Lecture 1: Encryption

(first step towards security)

1.1: Definition: Encryption/decryption/Keys

1.2: Classical ciphers + illustration of attacks

1.3: Modern Ciphers + recommended key length

1.4: Examples of attacks on crypto

- 1.4.1: Meet-in-the-middle
- 1.4.2: Padding Oracle & notion of “Oracle”

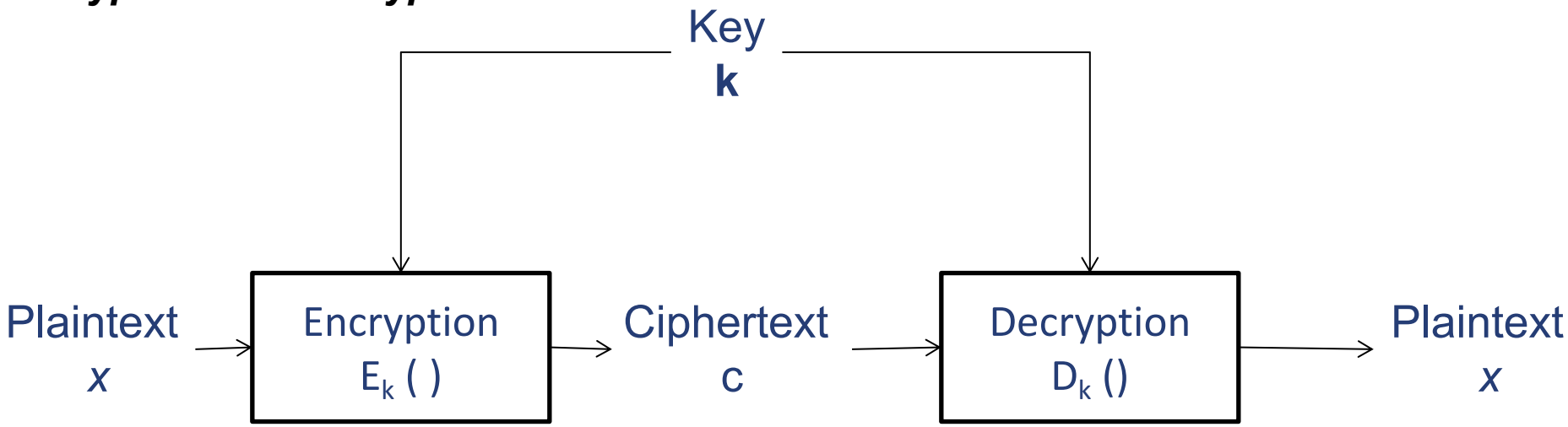
1.5: Pitfalls in usages and implementations

1.6: Interesting Historical facts

1.1 Definitions

Encryption

An *encryption scheme* (also known as *cipher*) consists of two algorithms: ***encryption*** and ***decryption.***



An encryption must meet the correctness property:

Correctness: For any plaintext x and key k,

$$D_k (E_k (x)) = x$$

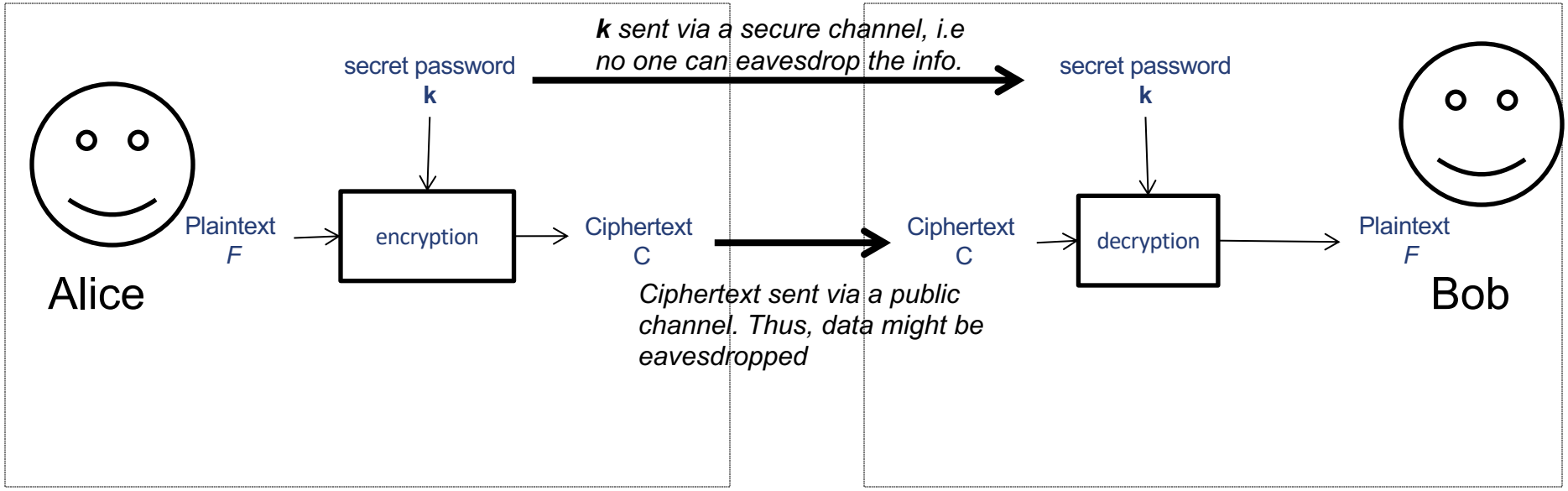
It also must be secure. Formal formulation of security is difficult (take the crypto class to learn more).

Security: Informally, from the ciphertext, it is "difficult" to derive useful information of the key k, and the plaintext x. To someone who doesn't know the key, the ciphertext resemble sequences of random bytes. (There are many refined formulations of security requirements, e.g. semantic security. In this module, we will not go into details).

An application scenario

Alice has a large file ***F*** (say info on her bank accounts and financial transactions in Excel). She "encrypted" the file ***F*** using winzip with a password "13j8d7wjnd" and obtained the ciphertext ***C.*** Next, she called Bob to tell him the 10-character password, and subsequently, she sent the ciphertext to Bob via email attachment. Later, Bob received ***C*** and decrypted the ciphertext with the password to obtain the plaintext ***F***.

Anyone who has obtained C, without knowing the password, is unable to get any information on ***F***. Although C indeed contains info of F, the information is "hidden". To someone who doesn't know the secret, C resembles a sequence of random bits.



Remark: Winzip is ***not*** an encryption scheme. It is an application that employs standard encryption schemes such as AES.

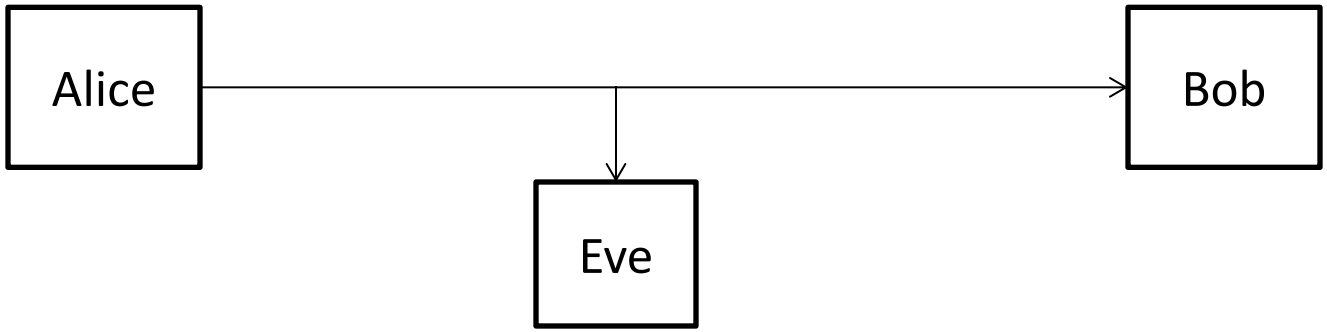
Cryptography (cryptology)

- Is the study of techniques in securing communication in the present of adversaries who have access to the communication.

- Although cryptography is commonly associated with encryption, note that encryption is just one of the primitives in cryptography (other include cryptographic hash, digital signature etc).

- Terminology: Common placeholders used in cryptography are Alice (usually the originator of message), Bob (usually the recipient), Eve (eavesdropper: can only listen), Mallory (malicious: can modify messages), (see the interesting list in https://en.wikipedia.org/wiki/Alice_and_Bob)

- Depending on context, Alice may not be a human. She could be the machine that encrypts the message.



In this module,
"see": Info that is good to know.
"read": Part of the teaching materials. Read it.
"optional": Optional information.

1.2 Classical Ciphers

For illustration, we will look into a few classical ciphers. Classical ciphers are not secure in the computer era. (exception: the "unbreakable" one-time-pad).

(see http://ciphermachines.com/index
for a good listing of classical ciphers and cipher machines used during WWII.)

1.2.1. Substitution Cipher
1.2.2. Permutation Cipher
1.2.3. One time Pad

1.2.1 Substitution Cipher

Substitution Cipher

- ***Plaintext*** and ***ciphertext***: a string over a set of symbols U.

  E.g.

  Let U={“a”, “b”, “c”, …, “z”, “_”}.

  e.g. of plaintext: “hello_world”

- **Key**: a substitution table S, representing an 1-1 onto function from U to U.

  E.g.

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	_
g	v	w	b	n	e	f	h	d	a	t	l	u	c	q	m	z	i	r	s	j	x	o	y	k	_	p

S(a) = g, S(b) = v, …

The inverse of S

$S^{-1}(g)=a$, $S^{-1}(v) = b$

- The **key space** is the set of all possible keys. The **key space size** or **size of key space** is the total number of possible keys. The **key size** or **key length** is the number of bits required to represent a key. Here, the key space size is (27!) key size is approximately 94 bits.

