

CSCI 476: Computer Security

Secret Key Encryption/Symmetric Cryptography

Reese Pearsall
Fall 2022

Announcement

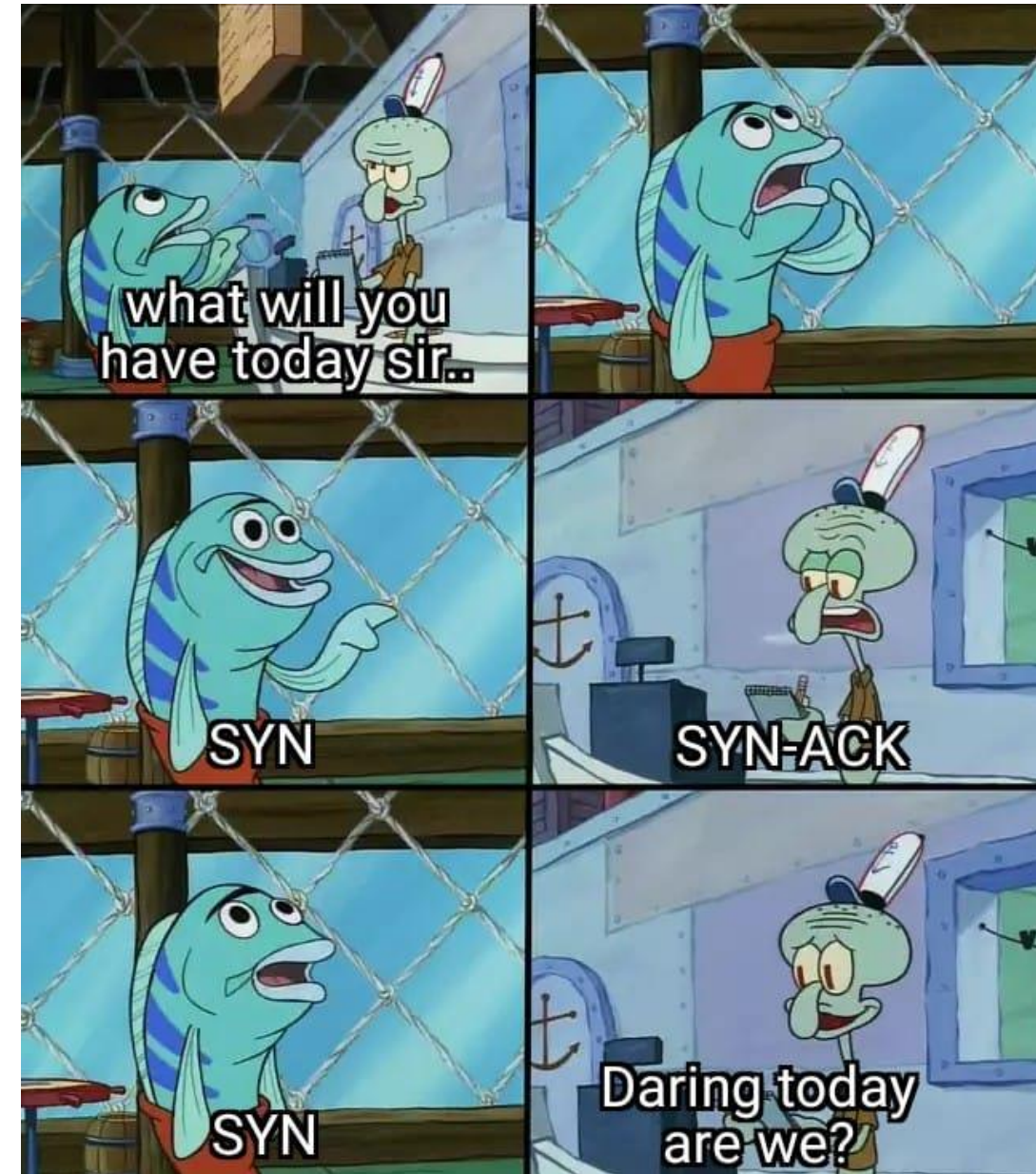
Lab 7 (TCP attacks) Due

Thursday November 10th

→ Sounds like we have some issues with the C program

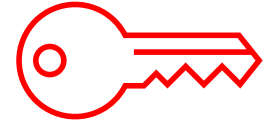
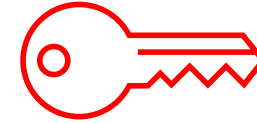
No class on Tuesday next week
(11/8) (go vote)

Grading rubric now on project
instructions webpage

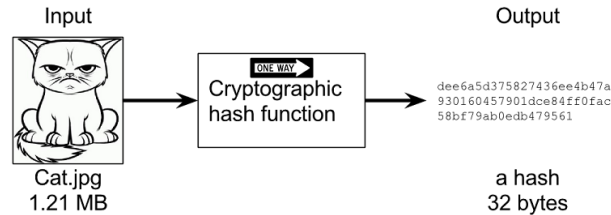


Crypto Roadmap

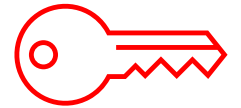
- Secret-Key Encryption (a.k.a Symmetric Key Encryption)



