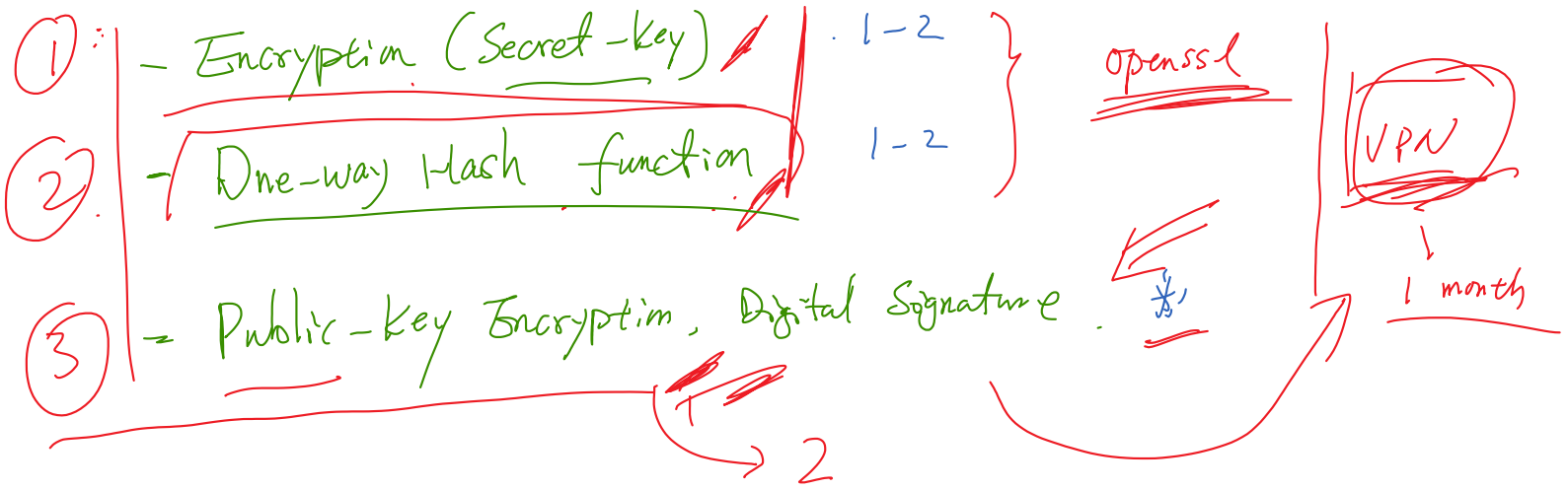
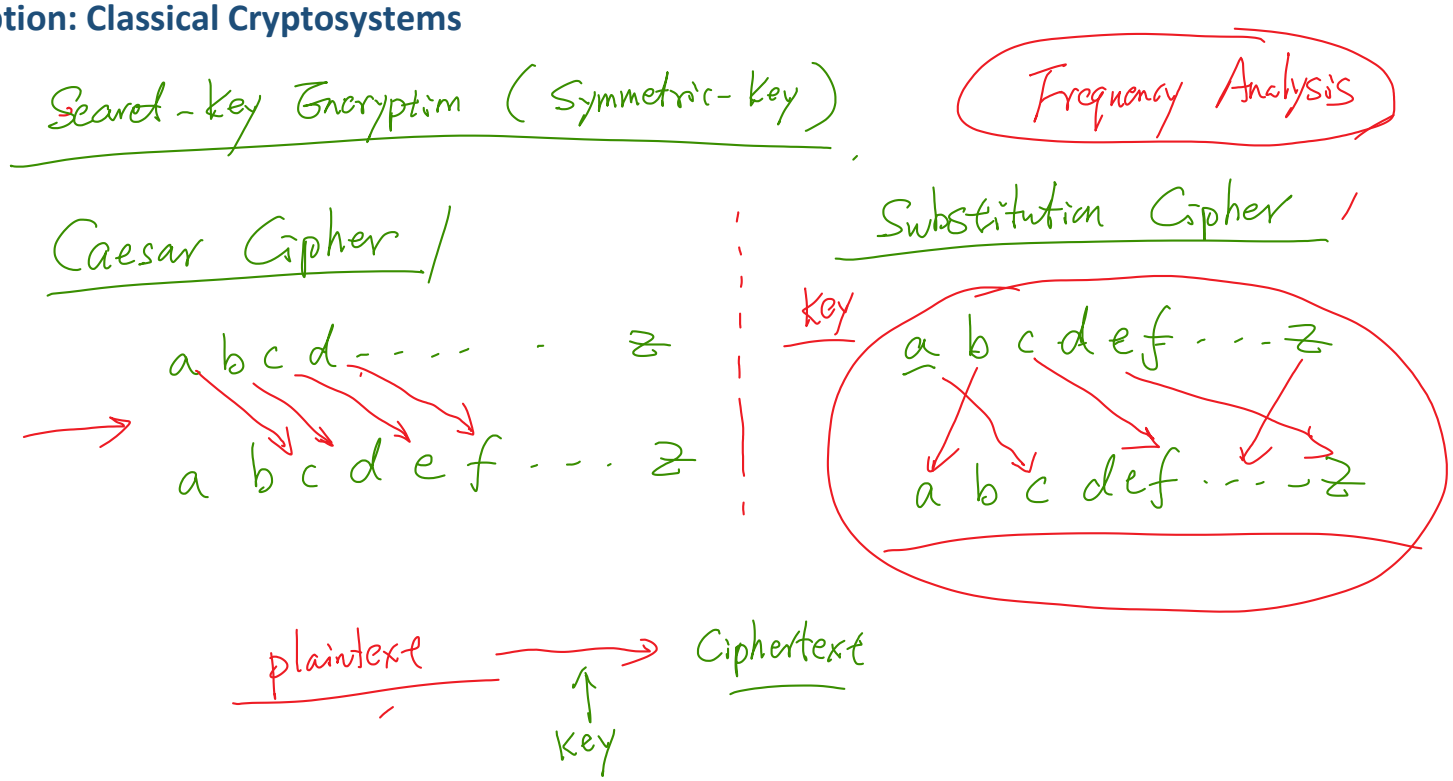
Internet Security