Substitution cipher: encryption/decryption

Encryption: Given a plaintext, which is a string

$X = x_1\ x_2\ x_3\ \ldots\ x_n$

and the key S, output the ciphertext

$E_S(X) = S(x_1)\ S(x_2)\ S(x_3)\ \ldots\ S(x_n)$

E.g.

plaintext: h e l l o _ w o r l d

ciphertext: h n l l q p o q i l b

Decryption: Given a string of ciphertext of length n

$C = c_1\ c_2\ c_3\ \ldots\ c_n$

and the key S, output the plaintext

$D_S(C) = S^{-1}(c_1)\ S^{-1}(c_2)\ S^{-1}(c_3)\ldots S^{-1}(c_n)$

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	_
g	v	w	b	n	e	f	h	d	a	t	l	u	c	q	m	z	i	r	s	j	x	o	y	k	_	p

Attacks on Ciphers

- The attacker's goal is to find the key, or to obtain some information of the plaintext (if the key can be found, then the plaintext can be obtained. Nevertheless, the converse not necessary holds.)

- The attackers need access to some information before commencing the attack. E.g, a large number of ciphertexts all encrypted using the same key, or pairs of ciphertext and the corresponding plaintext, etc.

- A simple attack is to exhaustively search the keys, i.e. examine all possible keys one by one. Using exhaustive search, eventually (although this might take very very long) the correct key can be found*.

- So, for a cipher to be secure, exhaustive search must be computationally *infeasible,* e.g. taking millions of years using state-of-the-art supercomputer (surprisingly, it is feasible to exhaustively search some modern ciphers, e.g. DES)

- Sophisticated attacks exploit weakness of the encryption scheme so that it can break faster than exhaustive search.

*: Exception: "Information theoretic secure" such as one-time-pad.

Exhaustive search (aka brute-force-search).

- Consider the substitution cipher (with table size of 27).

- We assume that the attacker knows a ciphertext ***C*** and the corresponding plaintext ***X***. The attacker want to find the key.

  Let ***S*** be the set of all possible substitution tables. Given ***X***, ***C***.

  1. For each S in ***S***
  2. Compute X' = D_S (***C***); If (X' == ***X***) then break;
  3. end-for
  4. Display ("The key is ", S);

- The running time depends on the size of the key space ***S.***

- Since a key can be represented by a sequence of 27 symbols. The size is key space is 27!

  (So, in any representation, the key length is at least $\log_2(27!) \approx 94$ bits.)

- Eventually, exhaustive search will find the key. However, in the worst case, the exhaustive search needs to carry out $27! \approx 2^{94}$ loops, and on average $\approx 2^{93}$ loops. This is infeasible using current compute power (see tutorial 1).

Attack: Known-plaintext-attack on Substitution Cipher

- You probably have realized that the attacker doesn't need to carry out exhaustive search. Given a plaintext and ciphertext, e.g

  plaintext: `h e l l o _ w o r l d`

  ciphertext: `h n l l q p o q i l b`

  The attacker can figure out the entries in the key

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	_
			b	n			h				l			q			i					o				p

For sufficiently long ciphertext, the full table can be found.

- "***Known plaintext attack":*** a scenario where the attackers have access to pairs of ciphertexts and the corresponding plaintexts.
- If the adversary can successfully derive the key, we say that the scheme is

  "***not secure under known plaintext attack***" or

  "***broken under known plaintext attack***"

Hence, substitution cipher is not secure under known plaintext attack.

Attack on Substitution cipher (Ciphertext only attack)

- **Ciphertext only attack:** The attackers have access to ciphertext only (i.e. without the corresponding plaintext).
- Suppose an attacker knows that the plaintext is an English sentence, can he find the key using exhaustive search under ciphertext only attack ? *Yes*.

Let ***S*** be the set of all possible substitution table. Given C.

1. For each S in ***S***
2. Compute $X = D_S(C)$; if X contains words in the English dictionary, then break;
3. end-for
4. Display ("The key is ", S);

- Likewise, eventually, the exhaustive search will find the key. However, the attack is computationally infeasible. (There is a very small probability that the above found the wrong key. For long ciphertext, say 50 characters long, the probability that a wrong key will give a meaningful English sentence is very low. The probability is so small that we treat it as "***negligible***").

- Are there efficient ciphertext only attacks on substitution cipher? ***Yes.***

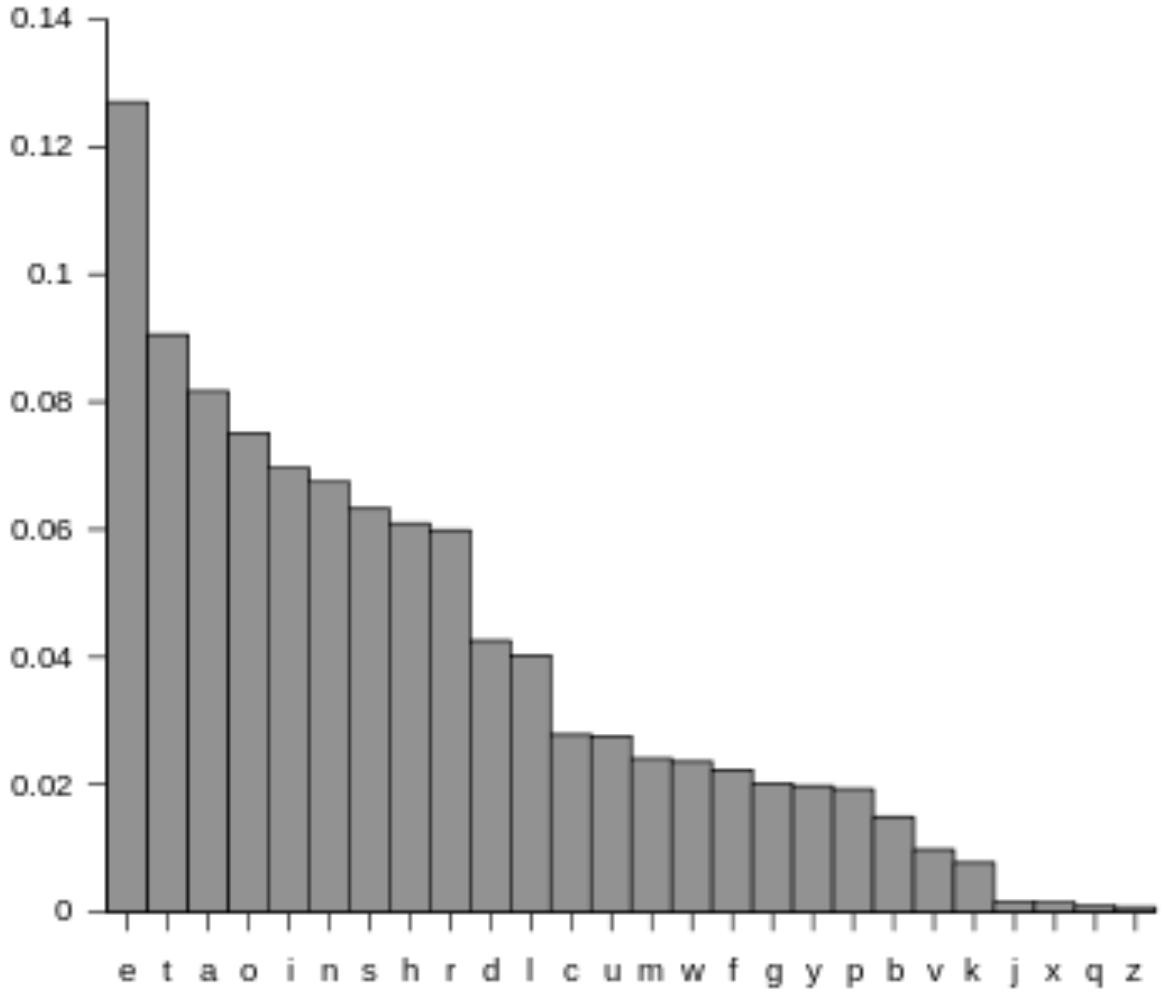
- Substitution cipher is vulnerable to ***frequency analysis*** attack.

- Note that in the hello_world example, “o” appears 2 times in the plaintext, whereas the corresponding “q” also appears 2 times in the ciphertext.

- Suppose the plaintexts are English sentences. The frequency of letters used in English is not uniform, for e.g. “e” is more commonly used than “z”. Given a sufficiently long ciphertext (say, around 50 characters), adversary may able to guess the plaintext by mapping the frequently occurred letters in the ciphertext to the frequent letters used in English.

- If adversary is aware that the plaintexts are English sentences, then he could successfully carry out frequency analysis attack. Hence, substitution cipher is ***not secure under ciphertext only attack***, when the plaintexts are English sentences.

Frequency of letters in English text.



from http://en.wikipedia.org/wiki/Letter_frequency

Remarks on Know plaintext attack.

To carry out known plaintext attack, the attacker needs to obtain at least a pair of ciphertext and the corresponding plaintext.

This is possible in practice. Eg, for email, certain words in the header are fixed such as “From”, “Subject”, etc. The first few bytes of many protocols are also fixed with only a few choices, or can be easily guessed (e.g. first few bytes of network packets).

During WWII, cryptologists exploited commonly used words like “weather” and “nothing to report” as the known plaintext to guess the secret keys.

(optional: read more about “Enigma Machine”)

1.2.2 Permutation cipher

Permutation Cipher

- Also known as transposition cipher.

  The encryption first groups the plaintext into blocks of ***t*** characters, and then applied a secret "permutation" to each block by shuffling the characters.

