes some unique identification of the specific participants.

For the D.-H., a generator *g* is used. Explain what a generator is and how can it be found?

"The protocol has two system parameters *p* and *g*. Parameter *p* is a prime number and parameter *g* (usually called a generator) is an integer less than *p*, with the following property: for every number *n* between 1 and *p*-1 inclusive, there is a power *k* of *g* such that *g*^*k*=*n* mod *p*."(security.nknu.edu.tw)
A generator creates numbers with specific rules set. There is most likely an equation in the background working with different most likely random numbers. These numbers seem random without the information about how they were created.

Show why for the primes [61,23] and the public key e=60 no private key d can be found

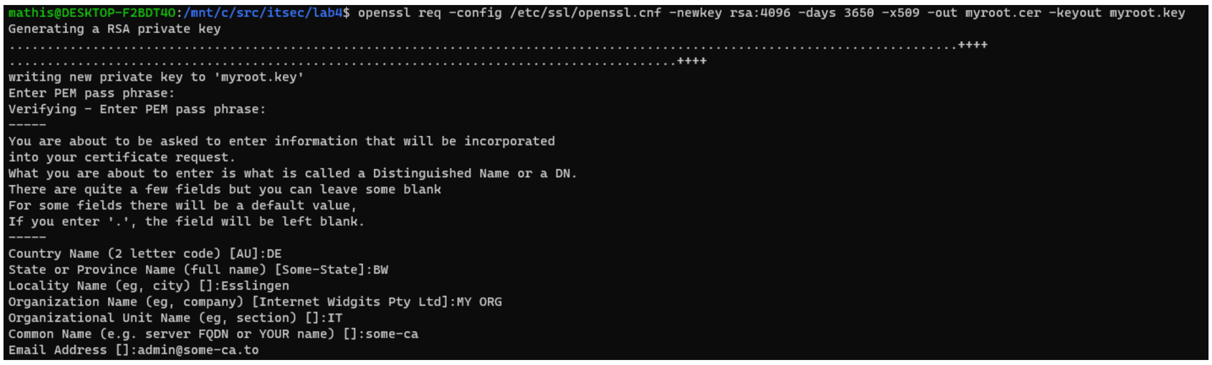
euler = (61 - 1) * (23 - 1) = 1320
60^(-1) mod 1320 *is not possible because 60 is not an invertible modulo of 1320.*

security.nknu.edu.tw;
http://security.nknu.edu.tw/crypto/faq/html/3-6-1.html#:~:text=The%20Diffie%2DHellman%20key%20exchange,own%20public%20value%20to%20Bob.

Part 6

Steps:

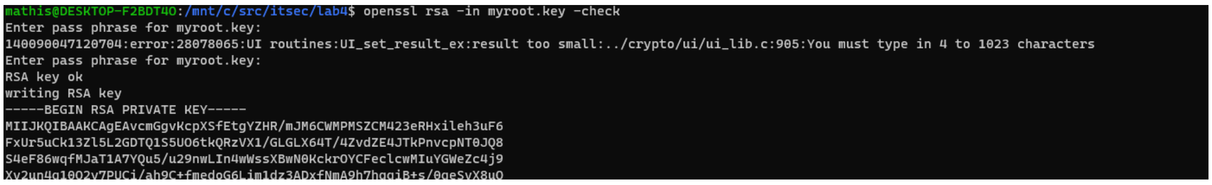
1. Generate certificate and private key with openssl
   a. choose strong passphrase
      aP2@iufu8w5K3raiZ7VEwlK6r7&egmx51Tky*R$39j9rp!pgA2VF%ACsa$m%71$2RG0
      m9Lrw5#o^54etE4PSbHRf@sDIQY$4s4q
   b. enter details



```
mathis@DESKTOP-F2BDT40:/mnt/c/src/itsec/lab4$ openssl req -config /etc/ssl/openssl.cnf -newkey rsa:4096 -days 3650 -x509 -out myroot.cer -keyout myroot.key
Generating a RSA private key
..........................................................................................................................++++
...........................................................................................++++
writing new private key to 'myroot.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:BW
Locality Name (eg, city) []:Esslingen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MY ORG
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:some-ca
Email Address []:admin@some-ca.to
```

