CSCI 476: Computer Security

Secret Key Encryption/Symmetric Cryptography (Part 1)

Reese Pearsall

Fall 2024

Announcement

DNS Lab due Sunday (11/10)

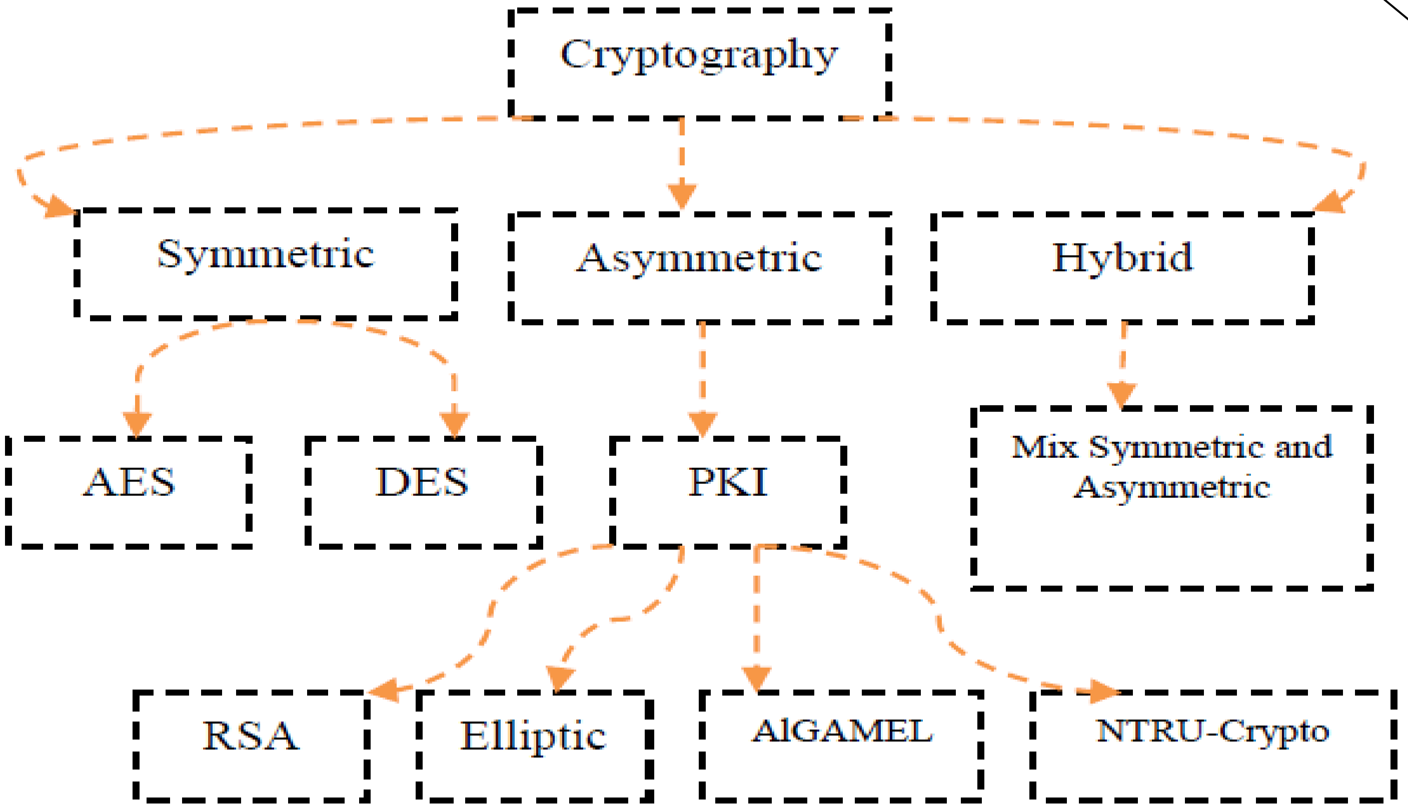
Project due one week from toady (11/21)

One week from today, I will be gone (recorded lecture will be posted before class time)

Information Security

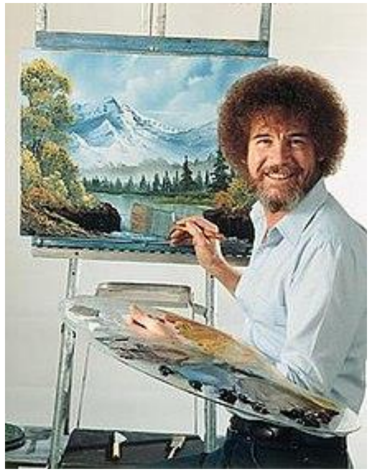
- The protection of information and information systems

Cryptography is the practice and study of techniques for securing communications and data in the presence of adversaries



There are many types of encryption

Bob

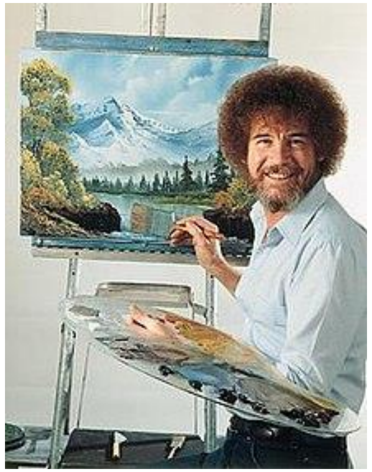


“Hi Alice, my address is 123 Painting Avenue. Please stop by at 6:00”

Over a wire, wirelessly, via a Pidgeon etc

Alice

Bob



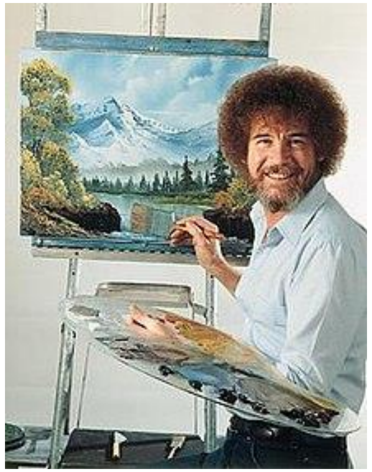
“Hi Alice, my address is
123 Painting Avenue.
Please stop by at 6:00”

Alice

Because our transmission medium is **shared**, there is a possible someone else could be eavesdropping

Eve

Bob



“Hi Alice, my address is
123 Painting Avenue.
Please stop by at 6:00”

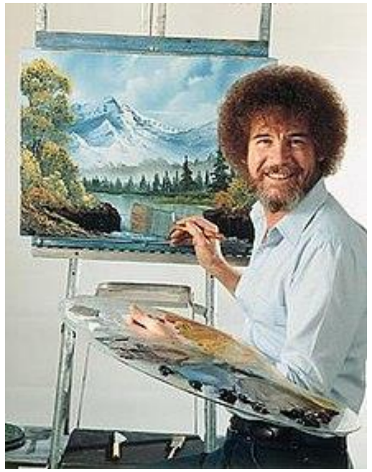
Alice

Because our transmission medium is **shared**, there is a possible someone else could be eavesdropping