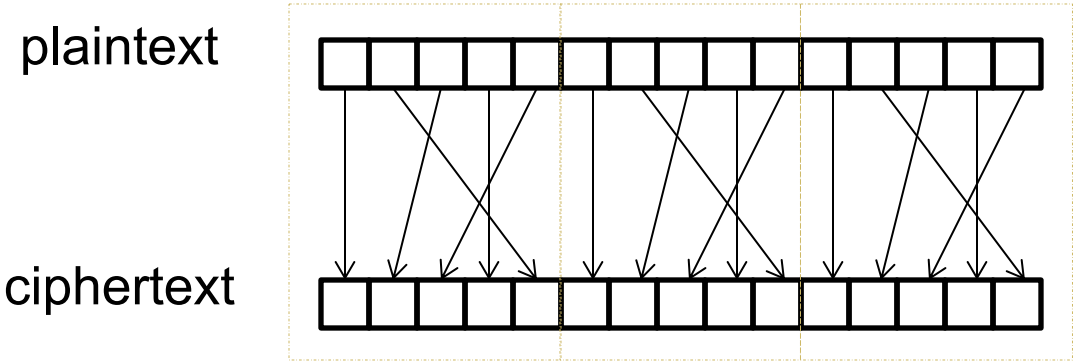
  The key is the secret "permutation", which is an 1-1 onto function *e* from {1,2,..,t} to {1,2,...,t}. The size ***t*** could be part of the key, that is, ***t*** is also kept secret. We can write the permutation ***p*** as a sequence

  $$\boldsymbol{p} = (p_1, p_2, p_3, \ldots p_t)$$

  which shift the character at position i (position within the block) to the position p_i.

Example:

Given the plaintext and the key t=5, ***p***=(1,5,2,4,3)



Cryptanalysis (known-plaintext attack)

- Permutation cipher fails miserably under known-plaintext attack.

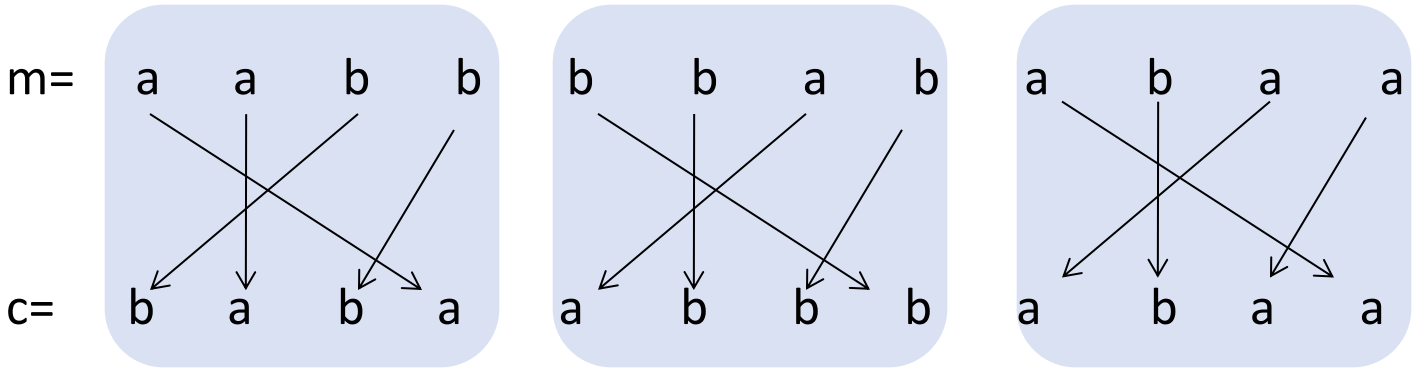
  Given a plaintext and a ciphertext, it is very easy to determine the secret key.

m= a a b b b b a b a b a a

c= b a b a a b b b a b a a

in the above, what is the block size *t?* What is the permutation?

- Permutation cipher is also easily broken under ciphertext only attack if the plaintext is English text.


m=
a a b b
c=
b a b a
b b a b
a b b b
a b a a
a b a a

1.2.3 One Time Pad

XOR operation

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

$A \oplus B = (A+B) \bmod 2$

Some interesting properties:

- Commutative: $A \oplus B = B \oplus A$
- Associative: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
- Identity element: $A \oplus 0 = A$
- Self-inverse: $A \oplus A = 0$

One-time-pad

Encryption:

Given n-bit Plaintext: $x_1, x_2, \ldots, x_n$ and n-bit key: $k_1, k_2, \ldots, k_n$

ciphertext C= $(x_1 \oplus k_1), (x_2 \oplus k_2), (x_3 \oplus k_3), \ldots, (x_n \oplus k_n)$

Decryption:

Given n-bit ciphertext: $c_1, c_2, \ldots, c_n$ and n-bit key: $k_1, k_2, \ldots, k_n$

plaintext $X = (c_1 \oplus k_1), (c_2 \oplus c_2), (c_3 \oplus k_3), \ldots, (c_n \oplus k_n)$

E.g. One-time-pad

Encryption: plaintext $\oplus$ key → ciphertext
Decryption: ciphertext $\oplus$ key → plaintext

PlainText	0	0	1	0	1	1	0
Key	**1**	**1**	**0**	**0**	**1**	**1**	**1**
Ciphertext	1	1	1	0	0	0	1

encryption

decryption

Correctness (decrypting the ciphertext give back the plaintext):

For any x, k, $(x \oplus k) \oplus k = x \oplus (k \oplus k) = (x \oplus 0) = x$

ciphertext

plaintext

Security of one-time-pad

- From a pair of ciphertext and plaintext, the attacker can derive the key. However, such key is useless, since it will not be used any more.

- Note that even exhaustive search can't work on one-time-pad. (Suppose we are given a 1Kbytes ciphertext, and are told that the plaintext is a jpeg image. By using exhaustive search, can we eventually find the plaintext?)

- In fact, It can be shown that one-time-pad leaks no information of the plaintext, even if the attacker has arbitrary running time. Hence, it is sometime called "unbreakable".

- CS4236 would look into the formulation of "Perfect Secrecy" of one-time-pad

- However, the length of the key is the same as plaintext. Useless in many applications.

- Nevertheless, it is practical in some scenarios.

  see http://ciphermachines.com/otp

and the Venona Story

(where one-time-pads fails)

(optional:

https://www.nsa.gov/Portals/70/documents/about/cryptologic-heritage/historical-figures-publications/publications/coldwar/venor

http://ciphermachines.com/otp

Intuitively, Perfect Secrecy means:

Optional

Attacker's prior knowledge of the unknown plaintext x. (before knowing y)

Definition: A cryptosystem has ***perfect secrecy*** if

for any distribution X, for all x, y

$\Pr(X=x \mid Y=y) = \Pr(X=x)$.

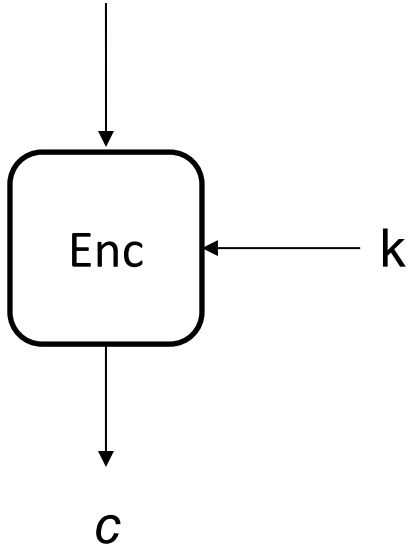
Attacker's updated knowledge of the unknown plaintext, after the attacker had seen the ciphertext y.

for any ciphertext y and plaintext x, the chances that an attacker correctly predicts x before knowing y, and after knowing y, are the same.

E.g.

- Let ***c*** be the ciphertext of the Oscar's winner name, encrypted using some secret key.
- Bob gathered information from many sources and believed that the movie *"Alice in Crypto Land"* has 42% chance of winning.
- Now, suppose somehow, Bob knew ***c***. With the additional knowledge of ***c,*** could Bob improve his guess? (any slight improvement would be useful).
- Let's assume that Bob have extremely powerful machine that can exhaustively search all keys. Can he improve the guess?
  - *(for DES, the answer is yes! How?)*
  - *(if the encryption scheme achieve perfect secrecy, the answer is no!)*

"And the Oscar goes to xxxxxxxxx"



1.3 Modern Ciphers

Modern ciphers generally refer to schemes that use computer to encrypt/decrypt.

E.g. RC4, DES, A5, AES, RSA

1.3.1 DES/Exhaustive Search

Modern ciphers

Designs of modern ciphers take into considerations of known-plaintext-attack, frequency analysis and other known attacks.

E.g. DES (Data Encryption Standard, 1977)
RC4 (Rivest's Cipher 4, 1987)
A5/1 (used in GSM, 1987)
AES (Advanced Encryption Standard, 2001)

They are supposed to be "secure", so that any successful attack does not perform noticeably better than exhaustive search.

(optional: Nevertheless, RC4 is broken in some adoptions, A5/1 is vulnerable, and DES's key length is too short. Wiki on RC4, A5/1 and DES give quite good description. AES is believed to be secure, and classified as "Type 1" by NSA https://en.wikipedia.org/wiki/NSA_cryptography#Type_1_Product)

Exhaustive search and key length (see Work factor [PF2.3page91])

If the key length is 32 bits, there are 2^{32} possible keys. Hence, the exhaustive search has to loop for 2^{32} times in the worst case. (On average, 2^{31} times)

We can quantify the security of an encryption scheme by the length of the key. Consider a scheme **A** with 64-bit keys and a scheme **B** with 54-bit keys. Scheme **A** is more secure w.r.t. exhaustive search. (note that some schemes, e.g. RSA, have known attacks that are more efficient than exhaustively searching all the keys. In those cases, we still want to quantify the security by the equivalent of exhaustive search. For e.g, in the best known attack on a 2048-bit RSA, roughly 2^{112} searches are required. Hence we treat its security equivalent to 112 bits. So, we say that the 2048-bit RSA has key strength of 112 bits).