Secret-Key Encryption

Introduction to Cryptography

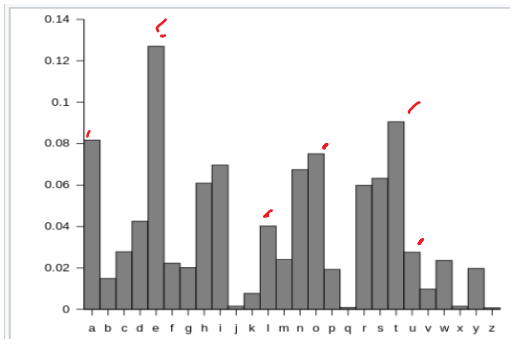


Encryption: Classical Cryptosystems



Frequency Analysis (Monoalphabetic)

```
ytn xqavhq yzhu  xu qzupvd ltmat qnncq vgxzy hmrty vbynh ytmq ixur qyhvurn
vlvhpq yhme ytn gvrrnh bnniq imsn v uxuvrnuvhmvu yxx

ytn vlvhpq hvan lvq gxxsnupnp gd ytn pncmqn xb tvhfnd lnmuqynmu vy myq xzyqny
vup ytn veevhnuy mceixqmxu xb tmq bmic axcevud vy ytn nup vup my lvq qtvenp gd
ytn ncnhrnuan xb cnyxx ymcnq ze givasrxlu eximymaq vhcavupd vaymfmqc vup
v uvymxuvi axufnhqvymxu vq ghmnb vup cvp vq v bnfnh phnvc vgxzy ltnytnh ytnhn
xzrty yx gn v ehnqmpnuy lmubhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
nkyhv ixur gnavzqn ytn xqavhq lnhn cxfnp yx ytn bmhqy lnnsnup mu cvhat yx
vfxmp axubimaymur lmyt ytn aixqmur anhncxud xb ytn lmuynh xidcemaq ytvusq
ednxuratvur

xun gmr jznqymxu qzhhxzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
anhncxud lmii vpphnqq cnyxx nqenamviid vbynh ytn rxipnu rixgnq ltmat gnavcn
v ozgmivuy axcmurxzy evhyd bxh ymcnq ze ytn cxfncnuy qenvhtnvpnp gd
exlnhbzi txiidlxxp lxcnu ltx tnienp hvmqn cmiimxuq xb pxiivhq yx bmrty qnkzvi
tvhvqqcnuy vhxzup ytn axzuyhd

qmruvimur ytnmh qzeexhy rxipnu rixgnq vyynupnnq qlvytnp ytncqnifnq mu givas
qexhynp iveni emuq vup qxzupnp xbb vgxzy qnkmqy exlnh mcgvivuanq bhxc ytn hnp
avheny vup ytn qyvrn xu ytn vmh n lvq aviinp xzy vgxzy evd munjzmyd vbynh
myq bxhcnh vuatxh avyy qvpinh jzmy xuan qtn invhunp ytvy qtn lvq cvsmur bvh
inqq ytvu v cvin axtxqy vup pzhmur ytn anhncxud uvyvimn exhycvu yxxs v gizuy
vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
ytvy gn yxeenp
```




```
n : 488
y : 373
v : 348
x : 291
u : 280
q : 276
m : 264
h : 235
t : 183
i : 166
p : 156
a : 116
c : 104
z : 95
l : 90
g : 83
b : 83
r : 82
e : 76
d : 59
f : 49
s : 19
j : 5
k : 5
o : 4
w : 1
```

```
yt => 116
tn => 89
mu => 74
nh => 66
nq => 62
hn => 59
vu => 58
vh => 57
qy => 55
xu => 53
nv => 50
up => 47
yn => 47
np => 46
vy => 45
xh => 45
nu => 44
ym => 39
uy => 37
vi => 37
yx => 36
vq => 35
uv => 34
gn => 32
my => 32
av => 31
xz => 30
ur => 29
na => 29
tv => 29
qn => 28
uq => 27
mq => 27
qv => 27
lv => 26
hq => 26
nc => 26
iv => 25
hm => 24
hy => 23
py => 23
zy => 23
```

```
ytn => 79
vup => 30
nqy => 22
pyt => 20
mur => 20
ynh => 18
xzy => 16
nhn => 16
nuy => 14
ytv => 14
bxh => 14
gnq => 14
mxu => 14
vii => 13
vyn => 13
uvy => 12
lvq => 12
nvh => 12
tmq => 12
qyt => 12
muv => 11
upy => 11
xhy => 11
vym => 11
lmu => 11
ymu => 11
yxh => 11
tnv => 11
cmu => 11
hna => 10
tnh => 10
xuq => 10
myt => 10
ymx => 10
tvy => 10
vhp => 10
```

```
pytn => 14
upyt => 11
gnqy => 10
ymxu => 10
lmyt => 9
vhpq => 9
vupy => 9
ytnh => 9
ytvy => 9
muvy => 9
dytn => 8
ytng => 8
lvhp => 8
ytnv => 8
bmic => 8
ytmq => 8
vlvh => 8
xcmu => 8
cmuv => 8
mxuq => 8
yzhn => 7
yytn => 7
nqyt => 7
fxyn => 7
vymx => 7
ayxh => 7
uytn => 7
uxcm => 7
vynp => 7
mury => 7
xbyt => 7
ltma => 7
tmat => 7
```

```
th 1.52     en 0.55     ng 0.18
he 1.28     ed 0.53     of 0.16
in 0.94     to 0.52     al 0.09
er 0.94     it 0.50     de 0.09
an 0.82     ou 0.50     se 0.08
re 0.68     ea 0.47     le 0.08
nd 0.63     hi 0.46     sa 0.06
at 0.59     is 0.46     si 0.05
on 0.57     or 0.43     ar 0.04
nt 0.56     ti 0.34     ve 0.04
ha 0.56     as 0.33     ra 0.04
es 0.56     te 0.27     ld 0.02
st 0.55     et 0.19     ur 0.02
```

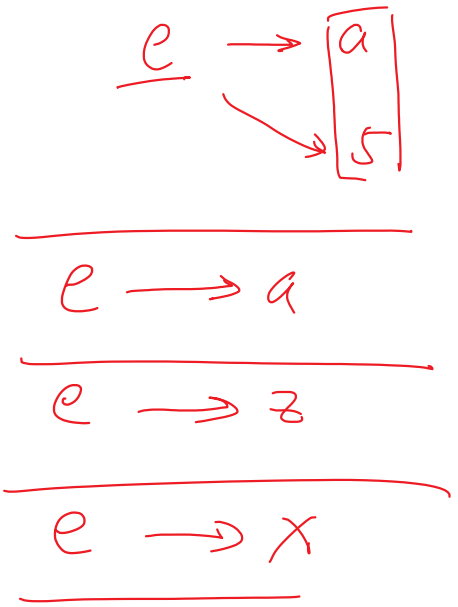
Rank[1]	Trigram	Frequency[3] (Different source)
1	the	1.81%
2	and	0.73%
3	tha	0.33%
4	ent	0.42%
5	ing	0.72%
6	ion	0.42%
7	tio	0.31%
8	for	0.34%
9	nde	
10	has	
11	nce	
12	edt	

Worksheet

```
n : 488
y : 373
v : 348
x : 291
u : 280
q : 276
m : 264
h : 235
t : 183
i : 166
p : 156
a : 116
c : 104
z : 95
l : 90
g : 83
b : 83
r : 82
e : 76
d : 59
f : 49
s : 19
j : 5
k : 5
o : 4
w : 1
```

```
yt => 116
tn => 89
mu => 74
nh => 66
nq => 62
hn => 59
vu => 58
vh => 57
qy => 55
xu => 53
nv => 50
up => 47
yn => 47
np => 46
vy => 45
xh => 45
nu => 44
ym => 39
uy => 37
vi => 37
yx => 36
vq => 35
uv => 34
gn => 32
my => 32
av => 31
xz => 30
ur => 29
na => 29
tv => 29
qn => 28
uq => 27
mq => 27
qv => 27
lv => 26
hq => 26
nc => 26
iv => 25
hm => 24
hy => 23
py => 23
zy => 23
```

```
ytn => 79
vup => 30
nqy => 22
pyt => 20
mur => 20
ynh => 18
xzy => 16
nhn => 16
nuy => 14
ytv => 14
bxh => 14
gnq => 14
mxu => 14
vii => 13
vyn => 13
uvy => 12
lvq => 12
nvh => 12
tmq => 12
qyt => 12
muv => 11
upy => 11
xhy => 11
vym => 11
lmu => 11
ymu => 11
yxh => 11
tnv => 11
cmu => 11
hna => 10
tnh => 10
xuq => 10
myt => 10
ymx => 10
tvy => 10
vhp => 10
```