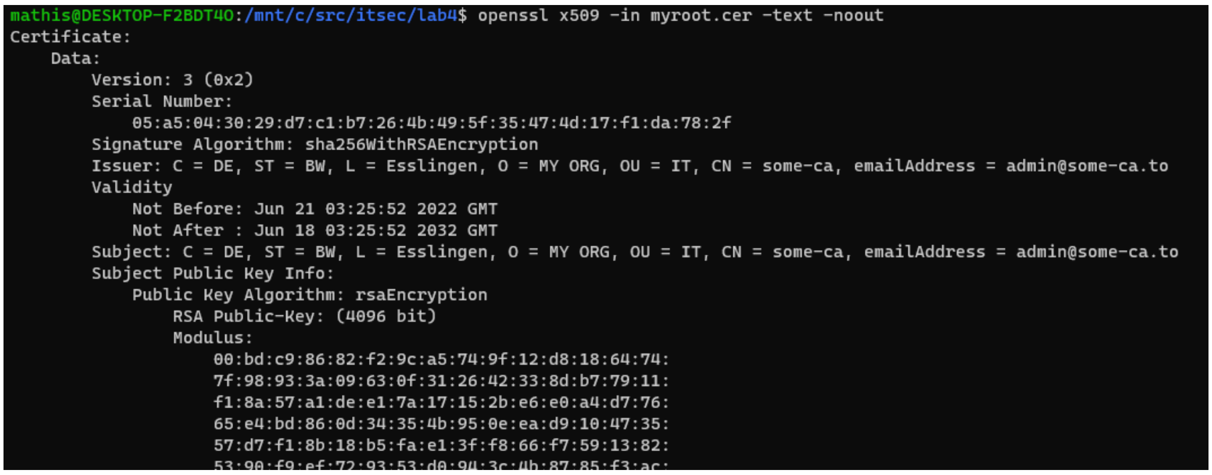
Figure 8: openssl generate cert and private key

Validation:



```
mathis@DESKTOP-F2BDT40:/mnt/c/src/itsec/lab4$ openssl rsa -in myroot.key -check
Enter pass phrase for myroot.key:
140090047120704:error:28078065:UI routines:UI_set_result_ex:result too small:../crypto/ui/ui_lib.c:905:You must type in 4 to 1023 characters
Enter pass phrase for myroot.key:
RSA key ok
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIIJKQIBAAKCAgEAvcmGgvKcpXSfEtgYZHR/mJM6CWMPMSZCM423eRHxileh3uF6
FxUr5uCk13Zl5L2GDTQ1S5UO6tkQRzVX1/GLGLX64T/4ZvdZE4JTkPnvcpNT0JQ8
S4eF86wqfMJaT1A7YQu5/u29nwLIn4wWssXBwN0KckrOYCFeclcwMIuYGWeZc4j9
Xv2un4q1002v7PUCi/ah9C+fmedoG6Lim1dz3ADxfNmA9h7hqqiB+s/0qeSyX8uO
```

Figure 9: validate private key



```
mathis@DESKTOP-F2BDT40:/mnt/c/src/itsec/lab4$ openssl x509 -in myroot.cer -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            05:a5:04:30:29:d7:c1:b7:26:4b:49:5f:35:47:4d:17:f1:da:78:2f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = DE, ST = BW, L = Esslingen, O = MY ORG, OU = IT, CN = some-ca, emailAddress = admin@some-ca.to
        Validity
            Not Before: Jun 21 03:25:52 2022 GMT
            Not After : Jun 18 03:25:52 2032 GMT
        Subject: C = DE, ST = BW, L = Esslingen, O = MY ORG, OU = IT, CN = some-ca, emailAddress = admin@some-ca.to
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:bd:c9:86:82:f2:9c:a5:74:9f:12:d8:18:64:74:
                    7f:98:93:3a:09:63:0f:31:26:42:33:8d:b7:79:11:
                    f1:8a:57:a1:de:e1:7a:17:15:2b:e6:e0:a4:d7:76:
                    65:e4:bd:86:0d:34:35:4b:95:0e:ea:d9:10:47:35:
                    57:d7:f1:8b:18:b5:fa:e1:3f:f8:66:f7:59:13:82:
                    53:90:f9:ef:72:93:53:d0:94:3c:4b:87:85:f3:ac:
```

Figure 10: validate public key