How many bits is considered "secure"? (Tutorial 1)

read NIST Recommended key length for AES http://www.keylength.com/en/4/

Exhaustive Search on DES

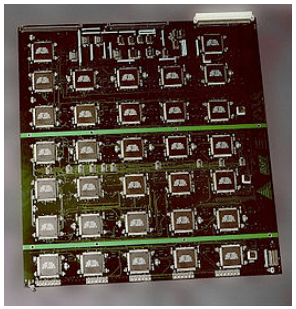
Key length of DES is 56 bits.

While exhaustive search on 56 bits seemed infeasible in the 70s, very soon, it is possible using distributed computing or specialized chip.

RSA Security hosted a few challenges on DES. (Note: RSA is an encryption scheme, RSA Security is a company.)

DES Challenge II-1: "The secret message is: Many hands make light work." (found in 39 days using distributed computing, early 1998)

DES Challenge II-2: "The secret message is: It's time for those 128-, 192-, and 256-bit keys." (found in 56 hours using specialized hardware, 1998)



EFF's DES cracking machine.

A question is, why would an agency design a scheme that can be broken in the near future? Many believed that it was intentional.

AES

AES

- In 2000, a new standard for block cipher AES (Advance Encryption Standard) was proposed by NIST. The selection process was transparent and with worldwide involvement.

- NIST called for proposal in 1997 and received 21 submissions by Jun 1998. In 2000, *Rijndael* was selected as AES.

- Rijndael was invented by Belgian researchers Daemen and Rijmen.

- AES block length is 128, and key length can be 128, 192 or 256 bits.

- Currently, no known attacks on AES. (there are some attacks on the mode-of-operation)

- NSA classifies AES as "Suite B Cryptography".

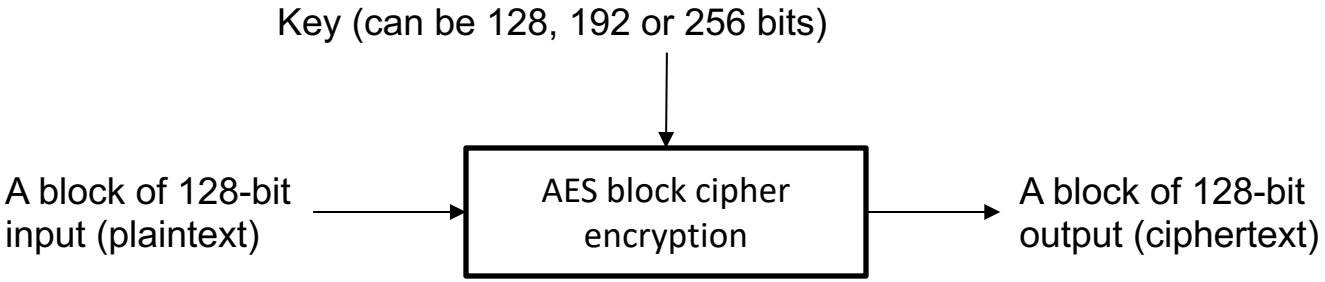
  "NSA Suite B Cryptography *is a set of cryptographic algorithms promulgated by the National Security Agency as part of its Cryptographic Modernization Program. It is to serve as an interoperable cryptographic base for both unclassified information and most classified information."*

  see https://en.wikipedia.org/wiki/NSA_Suite_B_Cryptography

1.3.2 Block cipher & Mode-of-Operations

Block Cipher

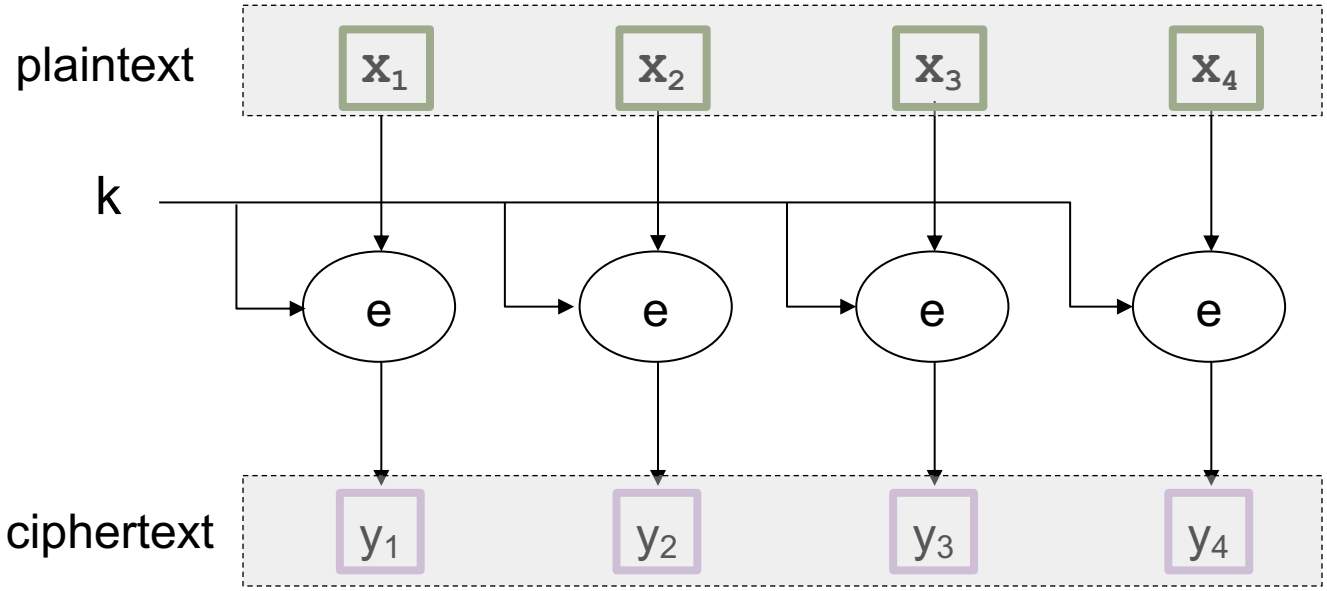
- DES and AES is also known as "Block Cipher". Block cipher are designed for some fixed size input/output. E.g. AES is designed for 128 bits input/output. (I try not to call the input as plaintext, this is because in some mode-of-operations, e.g. Counter Mode, the input is not the plaintext)



- For large plaintext (say 10 MB), it is first divided into blocks, and the block cipher is then applied. The method of extending encryption from a single block to multiple blocks is not straightforward. It is called *mode-of-operation*.

Mode-of-operation: ECB mode

- Electronic Code Book naturally is the first to come into our mind.
- It divides the plaintext into blocks and then applies block cipher to each block, all with the same key.

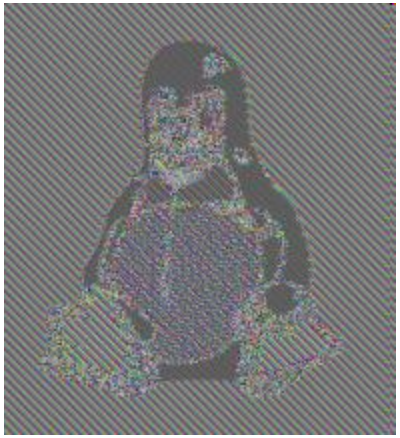


- ECB is not secure! It leaks information.

ECB

In the following example, the image is divided into blocks, and encrypted with some deterministic encryption scheme* using the same key. Since it is deterministic, any two blocks that are exactly the same (for e.g. from the white background) will be encrypted to the same ciphertext.