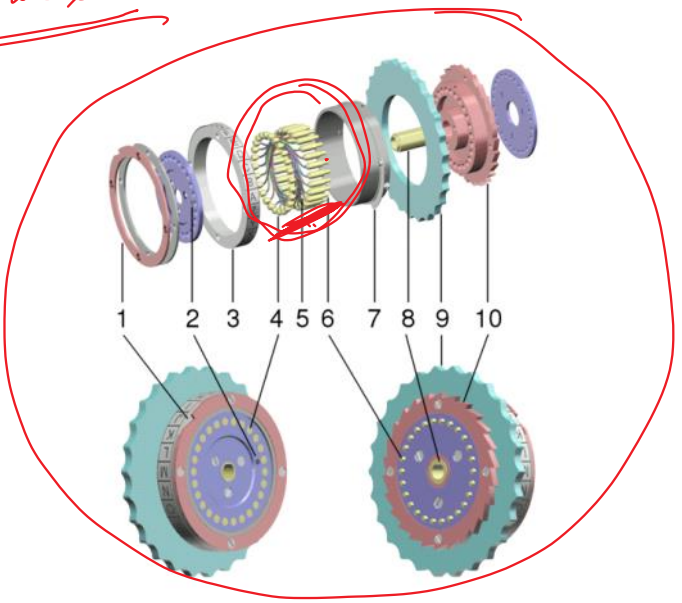


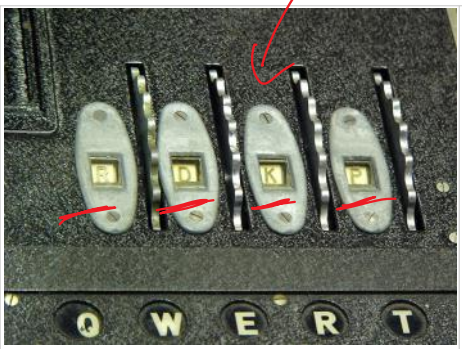
Polyalphabetic Cipher

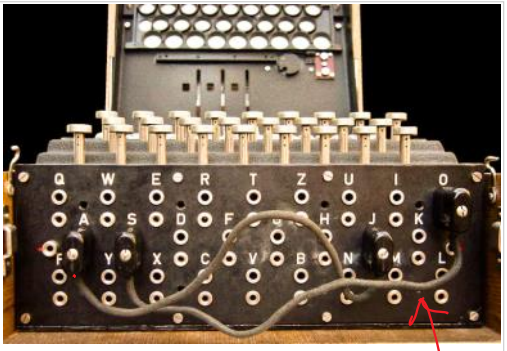
Monoalphabetic Cipher

polyalphabetic Cipher

Enigma Machine





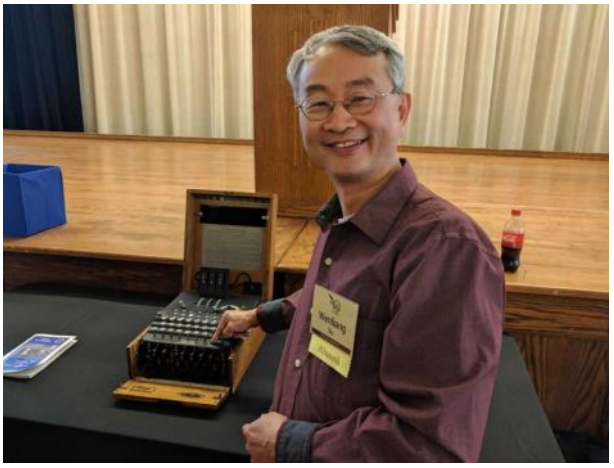


Combining the three rotors from sets of five, the rotor settings with 26 positions, and the plugboard with ten pairs of letters connected, the military Enigma has 158,962,555,217,826,360,000 (158 quintillion) different settings.[20]

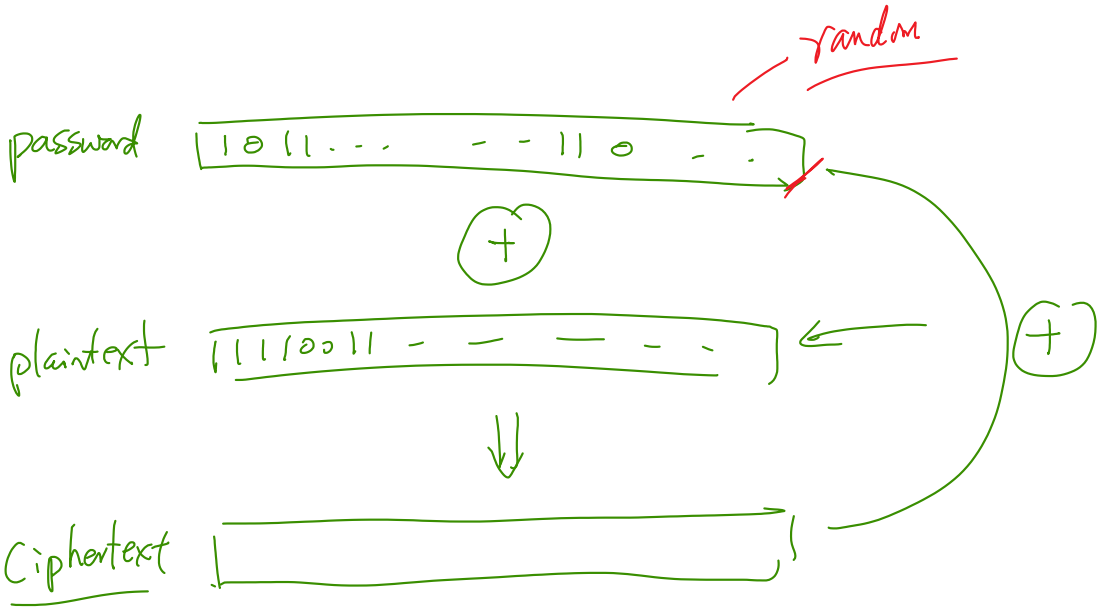
Enigma was designed to be secure even if the rotor wiring was known to an opponent, although in practice there was considerable effort to keep the wiring secret. If the wiring is secret, the total number of possible configurations has been calculated to be around 10^{114} (approximately 380 bits); with known wiring and other operational constraints, this is reduced to around 10^{23} (76 bits).[9] Users of Enigma were confident of its security because of the large number of possibilities; it was not then feasible for an adversary to even begin to try every possible configuration in a brute force attack.

A recent movie about the Enigma Machine


"BENEDICT CUMBERBATCH IS OUTSTANDING"
"A SUPERB THRILLER"
"THE BEST BRITISH FILM OF THE YEAR"
"AN INSTANT CLASSIC"
THE TRUE ENIGMA WAS THE MAN WHO CRACKED THE CODE
BENEDICT CUMBERBATCH
KEIRA KNIGHTLEY
THE IMITATION GAME
IN CINEMAS NOVEMBER 14