- Cryptographic Hash Functions



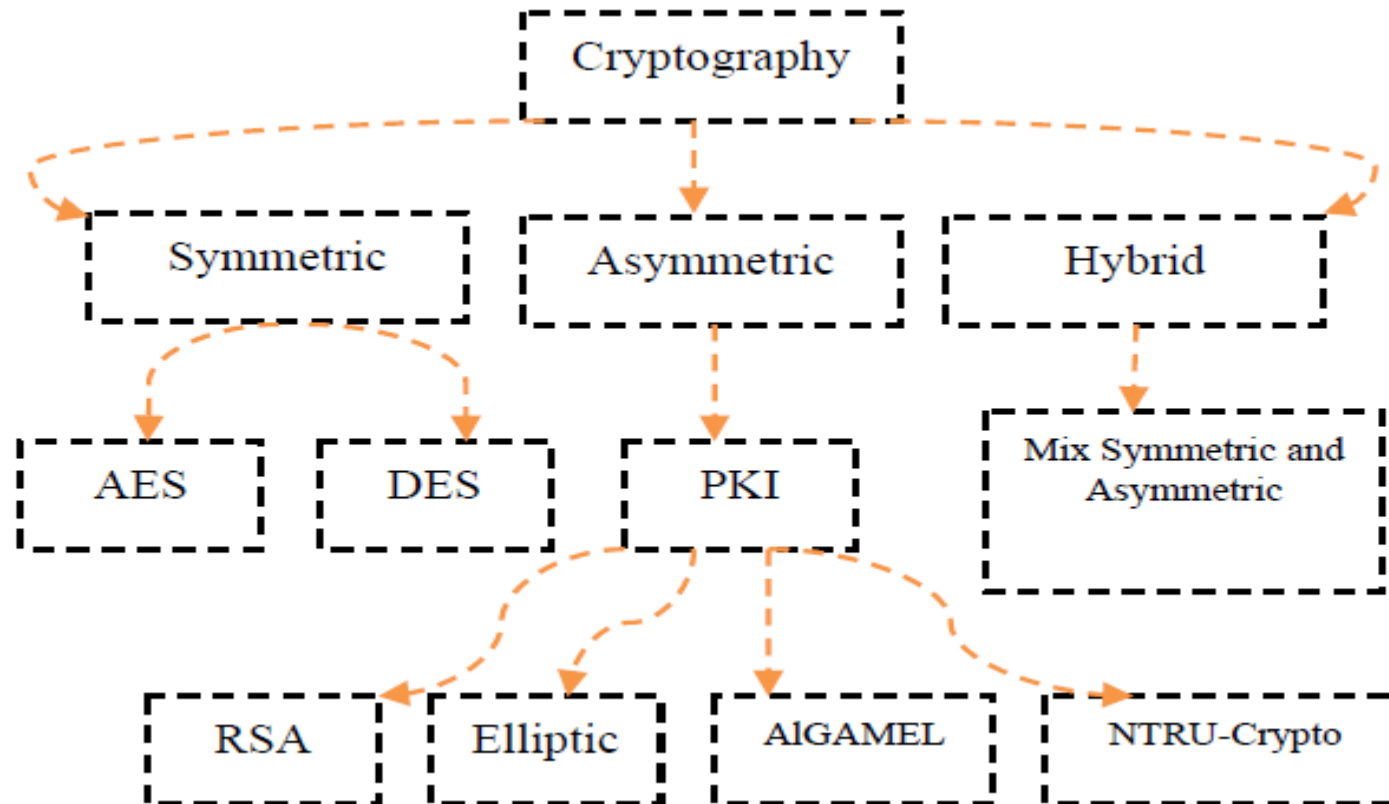
- Public-Key Encryption (a.k.a Asymmetric Key Encryption)



Sorry to the people that are in CSCI 476, CSCI 466 *and* CSCI 460

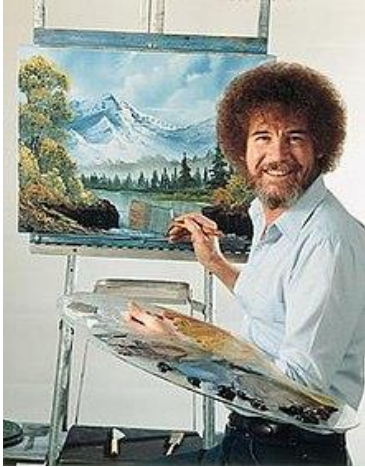
- The protection of information and information systems

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” →

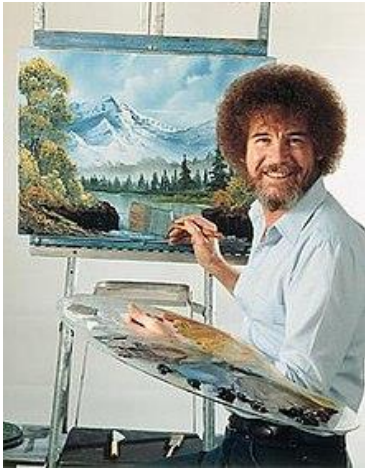


Over a wire, wirelessly, via a Pidgeon etc

Alice



Bob

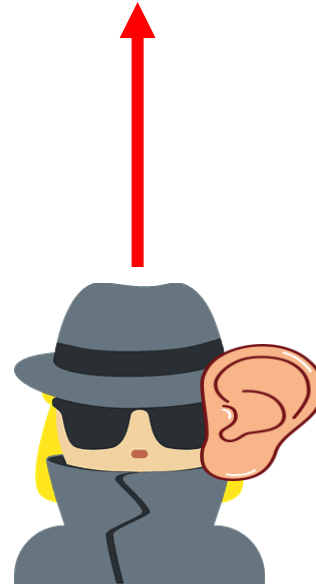


“Hi Alice, my address is
123 Painting Avenue.
Please stop by at 6:00”