Plaintext



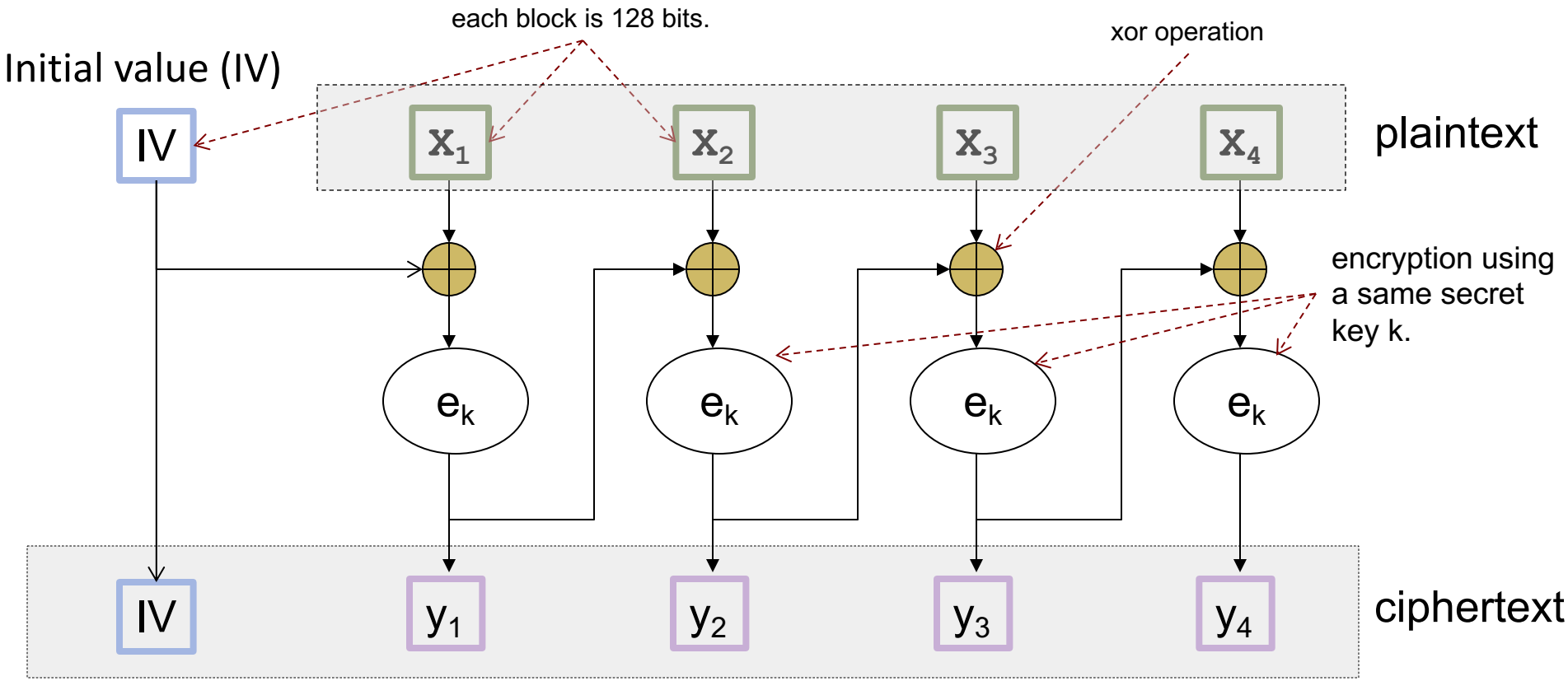
ciphertext

*: An encryption scheme is “deterministic” in the sense that, the encryption algorithm will always produce the same output (i.e the ciphertext) when given the same input (i.e. the key and plaintext).
In contrast, a “probabilistic” encryption scheme produces different ciphertext even with the same input (key, plaintext).

AES is deterministic. However, if we employ AES with a randomly chosen initial value (IV), then it is probabilistic.

- To prevent the leakage of information shown in the previous slide, additional mechanisms are required. Some mode-of-operations “link” the blocks, so that two blocks with the same content would give different ciphertext. Popular mode-of-operations are CBC and CTR (counter) mode.

mode-of-operation: Cipher Block Chaining (CBC) on AES

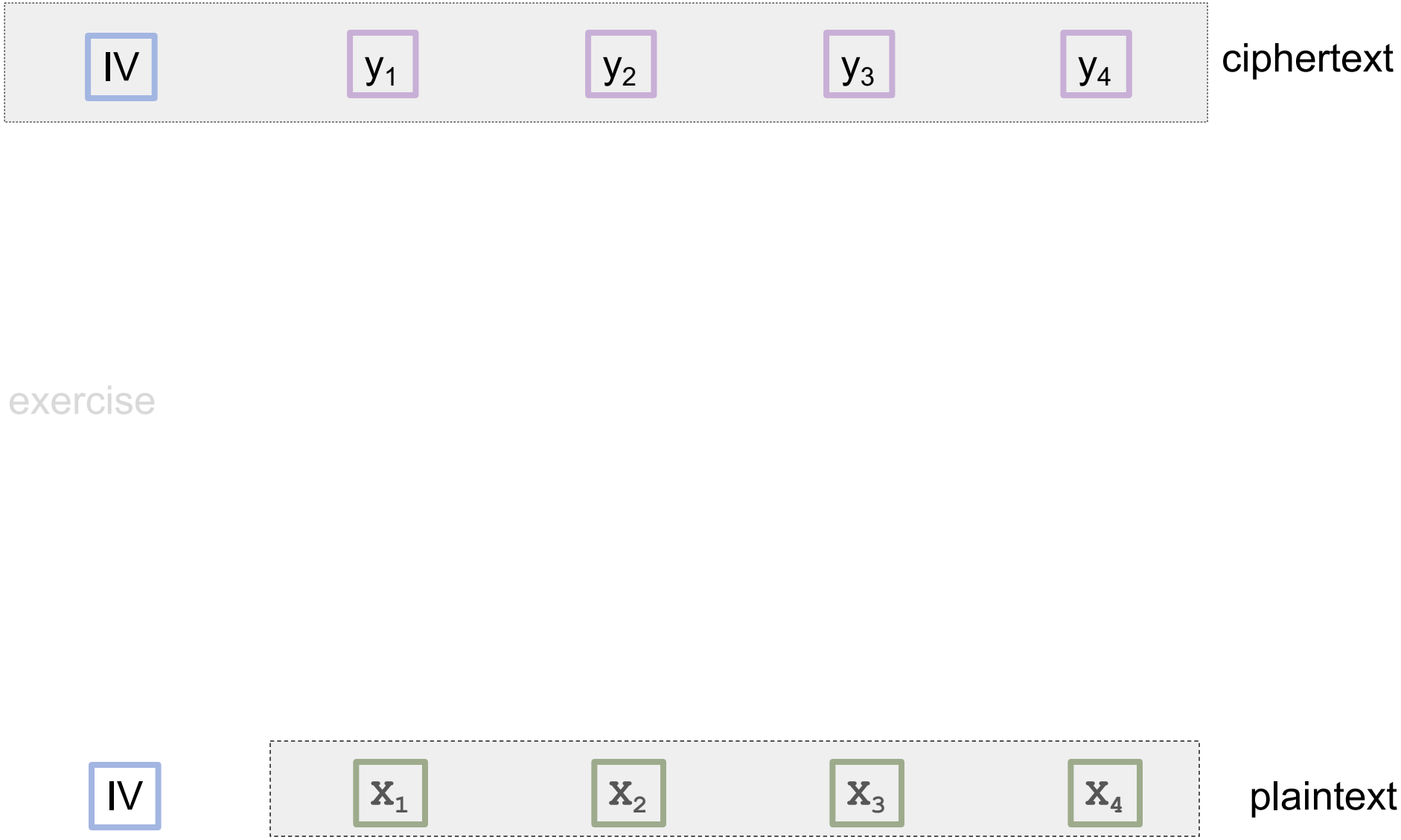


- The Initial Value (IV) is an arbitrary value chosen during encryption. So, it is different in different encryptions of the same plaintext. Depending on implementation, IV can be randomly chosen, obtain from a counter, or from other info.

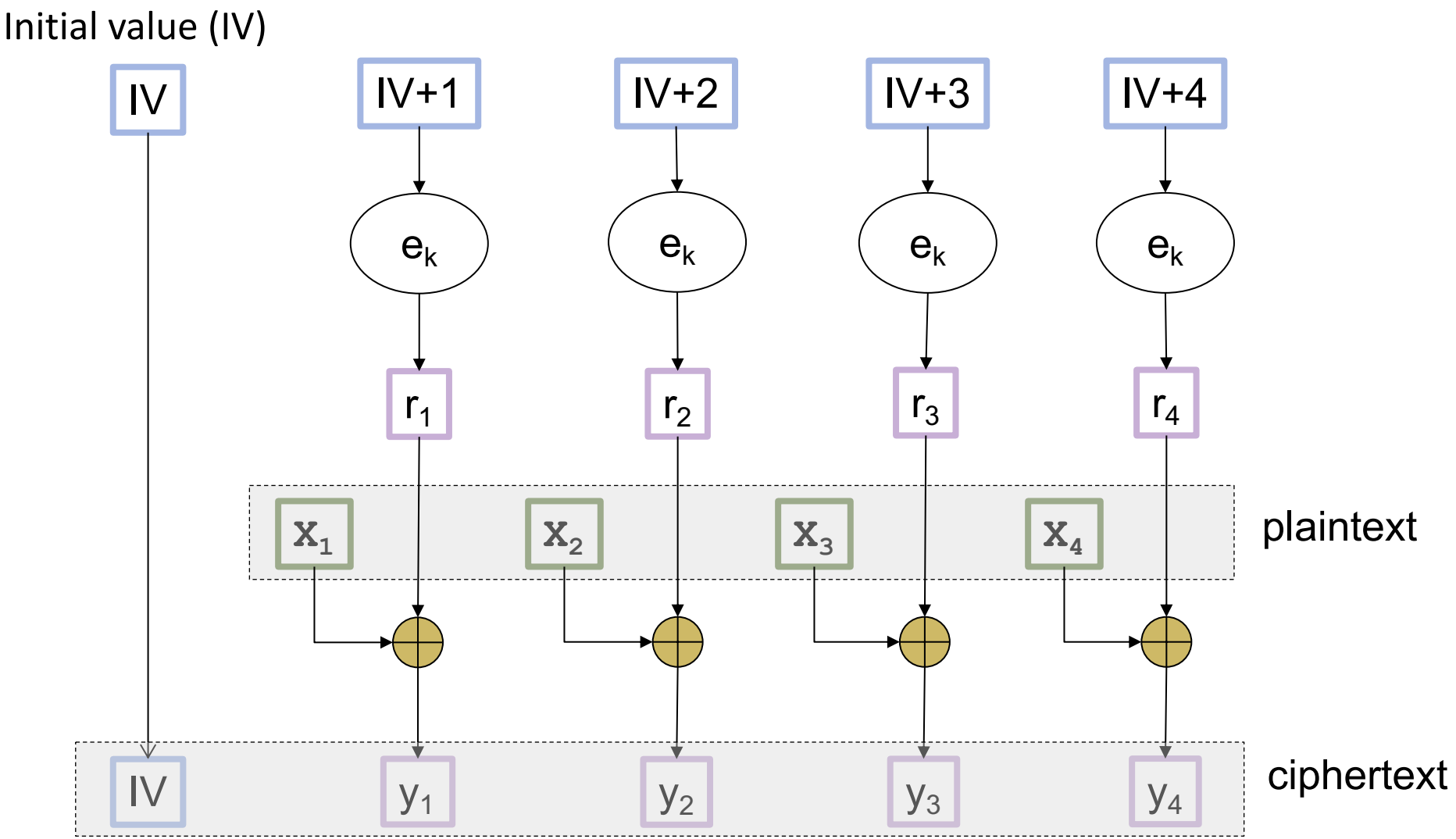
$$y_0 = IV. \qquad y_i = E_k(x_i \oplus y_{i-1}) \qquad \text{for } i>0$$

Note: In the above figure, we treat IV as part of the final ciphertext. The terminology is not consistent in the literature.
Some documents may state that "the final message to be sent are the IV and the ciphertext" (i.e. IV is not included in the "ciphertext").
In this module, when it is crucial, we will explicitly state whether IV is included or excluded. (e.g. "Ciphertext together with a IV").