Eve

Our goal is to make sure Alice can receive our message securely, and our original message cannot be intercepted

Bob



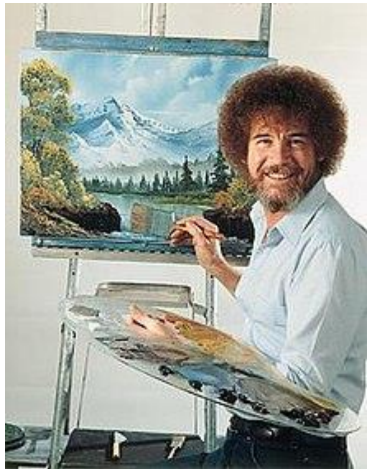
Alice

Eve

Cleartext/Plaintext

“Hi Alice, my address is 123 Painting Avenue. Please stop by at 6:00”

Bob

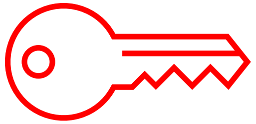


Alice

Eve

Cleartext/Plaintext

“Hi Alice, my address is 123 Painting Avenue. Please stop by at 6:00”

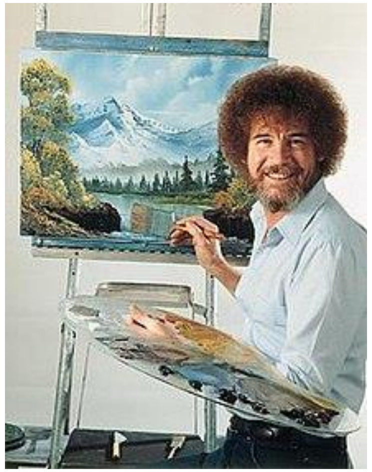


Bob **encrypts** his message with a **key**

MuYGoP5LiTTGPVX6U/r2VTpxPSqT
Fmy5nsoFWURThKMhHk/7tbjYsS2EJ
917q7megTAcV+V4ZMU4HjJjiW2DC
BroxvJ0V3ZYDgZ8B9lUvGUmdiRMH
25Xkf7QrhAGR3FF

The result is a **ciphertext**

Bob



MuYGoP5LiTTGPVX6U/r2VTpxPSqTFmy5nsoF
WURThKMhHk/7tbjYsS2EJ917q7megTAcV+V4Z
MU4HjJjiW2DCBroxvJ0V3ZYDgZ8B9lUvGUmdiR
MH25Xkf7QrhAGR3FF

Alice

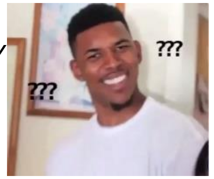
Cleartext/Plaintext

“Hi Alice, my address is 123 Painting Avenue. Please stop by at 6:00”

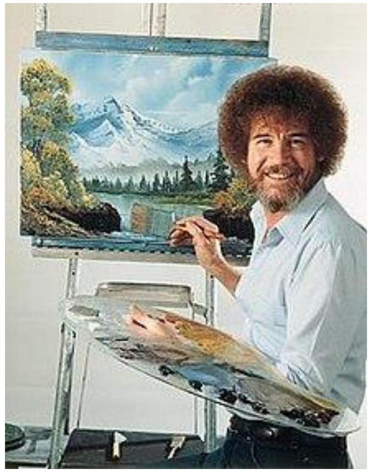
Eve

If Eve intercepts our ciphertext, she can't do very much with it

MuYGoP5LiTTGPVX6U/r2VTpx
PSqTFmy5nsoFWURThKMhHk
/7tbjYsS2EJ917q7megTAcV+V
4ZMU4HjJjiW2DCBroxvJ0V3ZY
DgZ8B9lUvGUmdiRMH25Xkf7
QrhAGR3FF



Bob



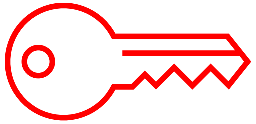
Cleartext/Plaintext

“Hi Alice, my address is 123 Painting Avenue. Please stop by at 6:00”