Alice

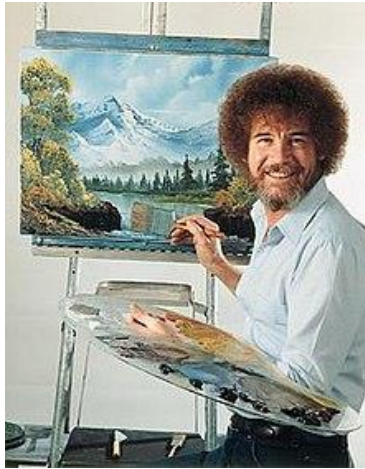


Because our transmission medium is
shared, there is a possible someone
else could be eavesdropping



Eve

Bob

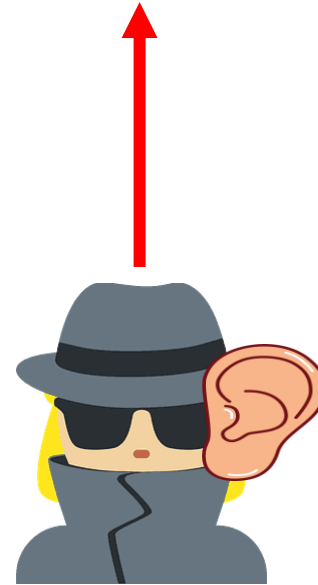


“Hi Alice, my address is
123 Painting Avenue.
Please stop by at 6:00”

Alice



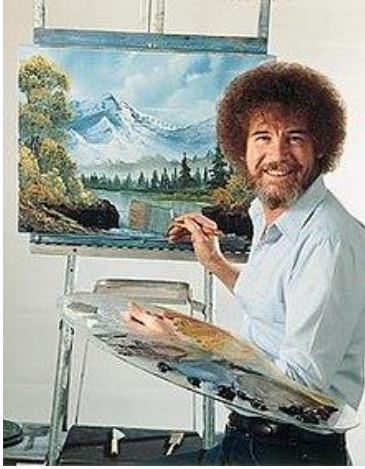
Because our transmission medium is
shared, there is a possible someone
else could be eavesdropping



Eve

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

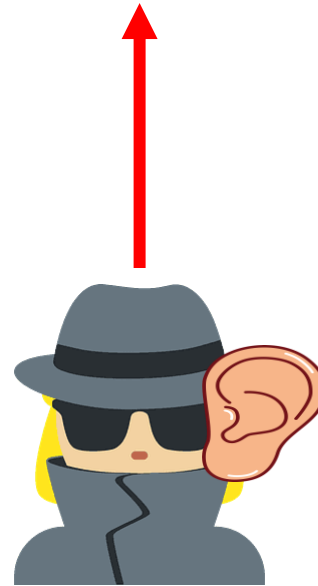
Bob



Cleartext/Plaintext

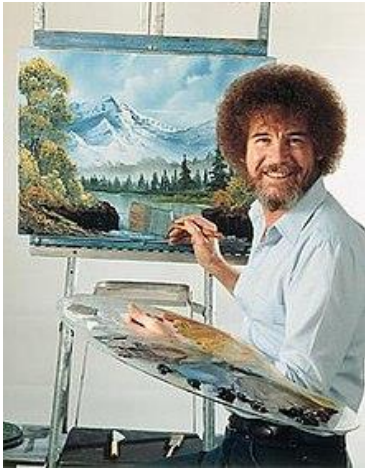
“Hi Alice, my address is
123 Painting Avenue.
Please stop by at 6:00”

Alice



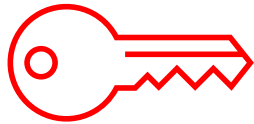
Eve

Bob



Cleartext/Plaintext

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

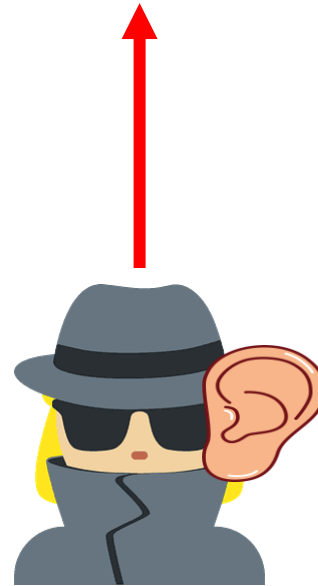


Bob **encrypts** his message with a **key**

MuYGoP5LiTTGPVX6U/r2VTpxPSqT
Fmy5nsoFWURThKMhHk/7tbjYsS2EJ
917q7megTAcV+V4ZMU4HjJjiW2DC
BroxvJ0V3ZYDgZ8B9IUvGUmdiRMH
25Xkf7QrhAGR3FF

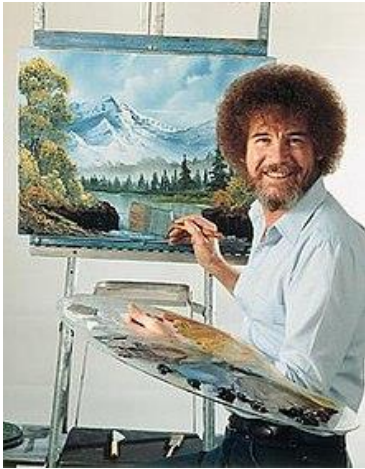
The result is a
ciphertext

Alice



Eve

Bob



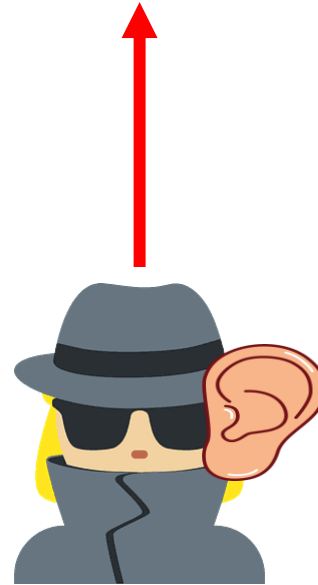
MuYGoP5LiTTGPVX6U/r2VTpxPSqTFmy5nsoF
WURThKMhHk/7tbjYsS2EJ917q7megTAcV+V4Z
MU4HjJjiW2DCBroxvJ0V3ZYDgZ8B9IUvGUmdiR
MH25Xkf7QrhAGR3FF