One-Time Pad



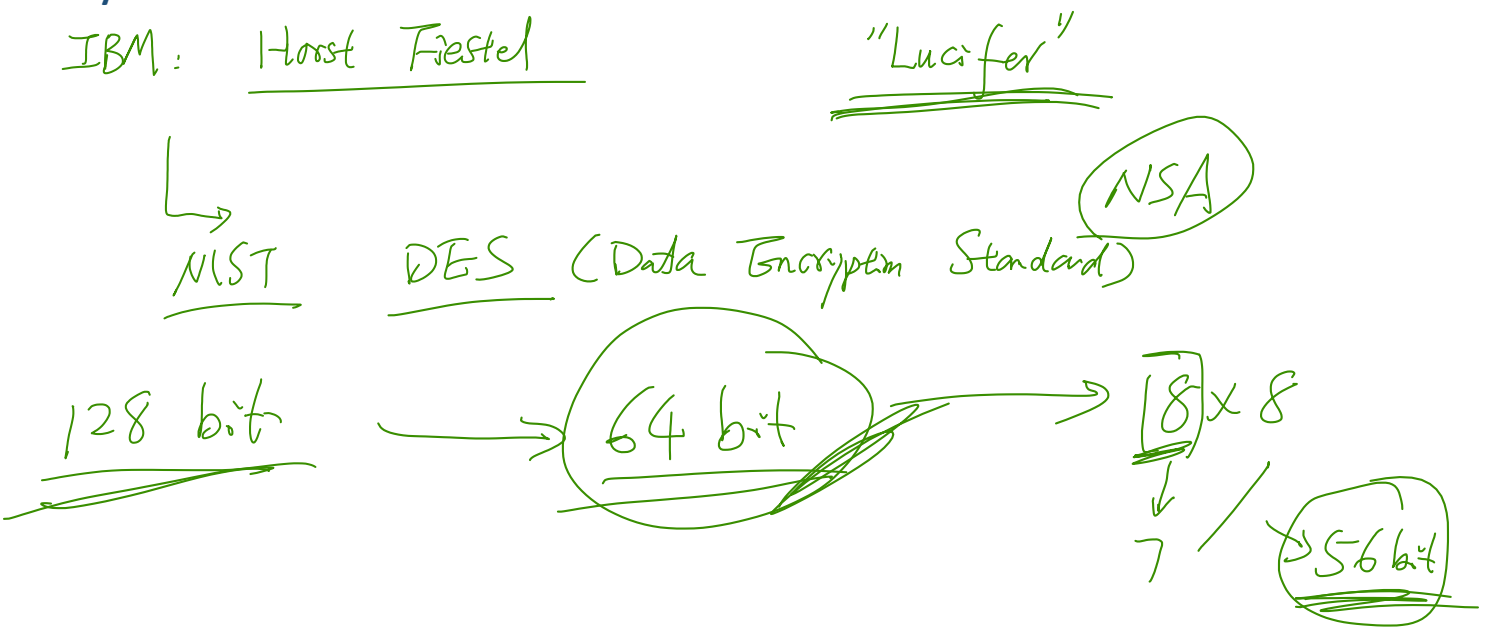
$1 \oplus 1 = 0$

$1 \oplus 0 = 1$

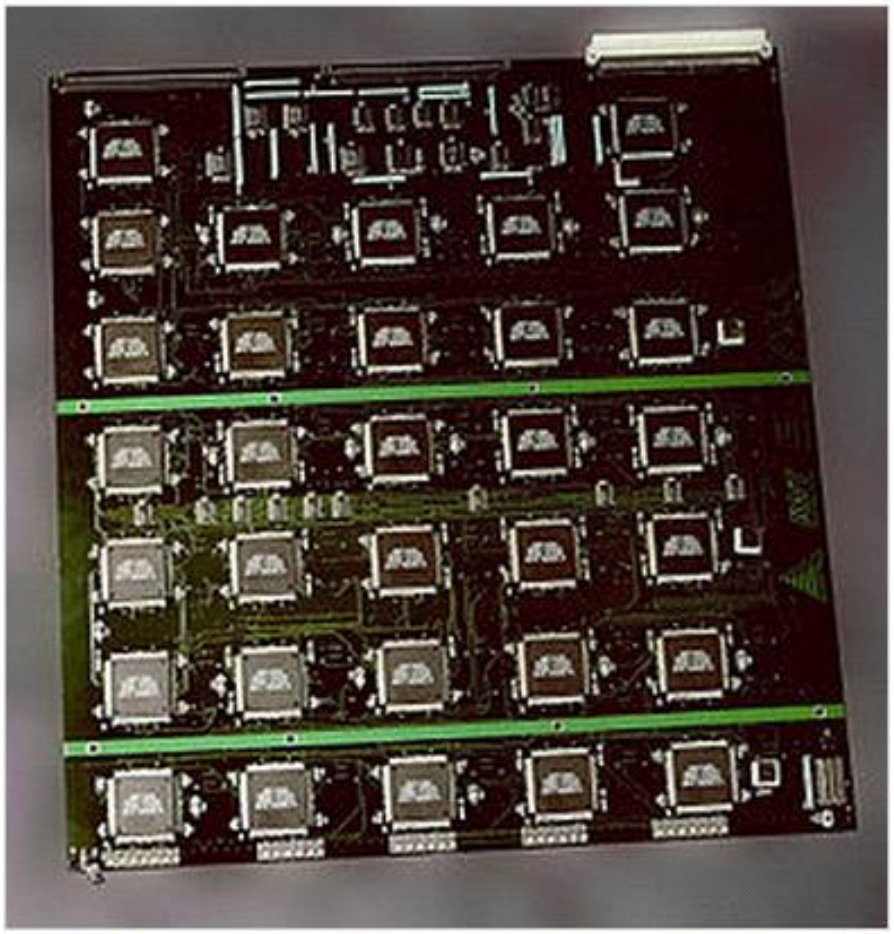
$0 \oplus 1 = 1$

$0 \oplus 0 = 0$

DES: History

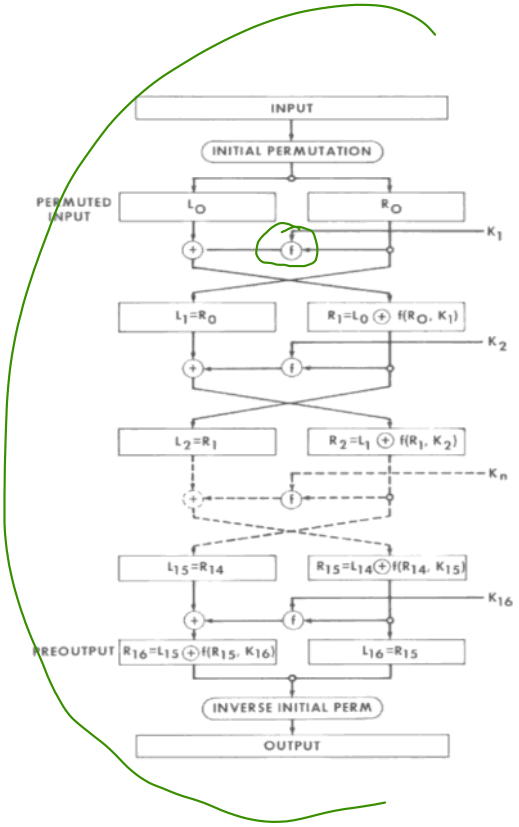

IBM: Horst Fiestel
"Lucifer"
NSA
NIST
DES (Data Encryptim Standard)
128 bit
64 bit
8 x 8
7
56 bit