Cipher Block Chaining (CBC) decryption



mode-of-operation: Counter mode (CTR) on AES



This is a stream cipher. (next section)

Programming example

This key can be randomly chosen or set by user (see crypto pitfalls). Of course, if the key is randomly chosen, it has to securely sent to the receiver and/or store in a secure place.

- Python.

(package PyCryptodome https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html)

The IV should be randomly chosen.

```
>>> from Crypto.Cipher import AES
>>>
>>> key = b'Sixteen-byte key'
>>> iv  = b'Sixteen-byte  IV'
>>> cipher = AES.new(key, AES.MODE_CBC, iv)
>>> c=iv+cipher.encrypt(b'Plaintext of length with multiple of 16    bytes')
```

In Python, to display a byte sequence, we can use…

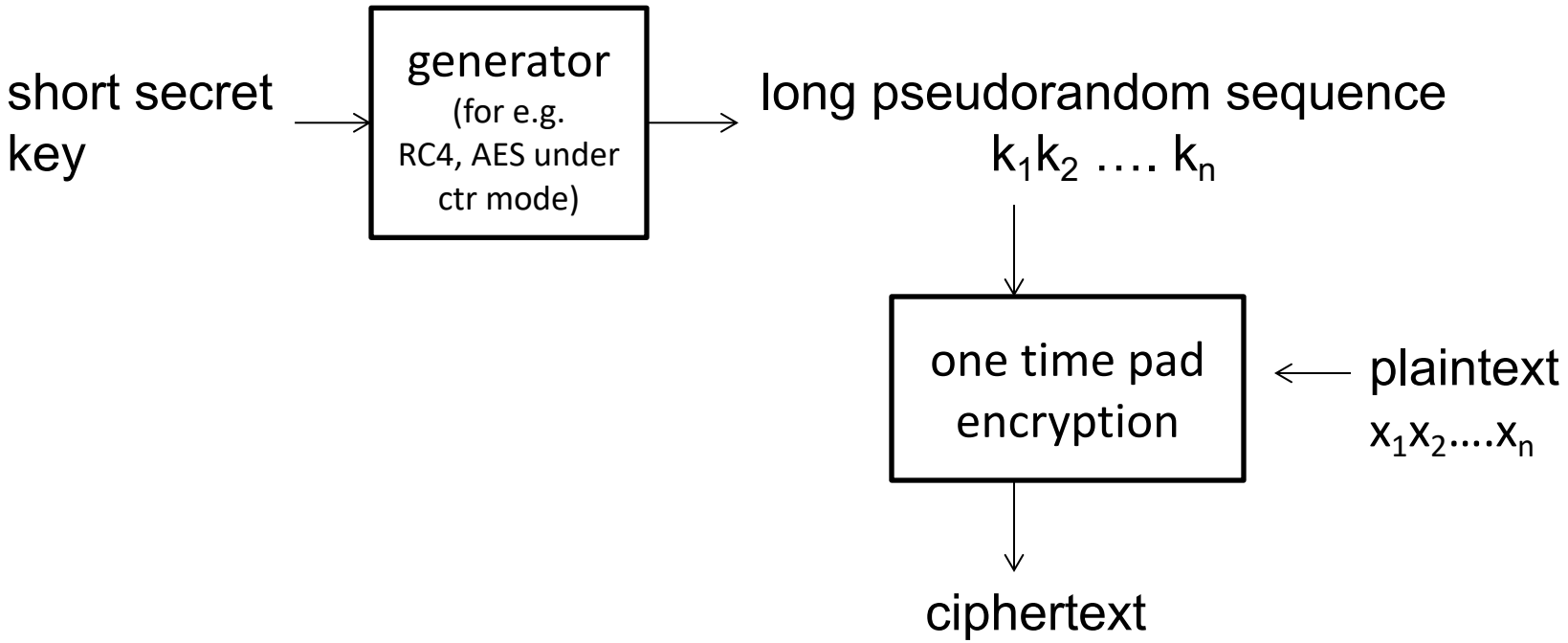
```
>>> from base64 import *
>>> b16encode(c)
b'5369787465656E20627974652020204956B186083256CACCBD1638AF4877FBF2AAFBECB66FE13C403D7CE8EA04D028E66CA6AE1294
FF51C2F363CCC8953137A6A3'
```

1.3.3 Stream Cipher and IVs

Stream Cipher

Stream Cipher is "inspired" by one-time-pad.

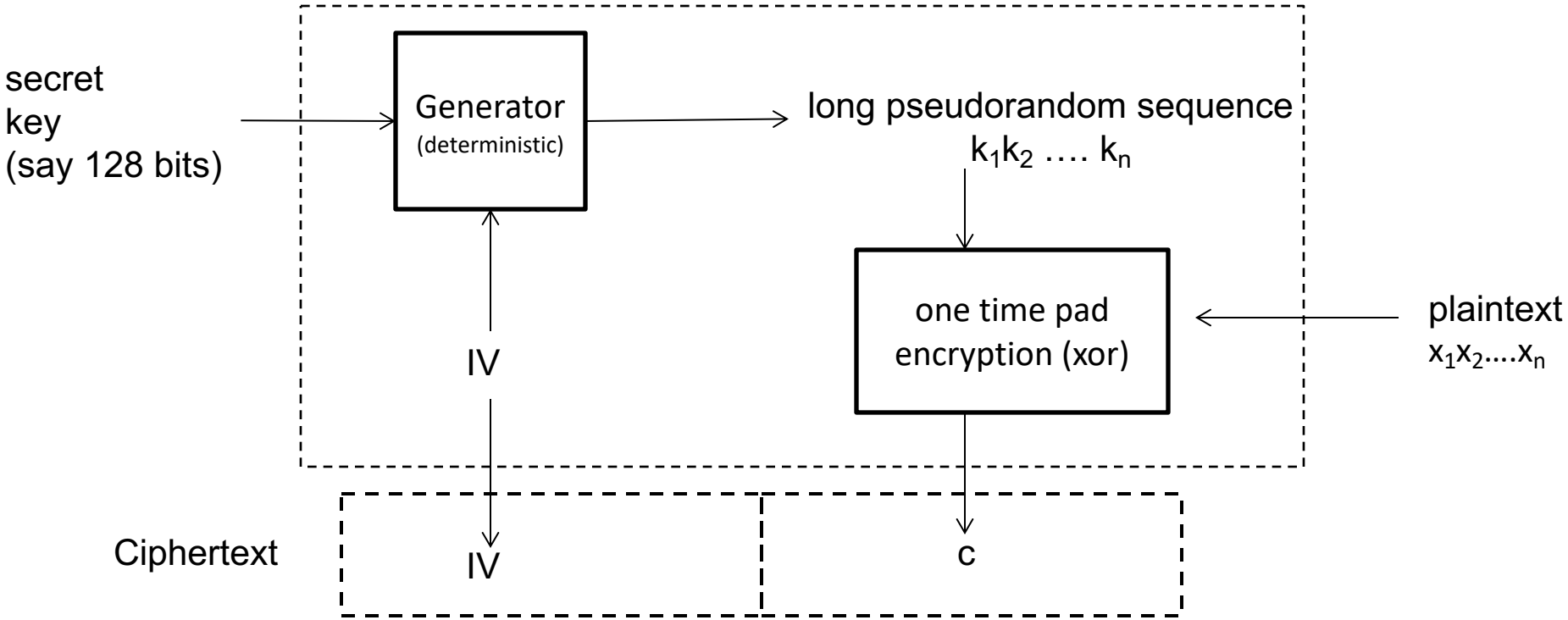
Suppose the plaintext is 2^{20} bits, but the secret key is only 256 bits. Steam cipher generates a 2^{20}-bit sequence from the key, and takes the generated sequence as the secret key in one-time-pad. The generator has to be carefully designed, so that it gives ***cryptographically secure pseudorandom sequence***. (security requirement: difficult to distinguish the sequence from a truly random sequence – this is quite a strong requirement. It implies that it is difficult to get the short secret key from the sequence. It also implies that it is difficult to get part of the sequence after seen another part of the sequence.)



Stream Cipher with IV

Most ciphers, including stream ciphers, have an Initial Value (IV). The IV can be randomly chosen, or from a counter.

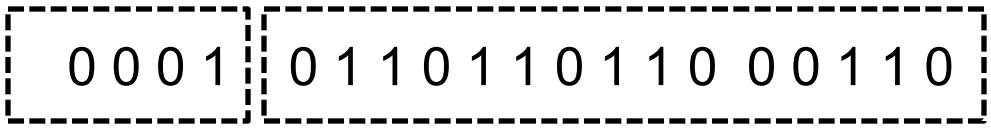
- In stream cipher, a long pseudorandom sequence is generated from the secret key together with the IV. The final ciphertext contains the IV, followed by the output of the one-time-pad encryption.
- For decryption, the IV is extracted from the ciphertext. From the IV and the key, the same pseudorandom sequence can be generated and thus obtain the plaintext.



e.g.

Encryption: Given
15-bit Plaintext X = 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0
short key = 0 1 0 1

Step 1: Randomly generates an IV, say
IV = 0 0 0 1
Step 2: From the short key and IV, generates a 20-bit sequence
K= 0 1 1 0 1 0 1 0 0 1 0 0 1 1 0
Step 3: outputs IV, follow by K xor X



Decryption: Given the short key and the ciphertext with the IV.
Step 1: Extracts the IV from the ciphertext
Step 2: From the short key and IV, generates the long sequence.
Step 3: Performs xor to get the plaintext.