Alice



Cleartext/Plaintext

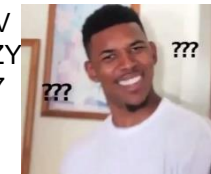
“Hi Alice, my address is
123 Painting Avenue.
Please stop by at 6:00”



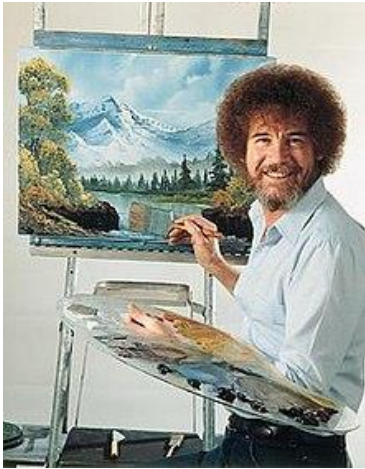
Eve

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

MuYGoP5LiTTGPVX6U/r2VTpx
PSqTFmy5nsoFWURThKMhHk
/7tbjYsS2EJ917q7megTAcV+V
4ZMU4HjJjiW2DCBroxvJ0V3ZY
DgZ8B9IUvGUmdiRMH25Xkf7
QrhAGR3FF



Bob



Cleartext/Plaintext

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



Eve

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

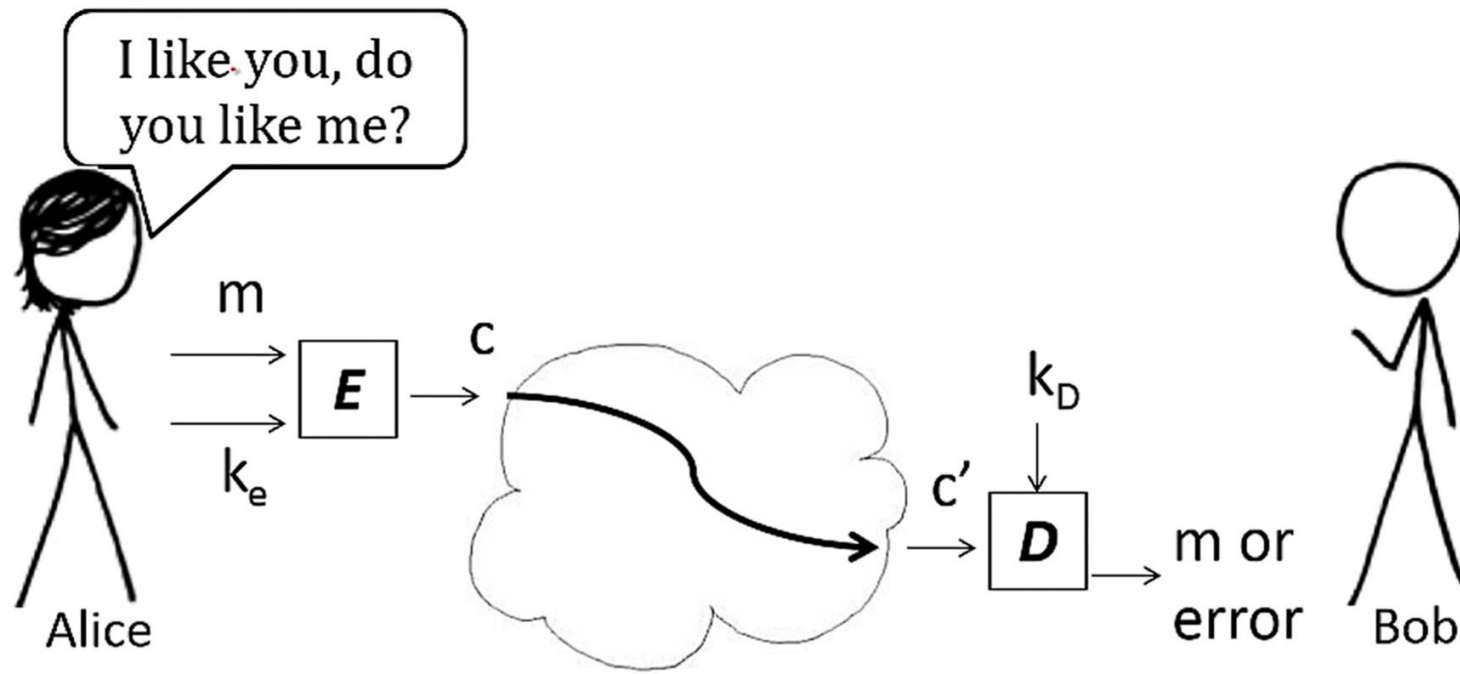
Alice



MuYGoP5LiTTGPVX6U/r2
VTpxPSqTFmy5nsoFWUR
ThKMhHk/7tbjYsS2EJ917
q7megTAcV+V4ZMU4HjJji
W2DCBroxvJ0V3ZYDgZ8
B9IUvGUmdiRMH25Xkf7
QrhAGR3FF