Eve

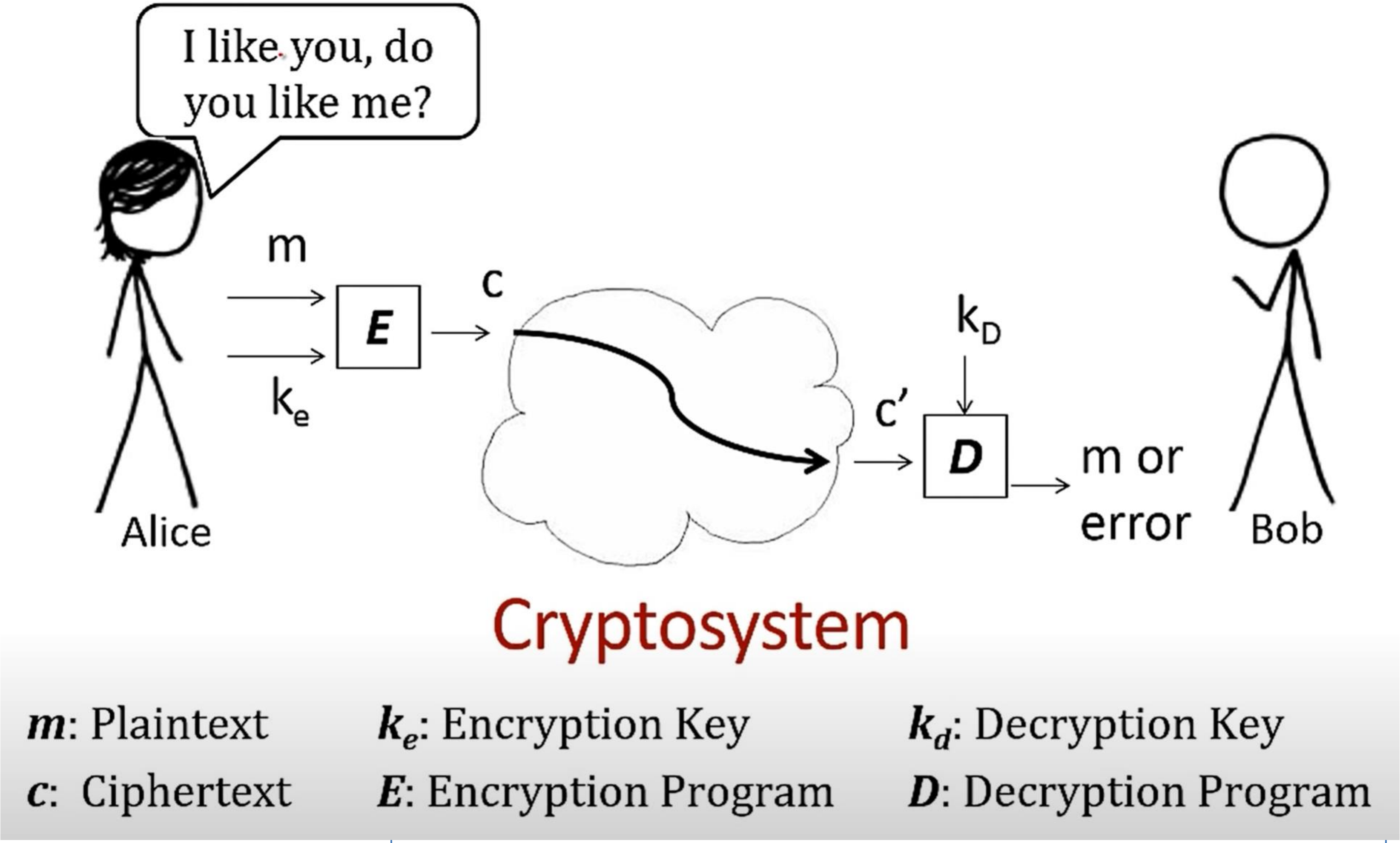
Alice

MuYGoP5LiTTGPVX6U/r2
VTpxPSqTFmy5nsoFWUR
ThKMhHk/7tbjYsS2EJ917
q7megTAcV+V4ZMU4HjJji
W2DCBroxvJ0V3ZYDgZ8
B9lUvGUmdiRMH25Xkf7
QrhAGR3FF



Alice receives the ciphertext, and then uses the **same key** that bob used, and then **decrypts** the ciphertext

“Hi Alice, my address is 123 Painting Avenue. Please stop by at 6:00”



Deterministic programs*

The importance here is that the **keys** used for encryption/decryption are secret (ie not public knowledge)

The innerworkings of the encryption/decryption program *is* public knowledge though

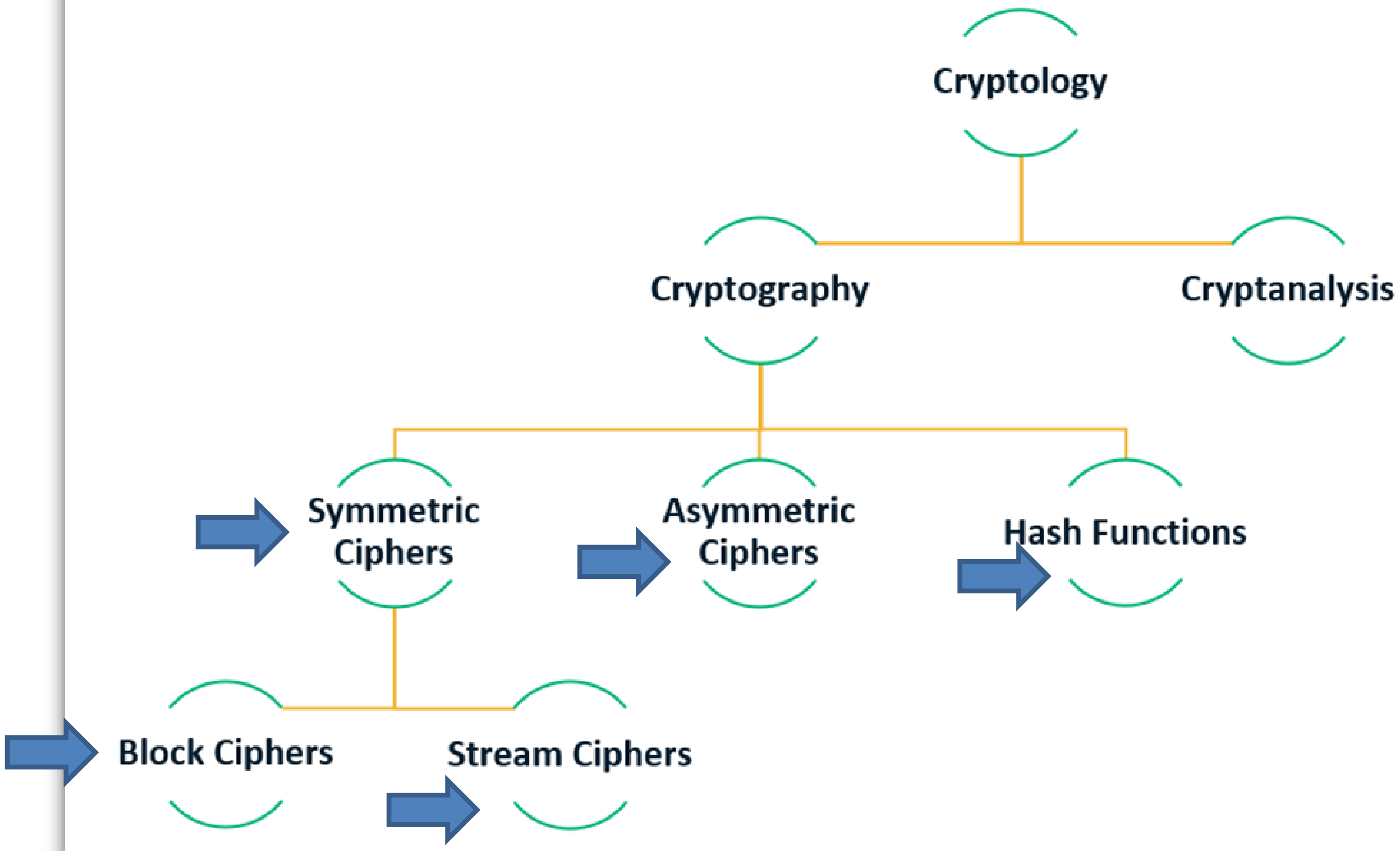
Secure cryptography is the foundation for our secure communications in the cyber world (HTTPS, SSH, etc)

The encryption algorithms are typically rooted in **very difficult problems** in computing (ie there does not exist a program that can efficiently break RSA **YET**)

There are very intense proofs and prove the secureness of the encryption procedures we use today