Why IV? What if the IV is always the same?

Suppose there isn't an IV (or the IV is always set to be a string of 0's)

Consider the situation where the same key is used to encrypt two different plaintexts

$\mathbf{X} = x_1, x_2, x_3, x_4, x_5$ and

$\mathbf{Y} = y_1, y_2, y_3, y_4, y_5$

Further suppose that an attacker eavesdropped and obtained the two corresponding ciphertexts **U**, **V**.

The attacker can now compute

$$\mathbf{U} \oplus \mathbf{V} = (\mathbf{X} \oplus \mathbf{K}) \oplus (\mathbf{Y} \oplus \mathbf{K})$$

By associative and commutative property of xor

$$\mathbf{U} \oplus \mathbf{V} = (\mathbf{X} \oplus \mathbf{Y}) \oplus (\mathbf{K} \oplus \mathbf{K}) = \mathbf{X} \oplus \mathbf{Y}.$$

So, from **U** and **V**, the attackers can obtain information about $\mathbf{X} \oplus \mathbf{Y}$, i.e. the following sequence

$$(x_1 \oplus y_1), (x_2 \oplus y_2), (x_3 \oplus y_3), (x_4 \oplus y_4), (x_5 \oplus y_5)$$

What so big deal about revealing X ⊕ Y?

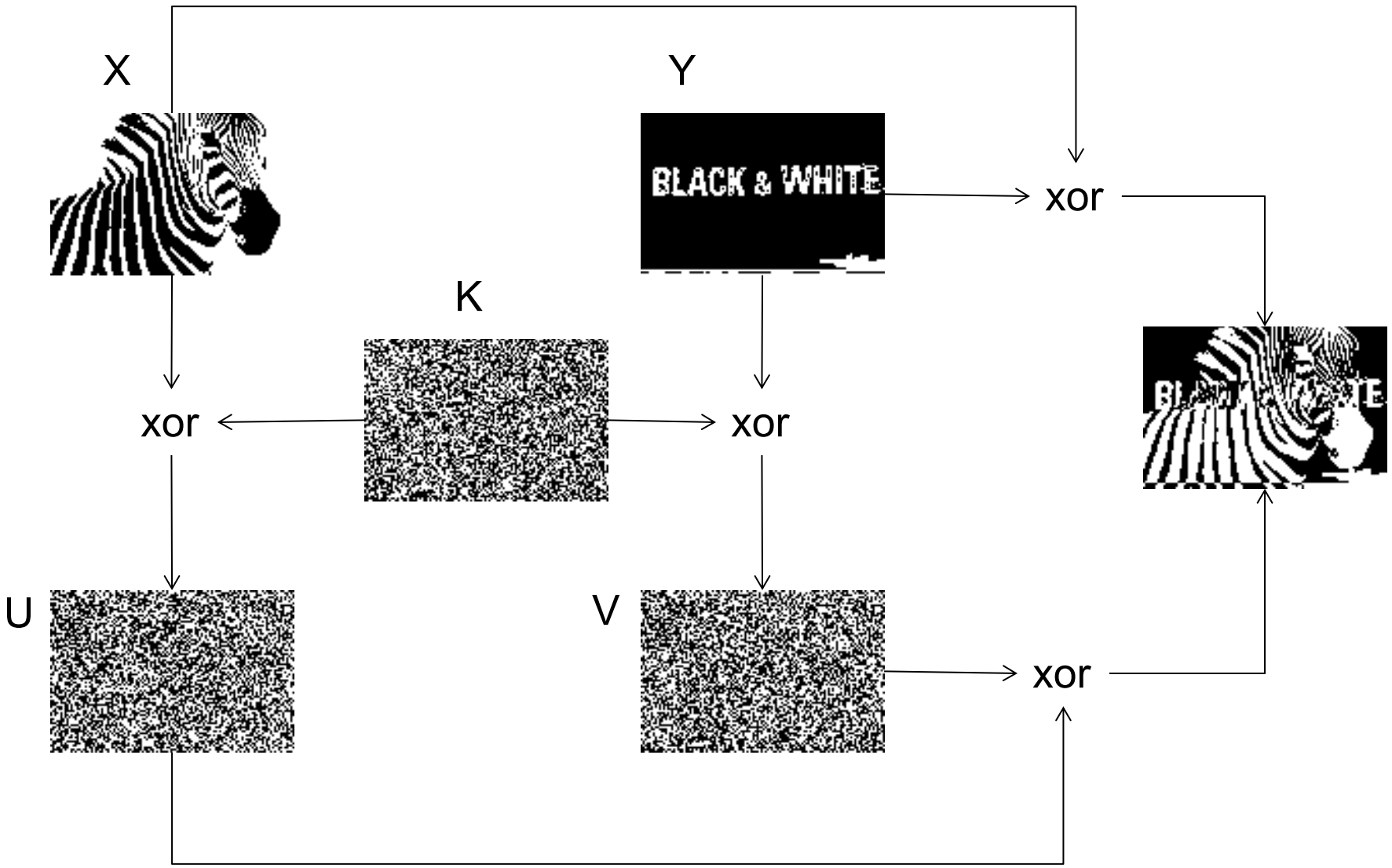
Suppose X is an 80x120 image of an animal. Each pixel is either black or white (0 or 1). The image can be represented as a (80x120)-bit sequence where each bit corresponds to a pixel.

Y is another 80x120 pixels image rendering two words, which is similarly represented as a sequence.

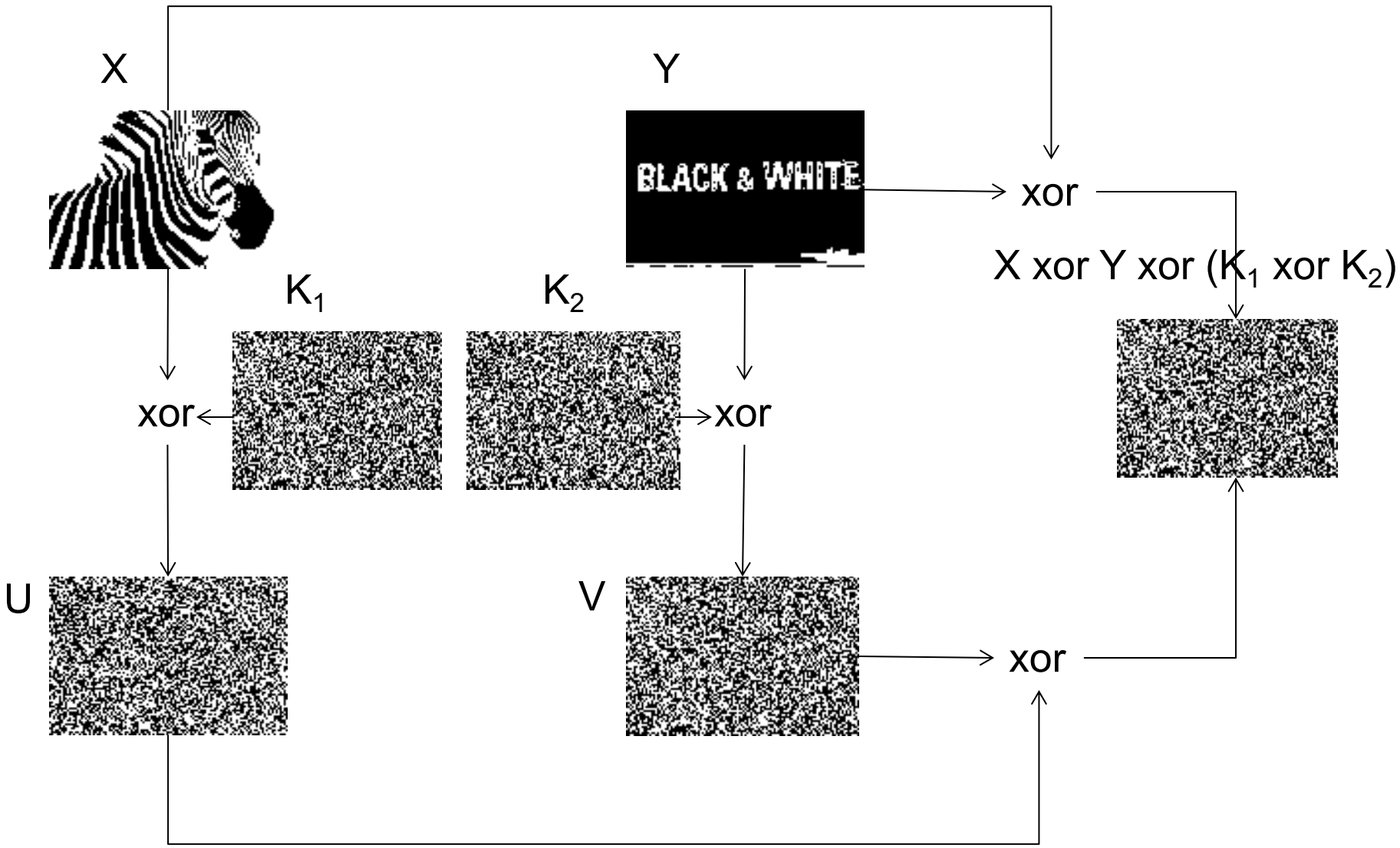
Here is X ⊕ Y.

What is X, Y?

stream cipher without IV



stream cipher with IV



Role of IV

- If the IVs are different in two different processes of encryption, the two pseudorandom sequences will be different. Hence,
  - The ciphertexts of a same plaintext will be different under these two different encryptions.
  - If the two plaintexts are different, by xor'ing the two ciphertexts would not cancel out the pseudorandom sequences.