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



Cryptosystem

m : Plaintext

k_e : Encryption Key

k_d : Decryption Key

c : Ciphertext

E : Encryption Program

D : Decryption Program

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)

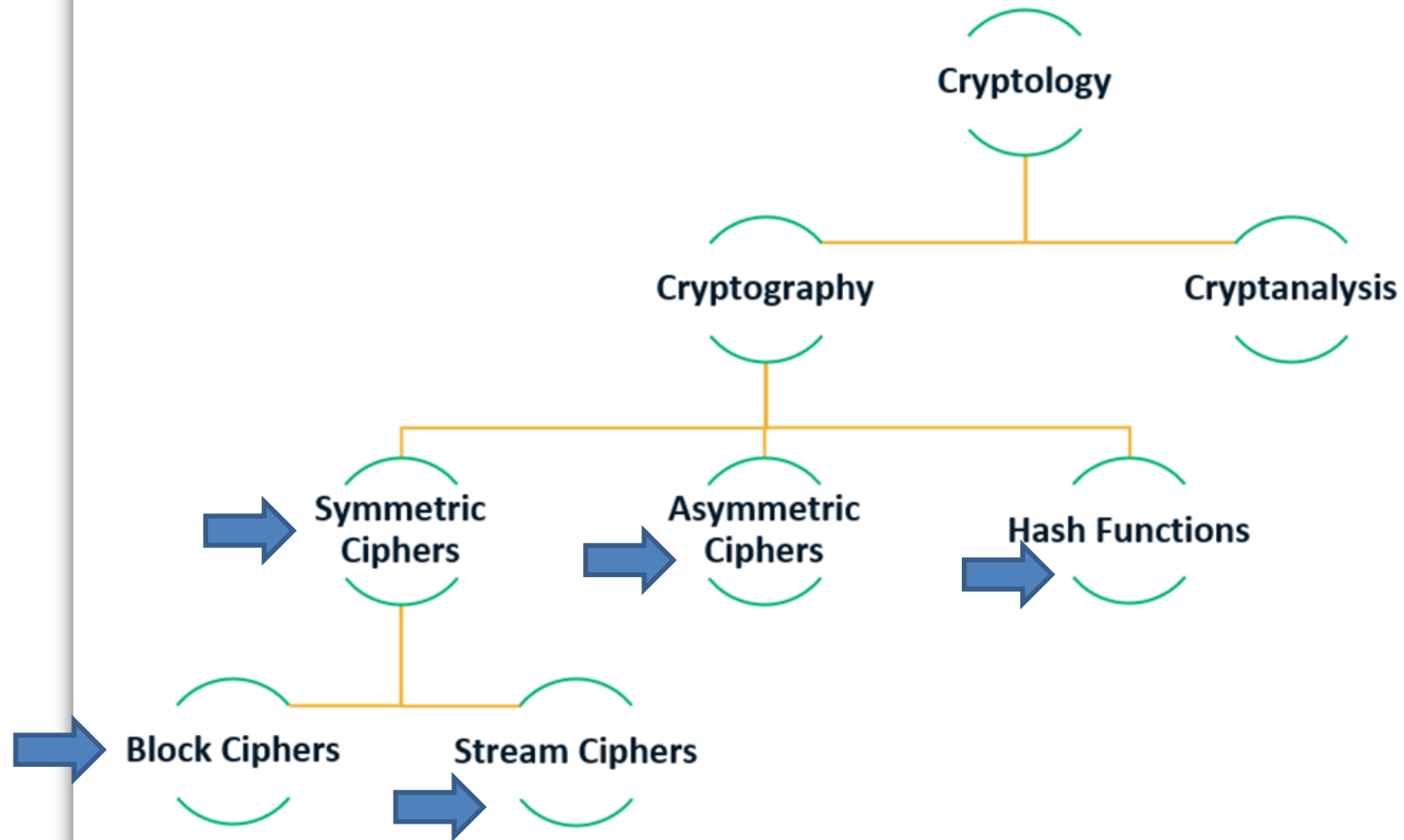
The encryption algorithms are typically rooted in **very difficult problems** in computing (ie there does not exist a program that can efficiently break RSA **YET**)

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

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

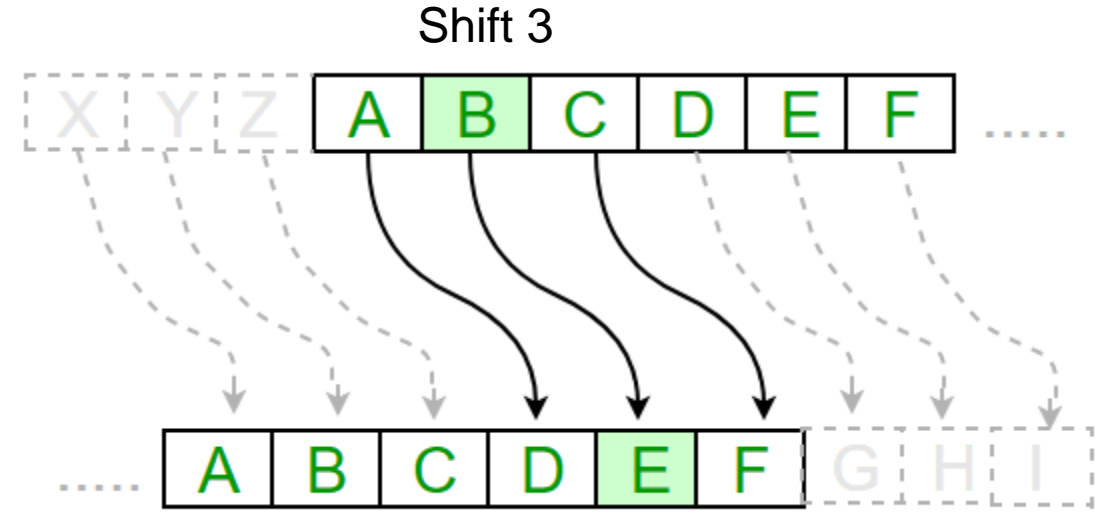
hello there world my name is reese



khoor wkhuh zruog pb
qdph 1v uhhvh

plaintext

ciphertext



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

open alphabet
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
KEYWORD A B C F G H I J L M N P Q S T U V X Z
cipher alphabet