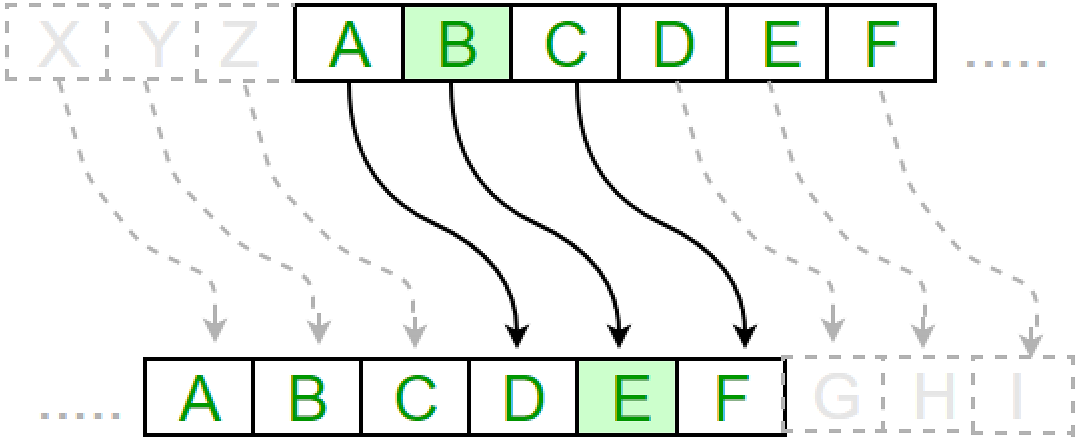
Never try to roll out your own cryptography scheme, and never use the built-in RNG for secure communications (import random)

Cryptology vs Cryptography



Early cryptography techniques

Shift 3



Caesar Cipher- letters in the plaintext will be replaced by some fixed number of positions downs in the alphabet.

plaintext

hello there world my name is reese

ciphertext

khoor wkhuh zruog pb
qdph lv uhhvh

Nifty, but we have the technology to brute force 26 possible shifts

Substitution Cipher

Letters in plaintext are substituted by another letter

E→X
R → Z

REESE = ZXXSX

Monolithic Substitution Cipher – Same “rules” are applied throughout the entire plaintext

Polyalphabetic Substitution Cipher – different “rules” are applied throughout the plaintext

Here is a ciphertext (cipher.txt)

```
ydq ufyiqoobxrk lrcqx yqoy fo r kwgyfoyrbq rqxepfc crlrcfyt yqoy ydry lxebxqoofvqgt bqyo kexq
mfuufcwgy ro fy ceiyfiwqo. ydq ysqiyt kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq
oyrxy. ydq xwiifib olqqm oyrxyo ogesgt, pwy bqyo uroyqx qrcd kfiwyq ruyqx tew dqrx ydfo ofbirg pqql r
ofibgq grl odewgm pq ceklgqyqm qrcd yfkq tew dqrx ydfo oewim. [mfib] xqkqkpqx ye xwi fi r oyxrfbdy
gfiq, rim xwi ro geib ro leoofpgq. ydq oqceim yfkq tew urfg ye ceklgqyq r grl pquexq ydq oewim, tewx
yqoy fo evqx. ydq yqoy sfgg pqbfi ei ydq sexm oyrxy. ei tewx krxj, bqy xqrmt, oyrxy.
```

Suppose we know that that this message is an english message encrypted with a monolithic substitution cipher

Can we crack this?

Here is a ciphertext (cipher.txt)

```
ydq ufyiqoobxrk lrcqx yqoy fo r kwgyfoyrbq rqxepfc crlrcfyt yqoy ydry lxebxqoofvqgt bqyo kexq
mfuufcwgy ro fy ceiyfiwqo. ydq ysqiyt kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq
oyrxy. ydq xwiifib olqqm oyrxyo ogesgt, pwy bqyo uroyqx qrcd kfiwyq ruyqx tew dqrx ydfo ofbirg pqql r
ofibgq grl odewgm pq ceklgqyqm qrcd yfkq tew dqrx ydfo oewim. [mfib] xqkqkpqx ye xwi fi r oyxrfbdy
gfiq, rim xwi ro geib ro leoofpgq. ydq oqceim yfkq tew urfg ye ceklgqyq r grl pquexq ydq oewim, tewx
yqoy fo evqx. ydq yqoy sfgg pqbfi ei ydq sexm oyrxy. ei tewx krxj, bqy xqrmt, oyrxy.
```

Frequency Analysis leverages the fact that in any given written language, certain letters and combinations occur more frequently than others

In English, T, A , I, and O are the most common letters, so it is likely the letters that appear the most frequently in our ciphertext are one of those

```
ydq ufyiqoobxrk lrcqx yqoy fo r kwgyfoyrbq rqxepfc crlrcfyt yqoy ydry lxebxqoofvqgt bqyo kexq
mfuufcwgy ro fy ceiyfiwqo. ydq ysqiyt kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq
oyrxy. ydq xwiifib olqqm oyrxyo ogesgt, pwy bqyo uroyqx qrcd kfiwyq ruyqx tew dqrx ydfo ofbirg pqql r
ofibgq grl odewgm pq ceklgqyqm qrcd yfkq tew dqrx ydfo oewim. [mfib] xqkqkpqx ye xwi fi r oyxrfbdy
gfiq, rim xwi ro geib ro leoofpgq. ydq oqceim yfkq tew urfg ye ceklgqyq r grl pquexq ydq oewim, tewx
yqoy fo evqx. ydq yqoy sfgg pqbfi ei ydq sexm oyrxy. ei tewx krxj, bqy xqrmt, oyrxy.
```

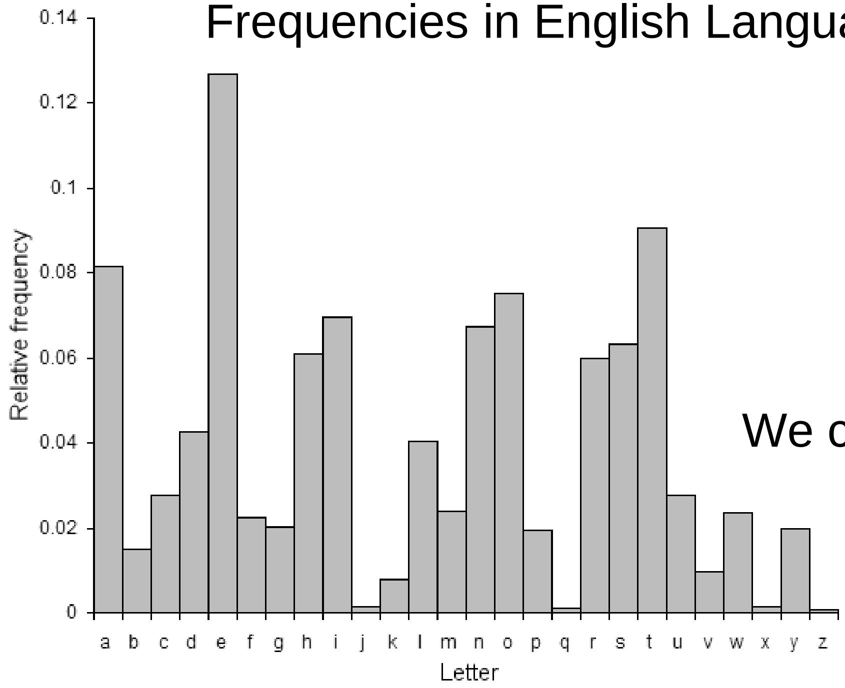
Here is a ciphertext (cipher.txt)