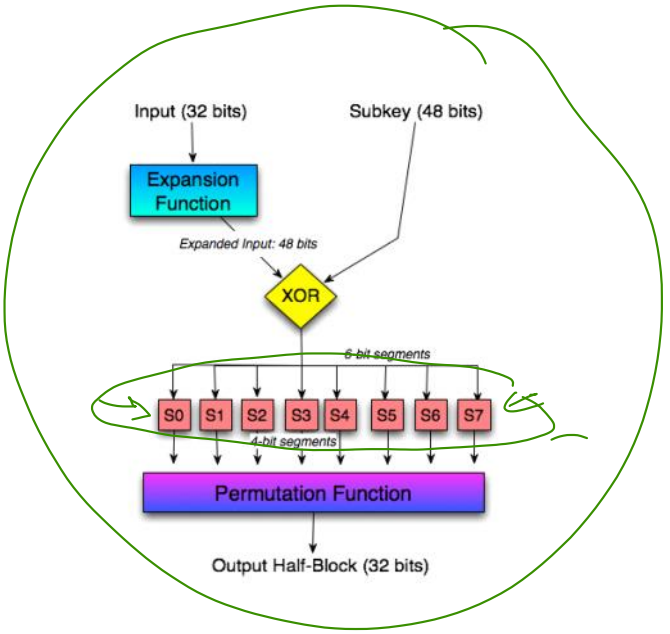
DES Cracking Machine



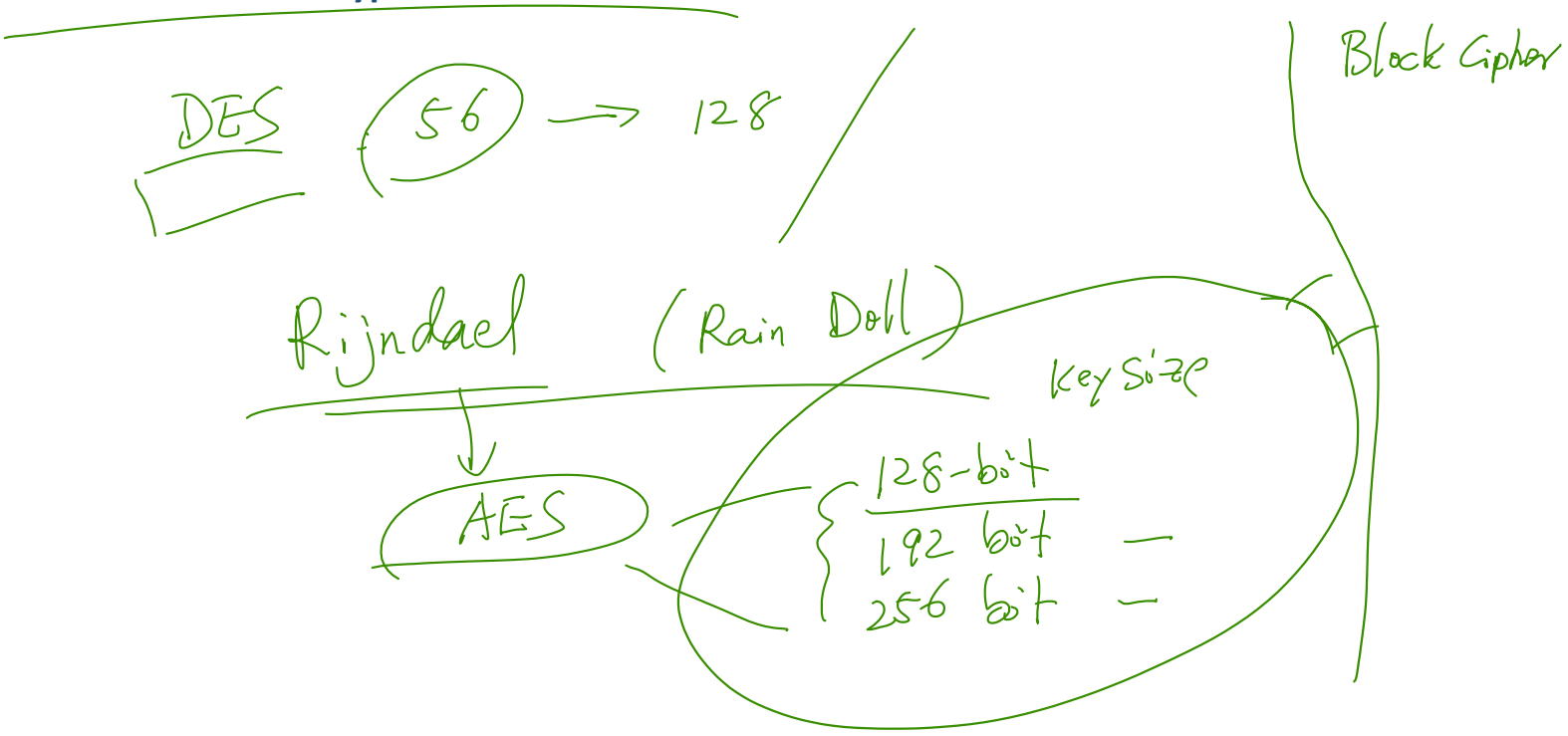
The EFF's US$250,000 DES cracking machine contained 1,856 custom chips and could brute force a DES key in a matter of days — the photo shows a two-sided DES Cracker circuit board fitted with 64 Deep Crack chips