keyword: KEYWORD
plain text: A L K I N D I
ciphertext: K

Here is a ciphertext (cipher.txt)

```
ydq ufyiqoobxrk lrcqx yqoy fo r kwgyfoyrbq rxxepfc crlrcfyt yqoy ydry lxebxqoofvqgt bqyo kexq  
mfuufcwgy ro fy ceiyfiwqo. ydq ysqiylt 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 dqr x ydfo ofbirg pql r  
ofibgq grl odewgm pq ceklgqyqm qrcd yfkq tew dqr x ydfo oewim. [mfib] xqkqkpqx ye xwi fi r oyxrfdy  
gfiq, rim xwi ro geib ro leoofpgq. ydq oqceim yfkq tew urfg ye ceklgqyq r grl pquexq ydq oewim, tew  
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 ysqiylt kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq  
oyrxy. ydq xwiifib olqqm oyrxyo ogesgt, pw y bqyo uroyqx qrcd kfiwyq ruyqx tew dqr x ydfo ofbirg pql r  
ofibgq grl odewgm pq ceklgqyqm qrcd yfkq tew dqr x 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, tew  
yqoy fo evqx. ydq yqoy sfgg pqbfi ei ydq sexm oyrxy. ei tewx krxj, bq 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

```

ydg ufyiqaobxrk lrcqx yqoy fo r kwgyfoyrbq rqxepfc crlrcfyt yqoy ydry lxebxqoofvqgt bqyo kexq
mfuufcwgy ro fy ceiyfiwqo. ydq ysaiyt kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq
oyrxy. ydq xwiifib olqam oyrxyo ogesgt, pwy bqyo uroyqx qrcd kfiwyq ruyqx tew dqrq ydfo ofbirg pqql r
ofibgq grl odewgm pq ceklgqym qrcd yfkq tew dqrq ydfo oewim. [mfib] xqkqkpx ye xwi fi r oyxfbdy
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 krxi, 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:~/encryption_lecture$ ./freq.py < cipher.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

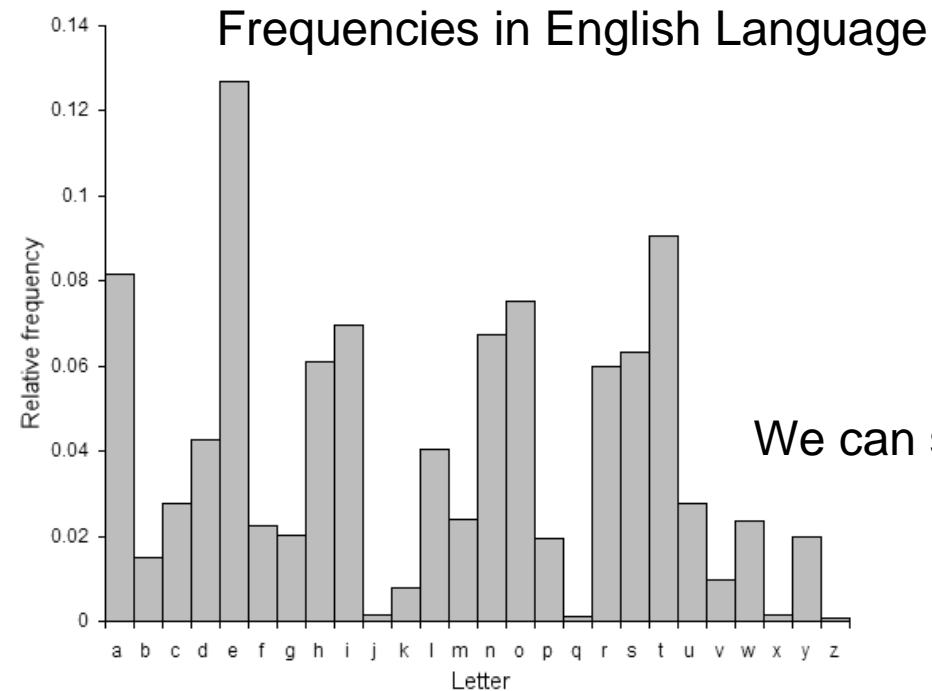
```

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