We can write a program that counts the frequency of characters (**1-gram**) and frequency of character pairs (**2-gram**)

```
[11/03/22]seed@VM:~/encyption_lecture$ ./freq.py < ciphertext.txt
-------------------------------------
1-gram (top 20):
q: 61
y: 58
o: 39
r: 34
f: 32
x: 30
i: 27
e: 26
g: 21
d: 18
w: 17
b: 14
c: 13
k: 12
l: 12
m: 12
t: 11
p: 9
.: 8
u: 7
```

```
2-gram (top 20):
yd: 12
oy: 12
yq: 11
fi: 11
dq: 10
qo: 8
qx: 8
ew: 8
rx: 7
qy: 6
ei: 6
pq: 6
rc: 5
fo: 5
yr: 5
xq: 5
ce: 5
xy: 5
im: 5
wi: 5
```

Frequencies in English Language



We can start making guesses!

Most common bigrams (in order)

th, he, in, en, nt, re, er, an, ti, es, on, at, se, nd, or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro.

```
ydq ufyiqoobxrk lrcqx yqoy fo r kwgyfoyrbq rqxepfc crlrcfyt yqoy ydry lxebxqoofvqgt bqyo kexq
mfuufcwgy ro fy ceiyfiwqo. ydq ysqiyt kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq
oyrxy. ydq xwiifib olqqm oyrxyo ogesgt, pwy bqyo uroyqx qrcd kfiwyq ruyqx tew dqrx ydfo ofbirg pqql r
ofibgq grl odewgm pq ceklgqyqm qrcd yfkq tew dqrx ydfo oewim. [mfib] xqkqkpqx ye xwi fi r oyxrfbdy
gfiq, rim xwi ro geib ro leoofpgq. ydq oqceim yfkq tew urfg ye ceklgqyq r grl pquexq ydq oewim, tewx
yqoy fo evqx. ydq yqoy sfgg pqbfi ei ydq sexm oyrxy. ei tewx krxj, bqy xqrmt, oyrxy.
```

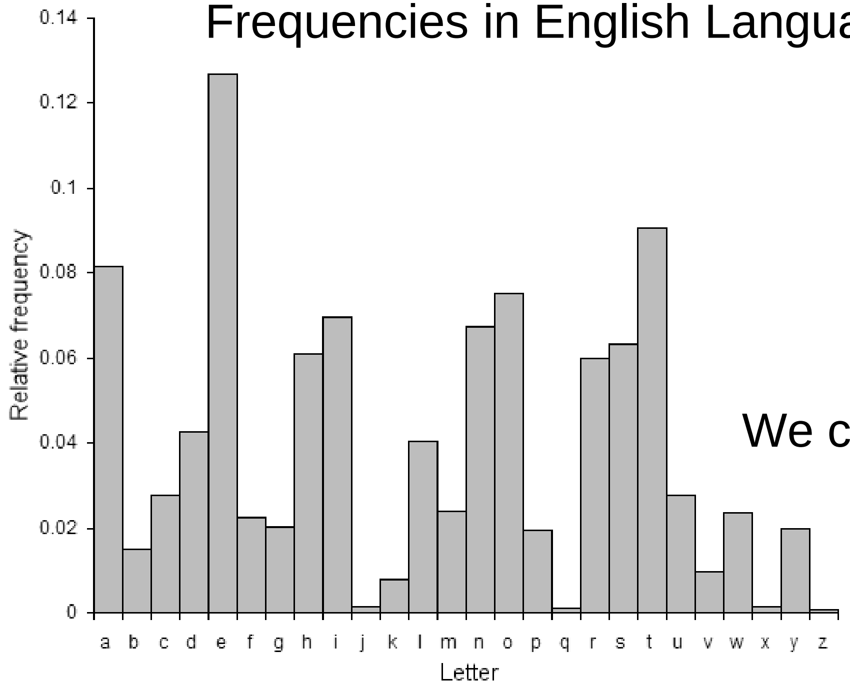
Here is a ciphertext (cipher.txt)

We can write a program that counts the frequency of characters (**1-gram**) and frequency of character pairs (**2-gram**)

```
[11/03/22]seed@VM:~/encyption_lecture$ ./freq.py < ciphertext.txt
-------------------------------------
1-gram (top 20):
q: 61
y: 58
o: 39
r: 34
f: 32
x: 30
i: 27
e: 26
g: 21
d: 18
w: 17
b: 14
c: 13
k: 12
l: 12
m: 12
t: 11
p: 9
.: 8
u: 7
```

```
2-gram (top 20):
yd: 12
oy: 12
yq: 11
fi: 11
dq: 10
qo: 8
qx: 8
ew: 8
rx: 7
qy: 6
ei: 6
pq: 6
rc: 5
fo: 5
yr: 5
xq: 5
ce: 5
xy: 5
im: 5
wi: 5
```

Frequencies in English Language



We can start making guesses!

Most common bigrams (in order)

th, he, in, en, nt, re, er, an, ti, es, on, at, se, nd, or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro.

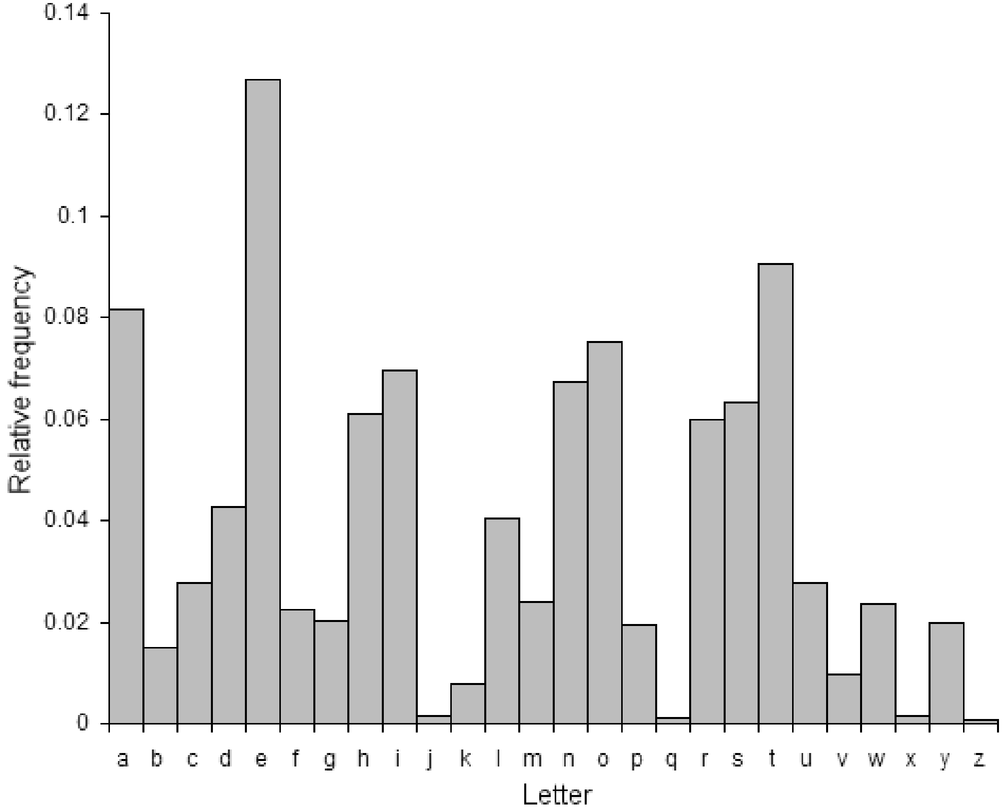
Listing 24.2: Bigram and trigram frequencies