DES Algorithm

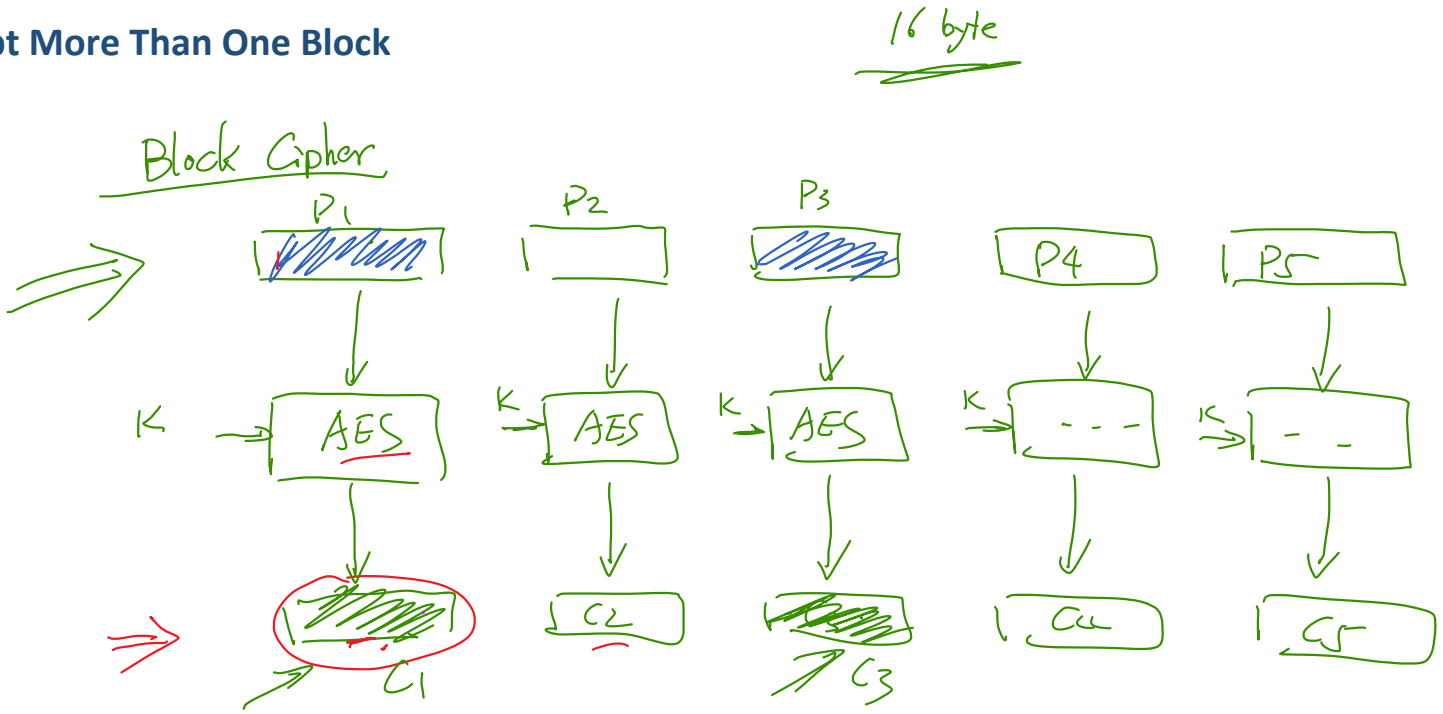




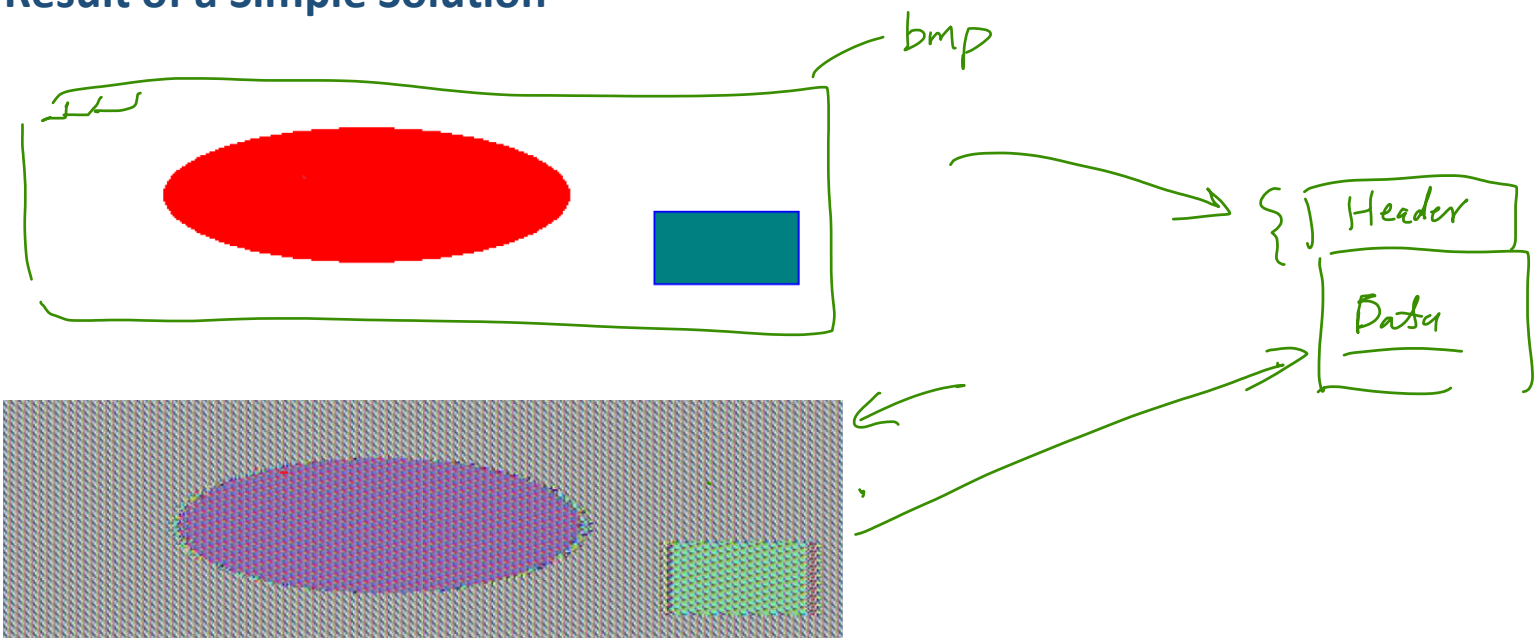
AES: Advanced Encryption Standard



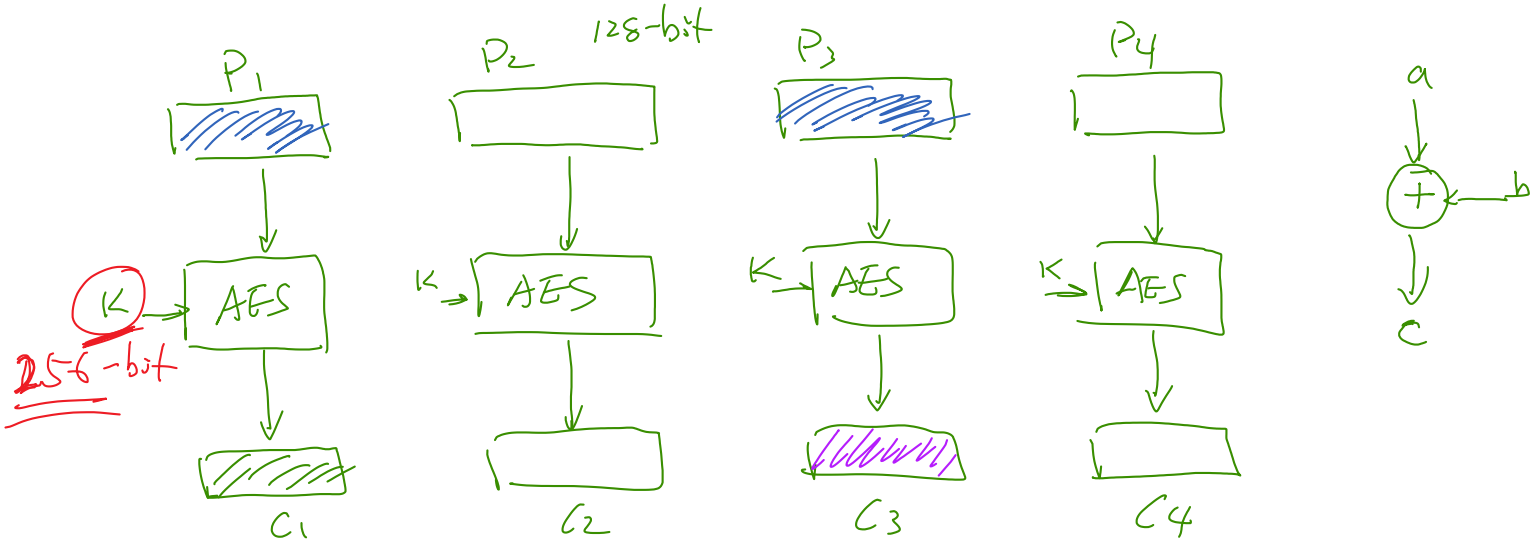
Encrypt More Than One Block



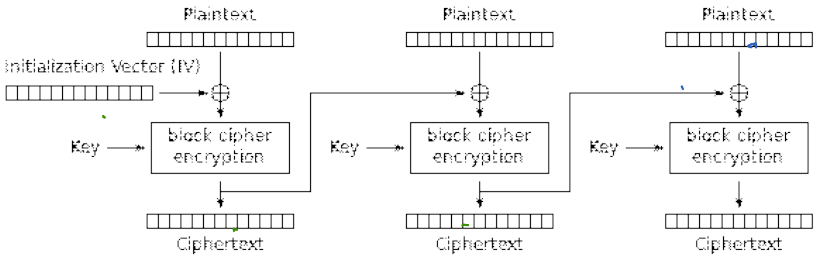
Result of a Simple Solution



Question: Given the Building Blocks, Develop a Multi-Block Encryption Mode

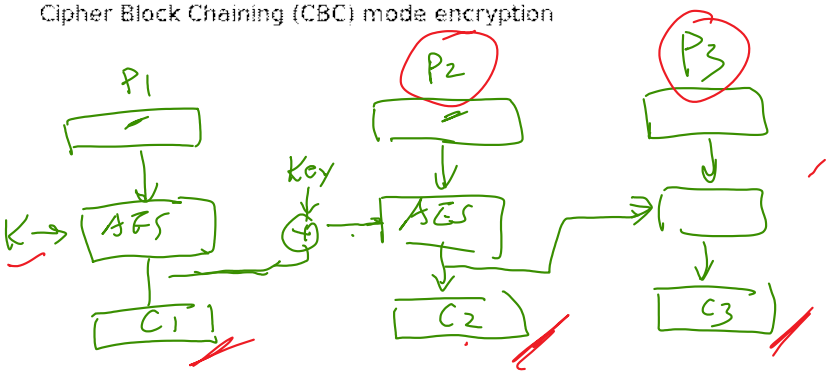


Cipher Block Chaining (CBC) Mode



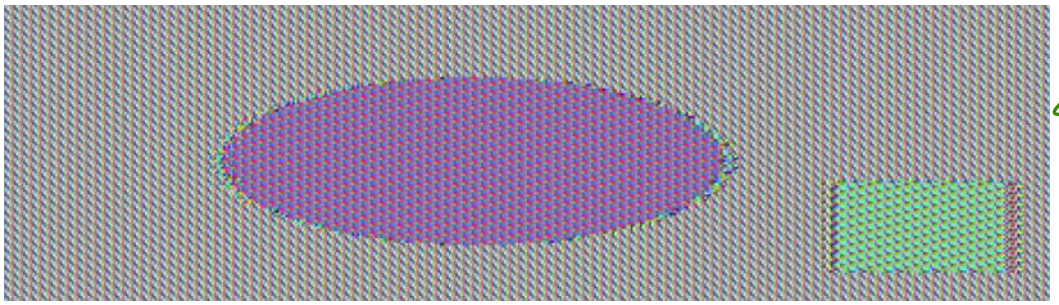
IV : randomly generated

Cipher Block Chaining (CBC) mode encryption



P1 P2 ...