```



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.

```

ydg ufyiqaobxrk lrcqx yqoy fo r kwgyfoyrbq rqxepfc clrlcft yqoy ydry lxbxqoofvqgt bqyo kexq
mfuufcwgy ro fy ceiyfiwqo. ydq ysqiyl kqyqx lrcqx yqoy sfgg pqbfi fi ydfxyt oqceimo. gfiq wl ry ydq
oyrxy. ydq xwiifib olqam oyrxyo ogesgt, pw y bqyo uroyqx qrcd kfiwyq ruyqx tew dqr x ydfo ofbirg pqql r
ofibgq grl odewgm pq ceklgqym qrcd yfkq tew dqr x ydfo oewim. [mfib] xqkqkpx ye xwi fi r oyxfbdy
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 krxi, 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:~/encryption_lecture$ ./freq.py < cipher.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

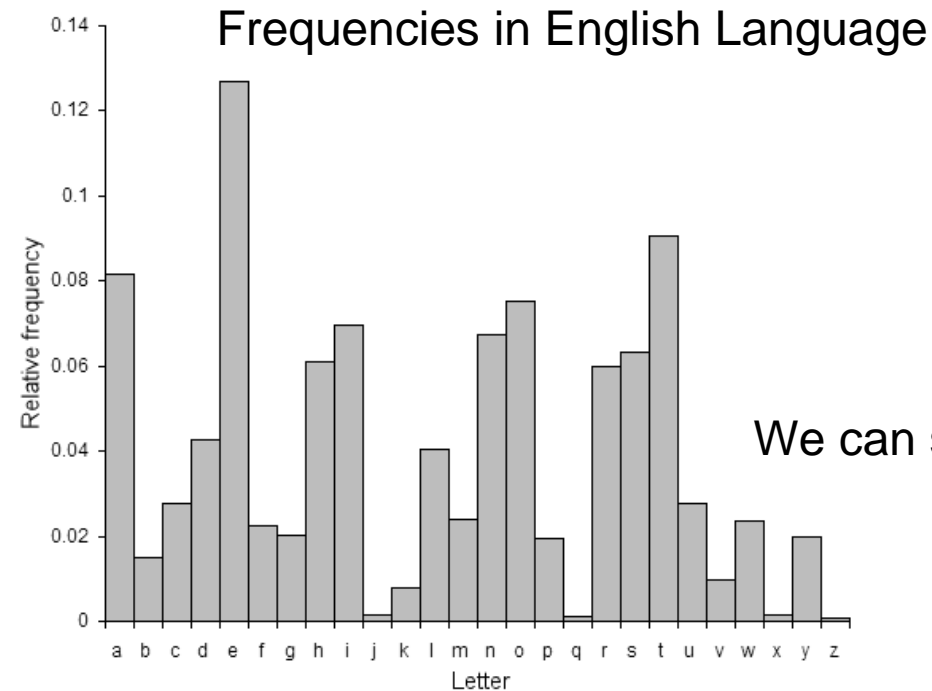
```