Bigram frequency in English

TH	2.71	EN	1.13	NG	0.89
HE	2.33	AT	1.12	AL	0.88
IN	2.03	ED	1.08	IT	0.88
ER	1.78	ND	1.07	AS	0.87
AN	1.61	TO	1.07	IS	0.86
RE	1.41	OR	1.06	HA	0.83
ES	1.32	EA	1.00	ET	0.76
ON	1.32	TI	0.99	SE	0.73
ST	1.25	AR	0.98	OU	0.72
NT	1.17	TE	0.98	OF	0.71

Trigram frequency in English

THE	1.81	ERE	0.31	HES	0.24
AND	0.73	TIO	0.31	VER	0.24
ING	0.72	TER	0.30	HIS	0.24
ENT	0.42	EST	0.28	OFT	0.22
ION	0.42	ERS	0.28	ITH	0.21
HER	0.36	ATI	0.26	FTH	0.21
FOR	0.34	HAT	0.26	STH	0.21
THA	0.33	ATE	0.25	OTH	0.21
NTH	0.33	ALL	0.25	RES	0.21
INT	0.32	ETH	0.24	ONT	0.20



```
[11/03/22]seed@VM:~/encyption_lecture$ tr 'y' 't' < ciphertext.txt > output.txt
```

Translate ciphertext.txt, and replace all **y** with **t**

```
[11/03/22]seed@VM:~/encyption_lecture$ cat output.txt
tdq uftiqoobxrk lrcqx tqot fo r kwgtfotrbq rqxepfc crlrcftt tqot tdrt lxebxqoofvqgt bqto kexq mfuufcwgt ro ft ceitfiwqo. tdq tsqitt kqtqx lrcqx tqot sfgg pqbfi fi tdfxtt
 oqceimo. gfiq wl rt tdq otrxt. tdq xwiifib olqqm otrxto ogesgt, pwt bqto urotqx qrcd kfiwtq rutqx tew dqrx tdfo ofbirg pqql r ofibgq grl odewgm pq ceklgqtqm qrcd tfkq t
ew dqrx tdfo oewim. [mfib] xqkqkpqx te xwi fi r otxrfbdt gfiq, rim xwi ro geib ro leoofpgq. tdq oqceim tfkq tew urfg te ceklgqtq r grl pquexq tdq oewim, tewx tqot fo evq
x. tdq tqot sfgg pqbfi ei tdq sexm otrxt. ei tewx krxj, bqt xqrmt, otrxt.
```

```
[11/03/22]seed@VM:~/encyption_lecture$ tr 'yd' 'th' < ciphertext.txt > output.txt
```

Translate ciphertext.txt, and replace all **y** with **t,** and replace all **d** with **h**

```
thq uftiqoobxrk lrcqx tqot fo r kwgtfotrbq rqxepfc crlrcftt tqot thrt lxebxqoofvqgt bqto kexq mfuufcwgt ro ft ceitfiwqo. thq tsqitt kqtqx lrcqx tqot sfgg pqbfi fi thfxtt
 oqceimo. gfiq wl rt thq otrxt. thq xwiifib olqqm otrxto ogesgt, pwt bqto urotqx qrch kfiwtq rutqx tew hqrx thfo ofbirg pqql r ofibgq grl ohewgm pq ceklgqtqm qrch tfkq t
ew hqrx thfo oewim. [mfib] xqkqkpqx te xwi fi r otxrfbht gfiq, rim xwi ro geib ro leoofpgq. thq oqceim tfkq tew urfg te ceklgqtq r grl pquexq thq oewim, tewx tqot fo evq
x. thq tqot sfgg pqbfi ei thq sexm otrxt. ei tewx krxj, bqt xqrmt, otrxt.
```

Keep adding more characters to your decryption scheme until you get the full answer ☺

Review the XOR operator:

Everything on a computer is **zeros** and **ones**

0101010101010010111101010100
1000010111001000101010101100
1010101010111110100100101010
1010010101010110010101011010
1001010101010101010101001010
1010101010101001010101010101
01011010010101010100101…

Hello world

01101000 01100101 01101100
01101100 01101111 00100000
01110111 01101111 01110010
01101100 01100100 00001010

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

$1 \oplus 0 = 1$
$0 \oplus 0 = 0$
$1 \oplus 1 = 0$
$0 \oplus 1 = 1$

Message:	0001 1010 0011
Key:	$\oplus$ 1100 1100 0101
Ciphertext:	1101 0110 0110

How to get original message?

Review the XOR operator:

Everything on a computer is **zeros** and **ones**

01010101010100101111101010100
10000101111001000101010101100
10101010101111110100100101010
1010010101010110010101011010
100101010101010101010101001010
1010101010101001010101010101
01011010010101010100101…

Hello world

01101000 01100101 01101100
01101100 01101111 00100000
01110111 01101111 01110010
01101100 01100100 00001010