ECB vs. CBC

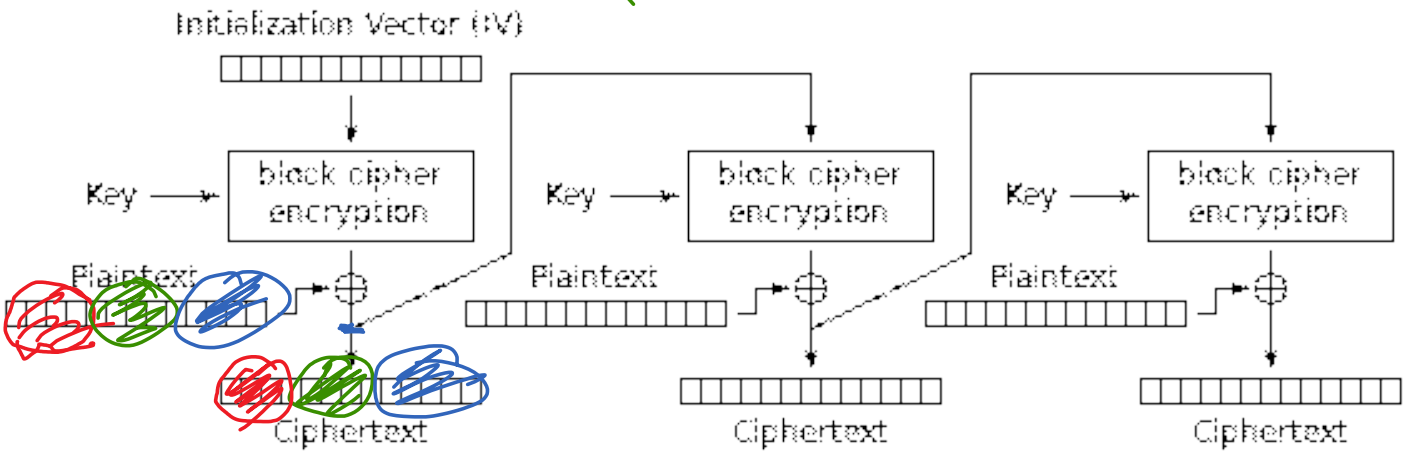


ECB

CBC

Cipher Feedback (CFB)

public

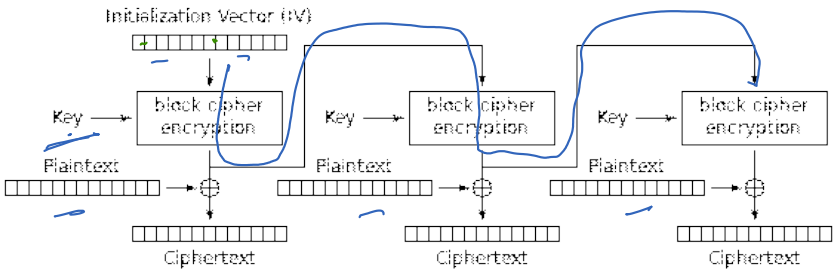


Cipher Feedback (CFB) mode encryption

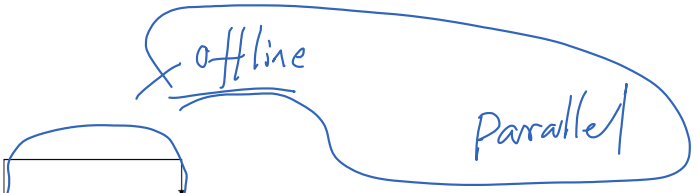
Stream Cipher :

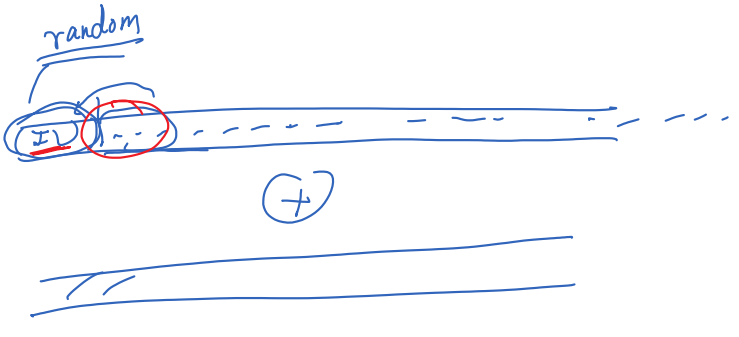
Block Cipher .

Output Feedback (OFB)