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

```



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.

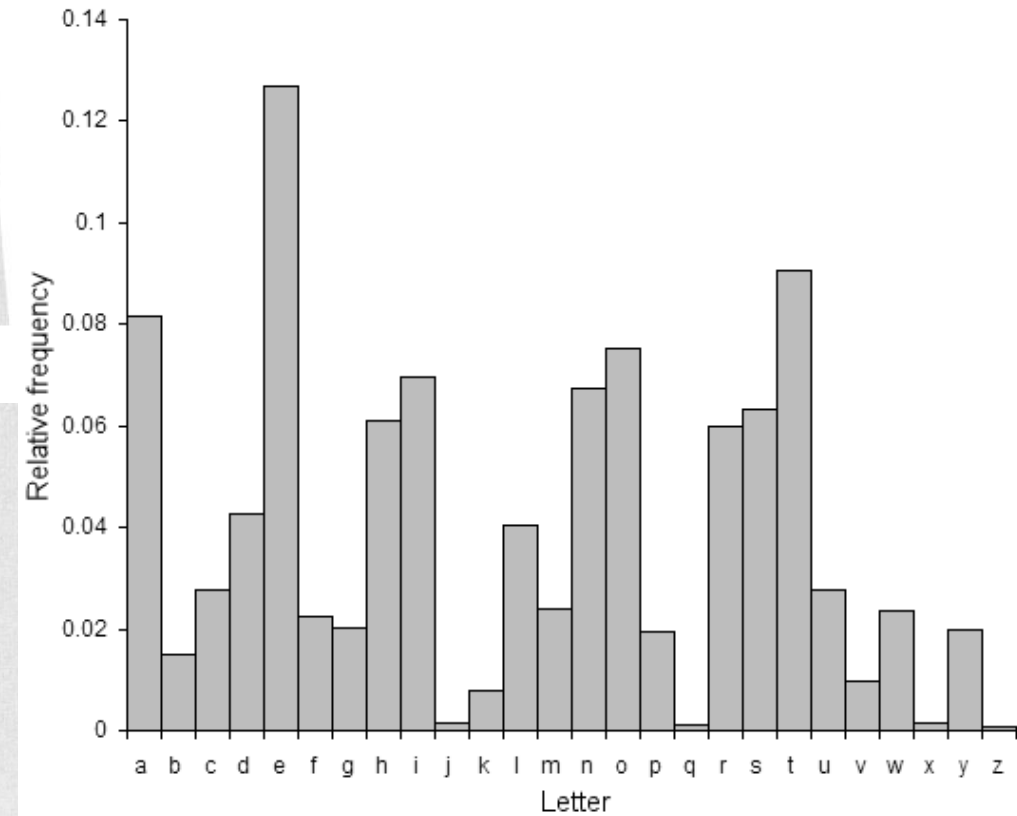
Listing 24.2: Bigram and trigram frequencies

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 rxepfc 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 qrch kfiwtq rutqx tew dqrq tdfo ofbirg pqq l r ofibgg grl odewgm pq ceklggtqm qrch tfkq t
ew dqrq tdfo oewim. [mfib] xqkqkpqx te xwi fi r otxrfbdt gfiq, rim xwi ro geib ro leoofpgg. tdq oqceim tfkq tew urfg te ceklggtq 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 rxepfc 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 hqrq thfo ofbirg pqq l r ofibgg grl ohewgm pq ceklggtqm qrch tfkq t
ew hqrq thfo oewim. [mfib] xqkqkpqx te xwi fi r otxrfbht gfiq, rim xwi ro geib ro leoofpgg. thq oqceim tfkq tew urfg te ceklggtq 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
101010101010001010101010101
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:

\oplus 0001 1010 0011
1100 1100 0101

Key:

Ciphertext:

1101 0110 0110

How to get original message?

Review the XOR operator:

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



0101010101010010111101010100
1000010111001000101010101100
1010101010111110100100101010
1010010101010110010101011010
1001010101010101010101001010
101010101010001010101010101
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:

\oplus 0001 1010 0011
1100 1100 0101

Key:

Ciphertext:

\oplus 1101 0110 0110
1100 1100 0101

XOR with the
key again!

0001 1010 0011

Block Cipher

Split in messages into fixed sized blocks, encrypt each 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

\oplus

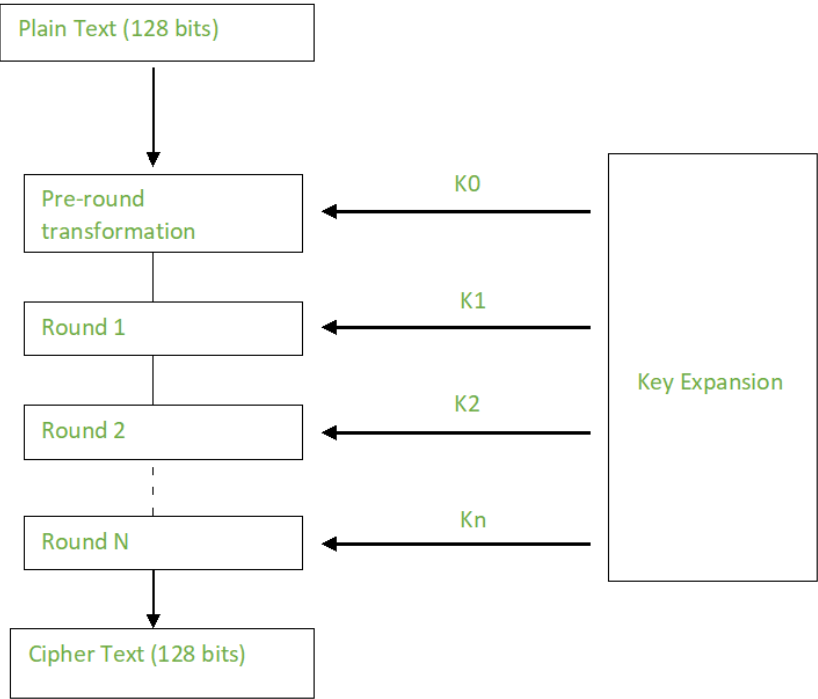
\oplus

\oplus



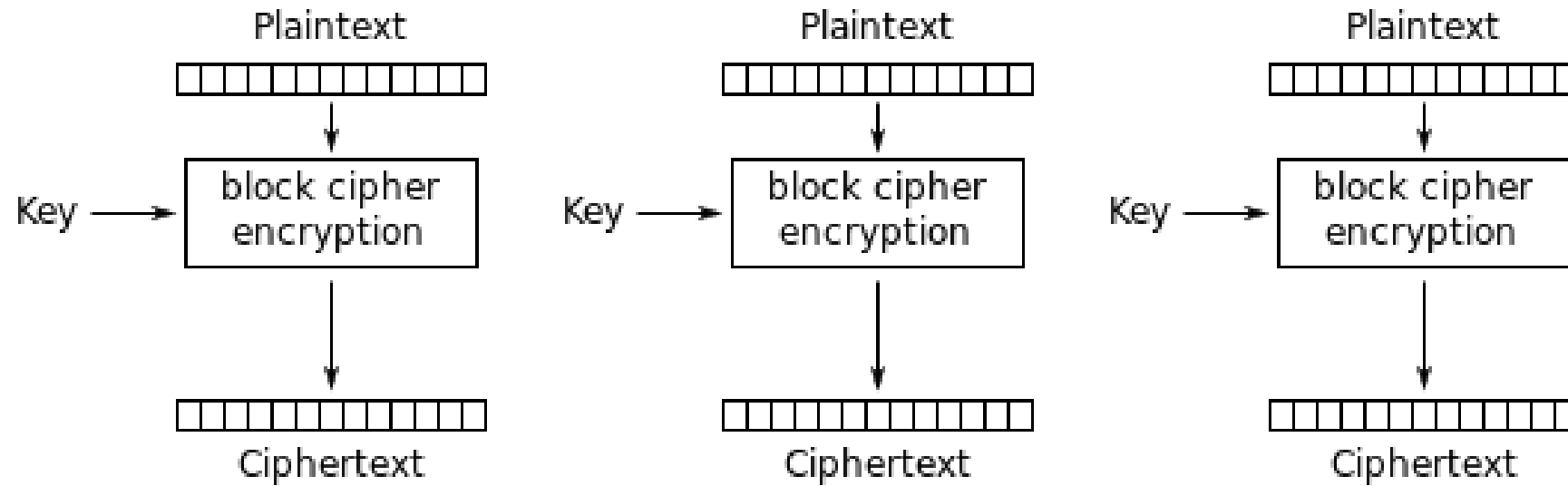
Ciphertext

Example AES



Three possible key sizes, 128, 192, 256

ECB



Electronic Codebook (ECB) mode encryption