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

$1 \oplus 0 = 1$
$0 \oplus 0 = 0$
$1 \oplus 1 = 0$
$0 \oplus 1 = 1$

Message: 0001 1010 0011

Key: $\oplus$ 1100 1100 0101

Ciphertext: 1101 0110 0110

$\oplus$ 1100 1100 0101

XOR with the key again! 0001 1010 0011

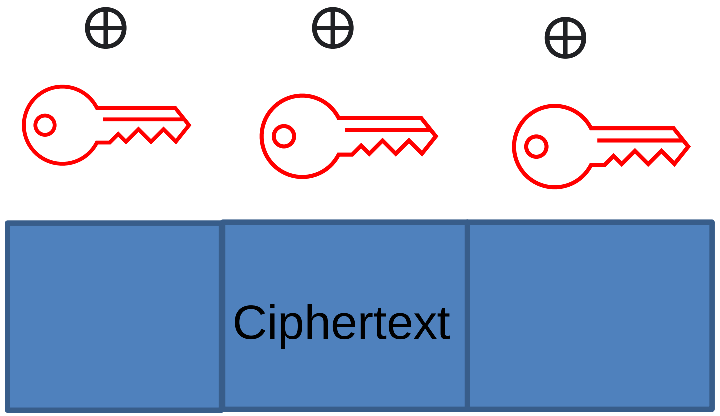
Block Cipher

Split in messages into fixed sized blocks, encrypt eac
block separately

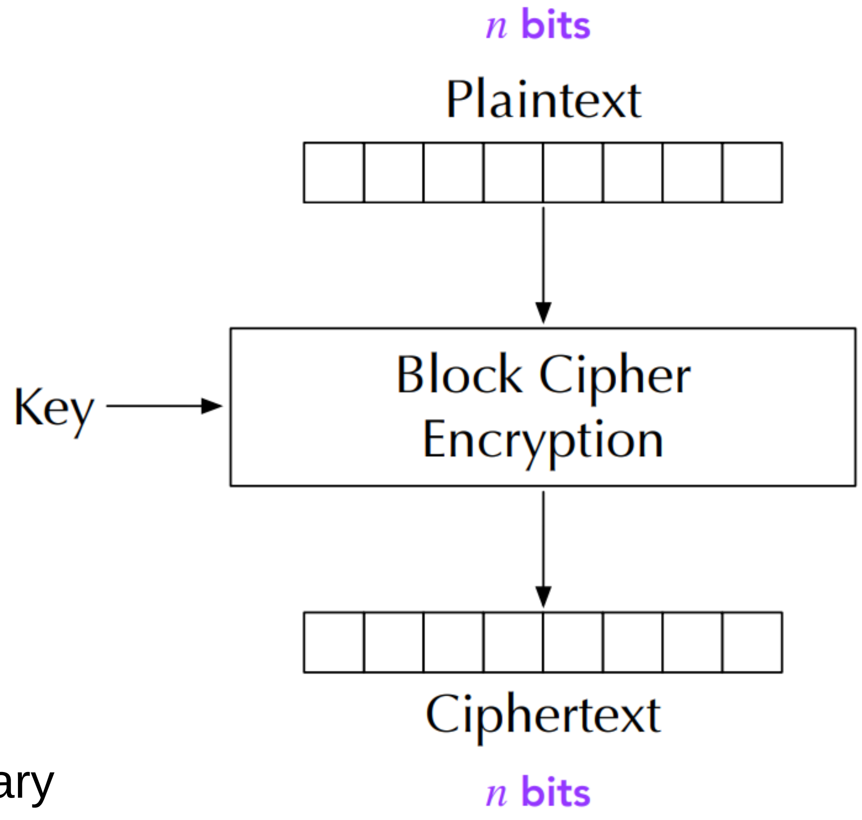
Hello there world

01101000	01100101	01101100
01101100	01101111	00100000
01110100	01101000	01100101
01110010	01100101	00100000
01110111	01101111	01110010
01101100	01100100	00001010

Block 1 Block 2 Block 3



The specifics of this operation vary depending on your mode of encryption



Decryption is performed by applying the reverse transformation to ciphertext blocks

Important Properties

- Even small differences in plaintext result in different ciphertexts
- Blocks in plaintext that are the same will also have matching ciphertexts

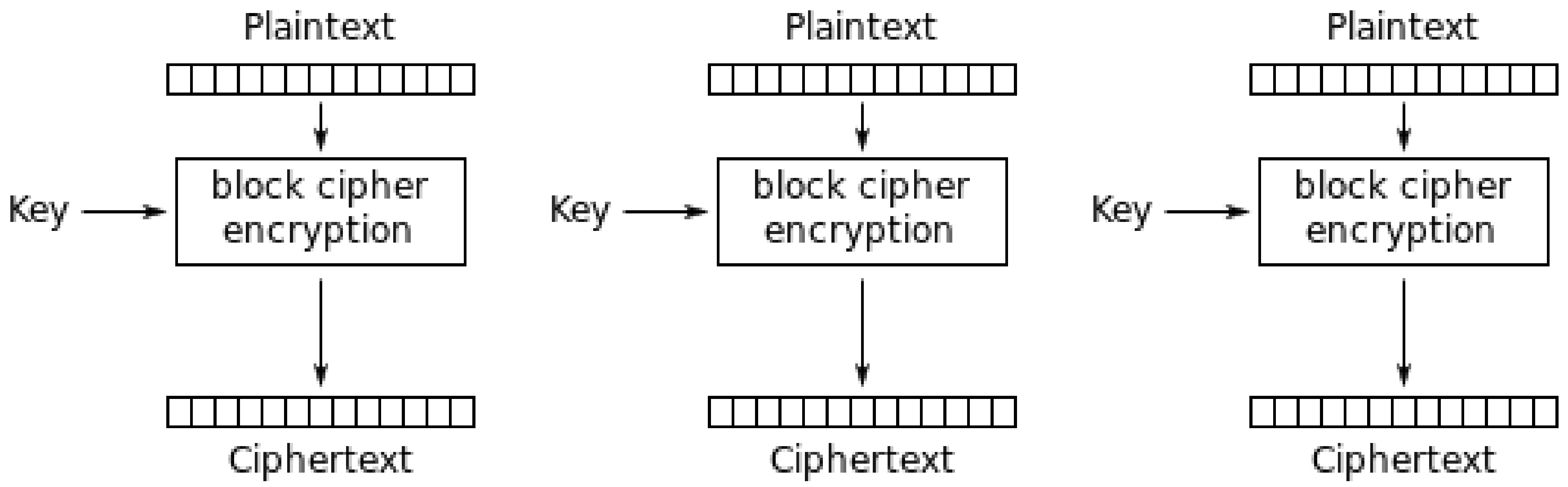
Modes of Encryption

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Propagating CBC (PCBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)

All block ciphers!