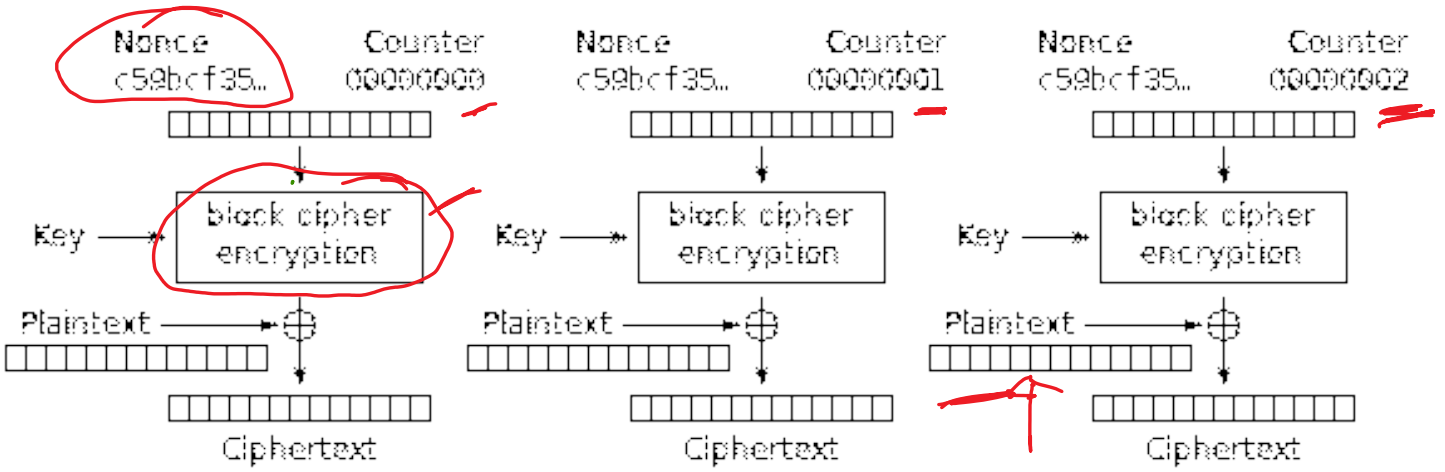


Output Feedback (OFB) mode encryption





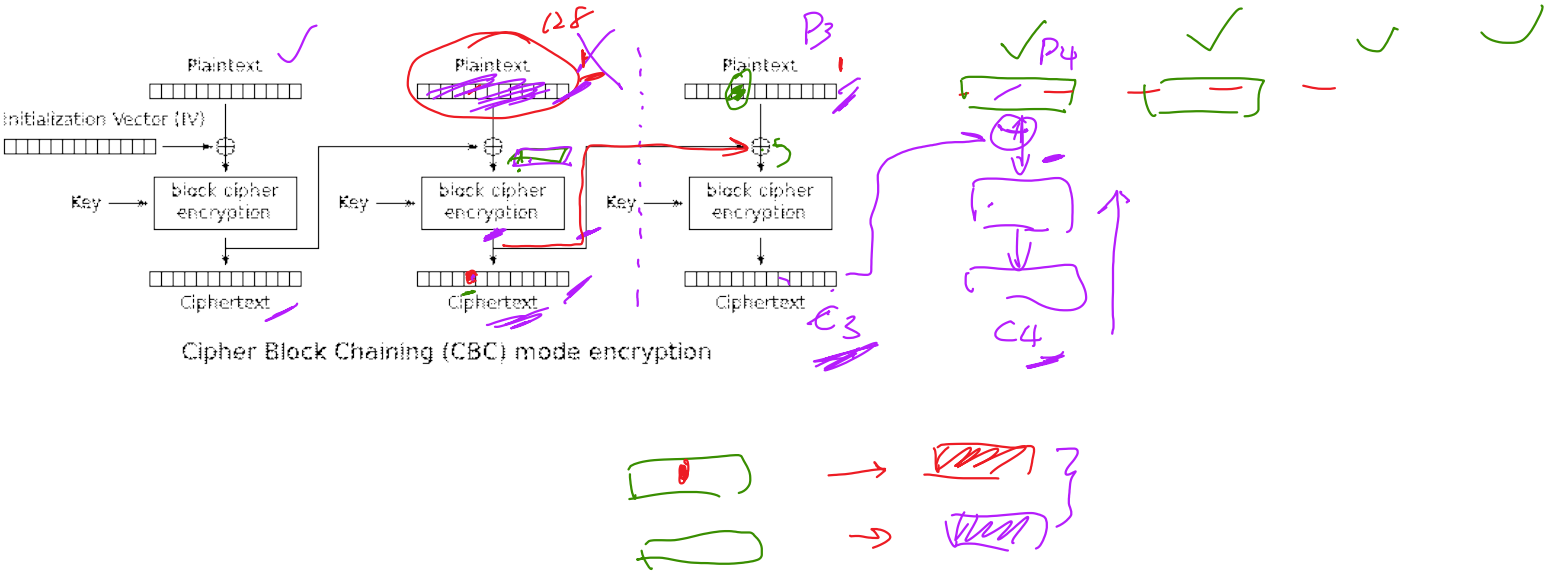
Counter Mode (CTR)



Counter (CTR) mode encryption

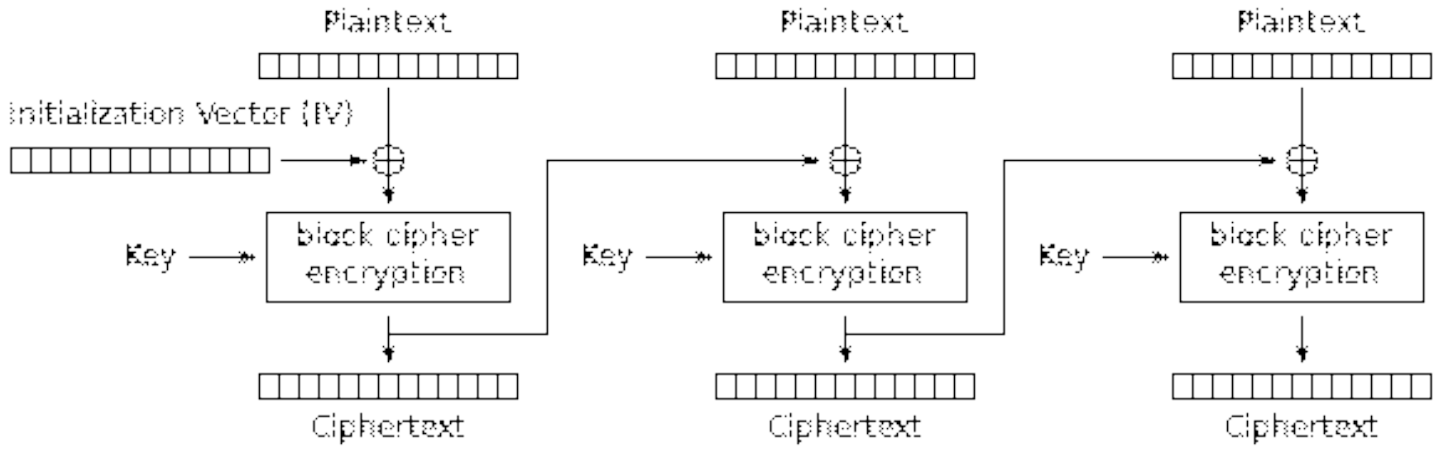
Question 1

During the transmission of the ciphertext, the fifth bit of the second block is corrupted. Without knowing that, the receiver decrypts the message. Please describe how much of the original plaintext the receiver can get. The diagram shows only 3 blocks, but assume there are 100 blocks of plaintext/ciphertext.



Question 2

IV should not be encrypted. Why?



Cipher Block Chaining (CBC) mode encryption

Padding

Padding: PKCS#5

```
Original plaintext 1:       0a23bac45092f7
Padded plaintext (PKCS#5): 0a23bac45092f70909090909090909

Original plaintext 2:       0a23bac45092f793273a7fe9093eaa88
Padded plaintext (PKCS#5): 0a23bac45092f793273a7fe9093eaa88
                           10101010101010101010101010101010
```

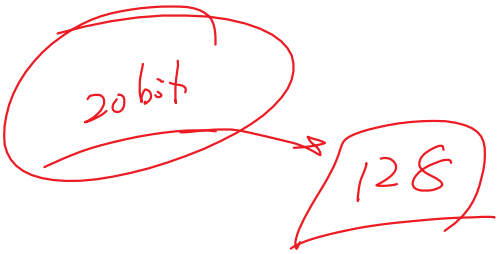
Why Do We Need Random Numbers?

IV

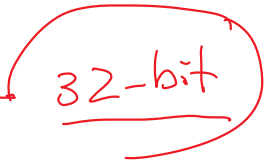
Mistake: What Is the Mistake?

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
 int c, n;

 printf("Ten random numbers in [1,100]\n");

 for (c = 1; c <= 10; c++) {
      n = rand()%100 + 1;
      printf("%d\n", n);
 }

 return 0;
}
```

Generate Random Number (Another Try)



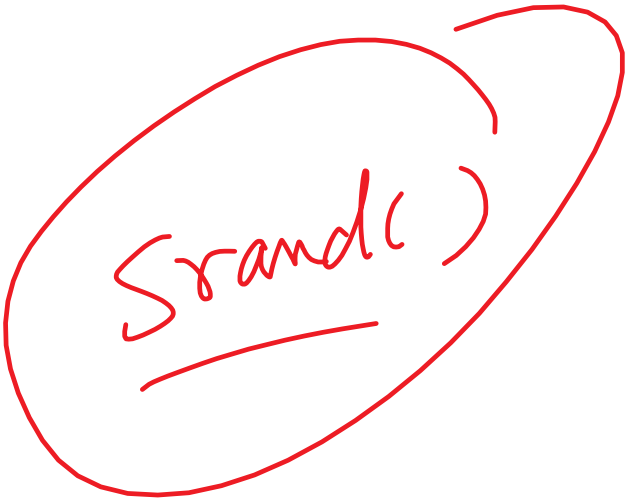

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
 int c, n;

 printf("Ten random numbers in [1,100]\n");

 srand (time(NULL));

 for (c = 1; c <= 10; c++) {
      n = rand()%100 + 1;
      printf("%d\n", n);
 }

 return 0;
}
```



Attack on the Netscape Browser in 1996

```
RNG_CreateContext()
   (seconds, microseconds) = time of day; /* Time elapsed since 1970 */
   pid = process ID;  ppid = parent process ID;
   a = mklcpr(microseconds);
   b = mklcpr(pid + seconds + (ppid << 12));
    seed = MD5(a, b);
```

Where Do We Get True Randomness?



rand()

Generate a Random 128-Bit Key

```
#define LEN 16 // 128 bits

unsigned char *key = (unsigned char *) malloc(sizeof(char)*LEN);
FILE* random = fopen("/dev/urandom", "r");
fread(key, sizeof(char)*LEN, 1, random);
fclose(random);
```

Use Special Hardware

Summary

- Classical ciphers
- DES and AES
- Encryption modes
- Random number generation