But if we aren't careful about how we conduct encryption operations, we may accidentally reveal information about the plaintext

Electronic Codebook **ECB**



Electronic Codebook (ECB) mode encryption

Notice: For the same key, a plaintext always maps to the same ciphertext

Using OpenSSL to encrypt w/ ECB

Encrypt a .txt file

```
openssl enc -aes-128-ecb -e -in plain.txt -out cipher.txt \
     -K 00112233445566778899AABBCCDDEEFF
```

1 Encrypt using AES (block cipher) with mode ECB using a 128-bit key

2 **E**ncrypt

3 Input file to be encrypted will be *plain.txt*

4 Output file created that contains the ciphertext will be *cipher.txt*

5 Key used for encryption will be 00112233445566778899AABBCCDDEEFF   32 characters in hex → 128 bits

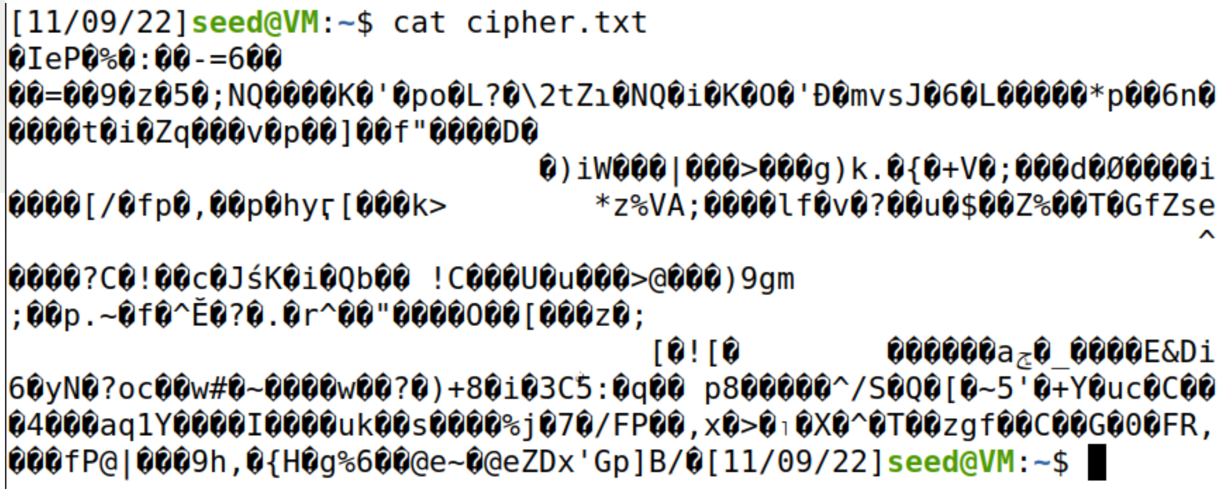
Using OpenSSL to encrypt w/ ECB

Encrypt a .txt file

```
openssl enc -aes-128-ecb -e -in plain.txt -out cipher.txt \
    -K 00112233445566778899AABBCCDDEEFF
```

plain.txt

```
1 The FitnessGram Pacer Test is a multistage aerobic capacity
  test that progressively gets more difficult as it continues.
  The 20 meter pacer test will begin in 30 seconds. Line up at
  the start. The running speed starts slowly, but gets faster
  each minute after you hear this signal. [beep] A single lap
  should be completed each time you hear this sound. [ding]
  Remember to run in a straight line, and run as long as
  possible. The second time you fail to complete a lap before the
  sound, your test is over. The test will begin on the word
  start. On your mark, get ready, start.
```



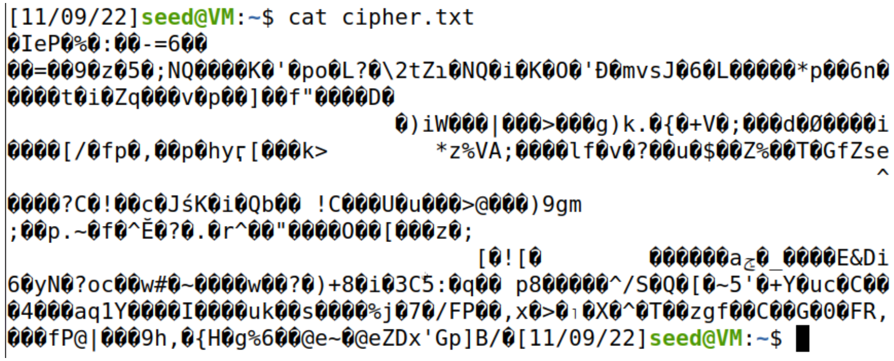
Using OpenSSL to encrypt w/ ECB

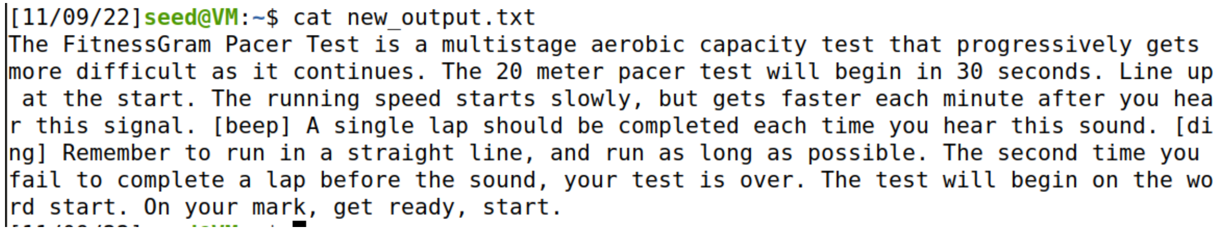
Encrypt a .txt file

```
openssl enc -aes-128-ecb -e -in plain.txt -out cipher.txt \
     -K 00112233445566778899AABBCCDDEEFF
```

Decrypt a .txt file

```
openssl enc -aes-128-ecb -d -in cipher.txt -out new_output.txt \
     -K 00112233445566778899AABBCCDDEEFF
```





Using OpenSSL to encrypt w/ ECB

Encrypt a .txt file

```
openssl enc -aes-128-ecb -e -in plain.txt -out cipher.txt \
     -K 00112233445566778899AABBCCDDEEFF
```

Decrypt a .txt file

```
openssl enc -aes-128-ecb -d -in cipher.txt -out new_output.txt \
     -K 00112233445566778899AABBCCDDEEFF
```

Changing the key used for decryption wont decrypt correctly!

```
[11/09/22]seed@VM:~$ openssl enc -aes-128-ecb -d -in cipher.txt -out new_output.txt -K 00
112233445566778899AABBCCDDEEEE
bad decrypt
140636099929408:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:
crypto/evp/evp_enc.c:583:
[11/09/22]seed@VM:~$ cat new_output.txt
��v.��>Ӝ!�@.�~hy4c��A}�����(��tg{�M��q��u(��KU�h�%g�zmH����(���
g'���]���5n�����kD���'L���a���7�Vf�(���K�^2��J/3;2Y�q��b���&w�O-hQ���zY��R+���C�?��ͱ�����
?�'�o��qj?�~A5J/;F.L/D?V�Ů/��fO�m�����M��t��HɑDr.#.�
                                            ����i��s*����&F/���Bv�w=
                                                                 d�>��r�Q�3�i����r�z
}d��dA�����]F����3����]̊:�ZX�/�?h�Y�ͺd�2W��w�5桢�<͓�z����r��|��2�|�U�bb�[11/09/22]seed@VM:
~$
```