- IV makes an encryption "probabilistic".

Role of IV

- IV are also applied to CBC. The reason is the same. We want the encryption to be non-deterministic. That is, two different encryptions of the same plaintext would give two different ciphertexts.

- Without IV, the encryption is deterministic. If the encryption is deterministic, this leaks information on whether the plaintext of two ciphertexts are the same.

Consider the scenario where an attacker is given C_1 and C_2. If the encryption is deterministic, then the attacker can derive that the plaintext are the same iff $C_1 == C_2$

1.4. Examples of attacks

1.4.1 Triple DES & Meet-in-the-middle

1.4.2 Padding Oracle Attack

- Notions of Oracle in security analysis
- The attack
- Implications

1.4.1 Triple DES & Meet-in-the-middle attack

http://en.wikipedia.org/wiki/Meet-in-the-middle_attack

(d) Triple DES

- DES is not secure w.r.t. today computing power. One way to improve it is by multiple encryptions: encrypt the plaintext twice or more, using different keys.

- DES doesn't form a group in the sense that, E_{k1} (E_{k2} (x)) is not the same as E_{k3} (x) for some k3.

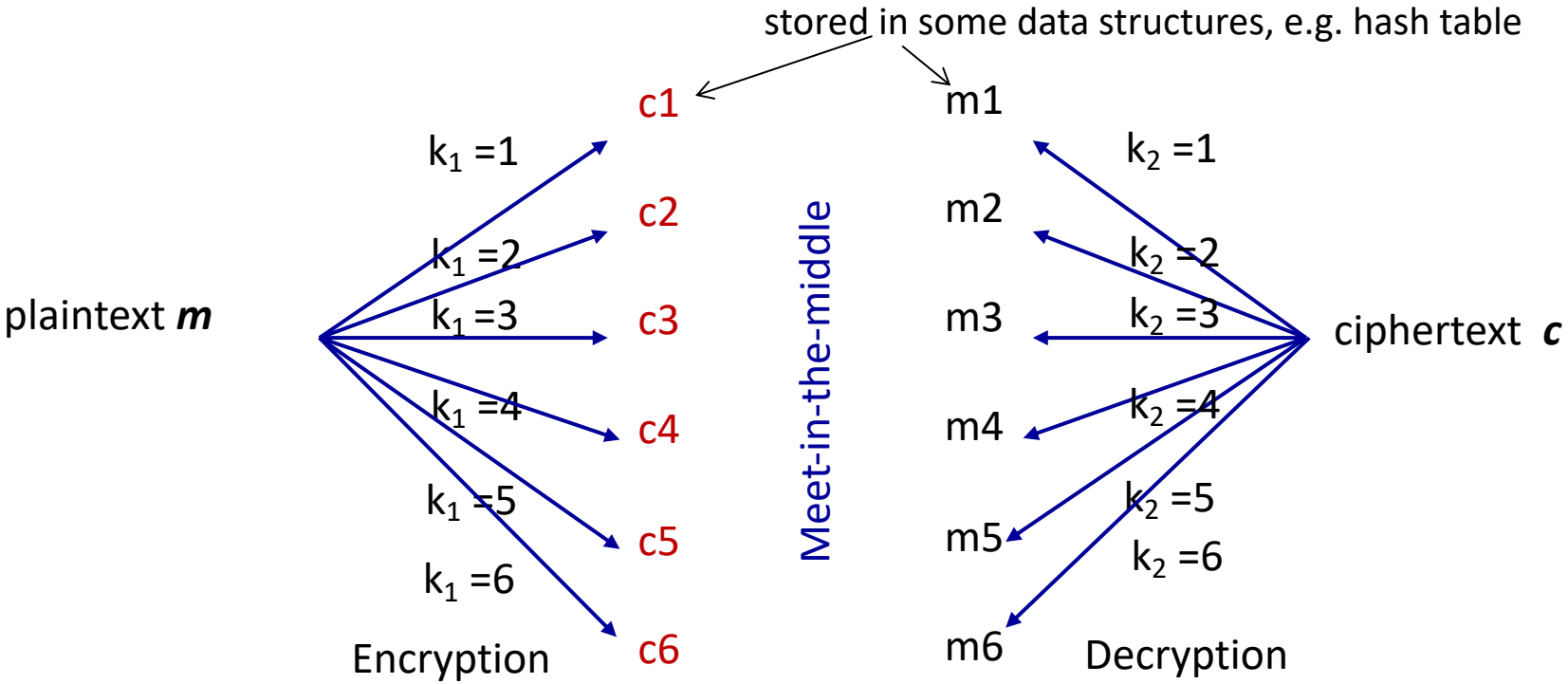
- Let us consider *double* encryption and known plaintext attack. That is, the adversary has a plaintext ***m*** and the corresponding ciphertext ***c***, and wants to find the two secret keys $\boldsymbol{k_1}$, $\boldsymbol{k_2}$.

- Using exhaustive search, what is the amount of DES encryption/decryption required? 2^{56+56} . That is key-strength seems to double to 112. But this is not true. Attacker can do better…. using more storage space.

Meet-in-the-middle Attack

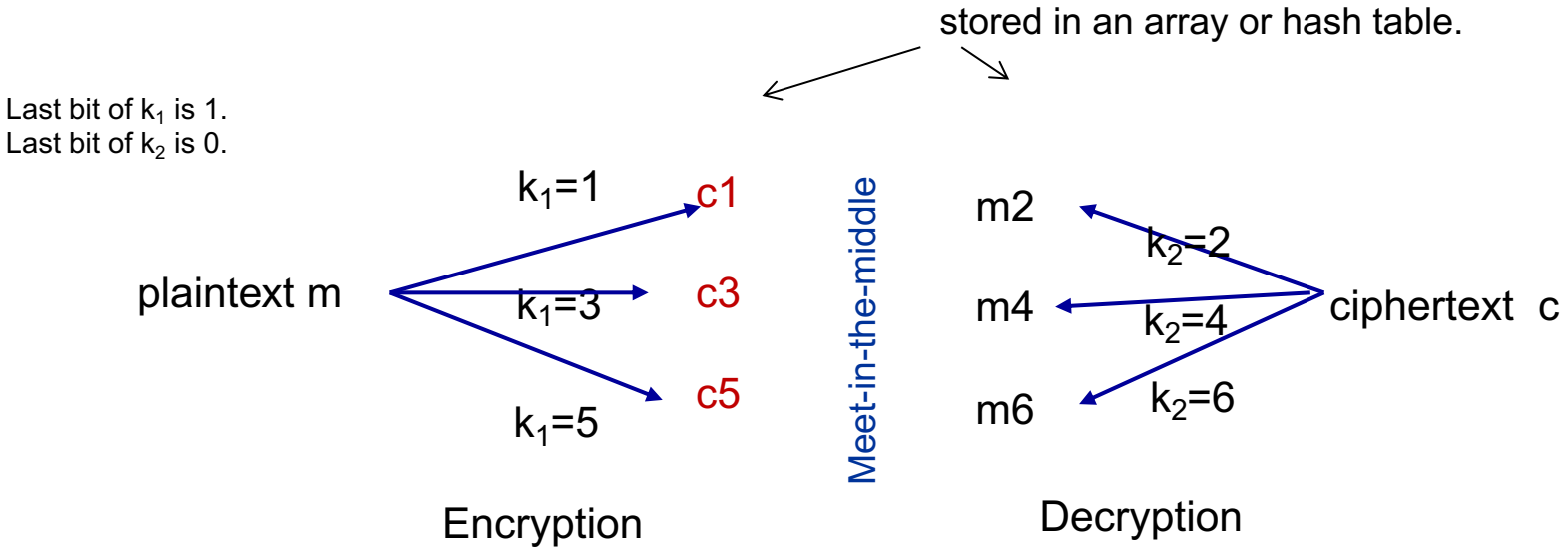
- Not to confuse “meet-in-the-middle” with “man-in-the-middle” attack.
- Introduced by Diffie & Hellman in 1977.
- This is a *known plaintext attack.* That is, attacker has at least a pair (***m***, ***c***) of plaintext and the corresponding ciphertext.

Meet-in-the-middle attack



- Given *c* and *m,* find the two keys.
  1. Compute two sets ***C*** and ***M***. ***C*** contains ciphertexts of ***m*** encrypted with all possible keys. ***M*** contains plaintexts of ***c*** decrypted with all possible keys.
  2. Find common element in ***C*** and ***M***. From the common element, we can obtain the two keys.
- In the above meet-in-the-middle attack, the attacker only have to perform 6 encryptions and 6 decryptions. In general, for *k*-bit keys, it reduces the number of crypto operations to 2^{k+1} using 2^{k+1} space.

Tradeoff with time and space



- If 2^{k+1} storage is too much, we can have a tradeoff.
- Given ***m***, ***c***.
  1. Exhaustively selects last *s* bits of k_1 , and last *s* bits of k_2
  2. For each selection above, carries out meet-in-the-middle attack. If successful, stops the exhaustive search.
- The storage requirement dropped to 2^{k-s+1}, but number of cryptographic operations increased to $2^{2s}\ 2^{k-s+1} = 2^{s+k+1}$.

3DES

- Remedy--- Use Triple encryptions, but 2 keys.

  a) E_{k1} (E_{k2} (E_{k1} (x))).

or

  b) E_{k1} (D_{k2} (E_{k1} (x))).

  Both (a) and (b) are believed to have the same level of security. However, version (b) can be more convenient. By choosing K1=K2, then it is same as single encryption.

- Also known as 3DES, TDES, TDEA, 2TDES, 3TDES (using 3 keys)

- Currently, there is no known efficient attacks on triple DES. However, compare to AES, the triple DES is less efficient w.r.t. encryption/decryption time)

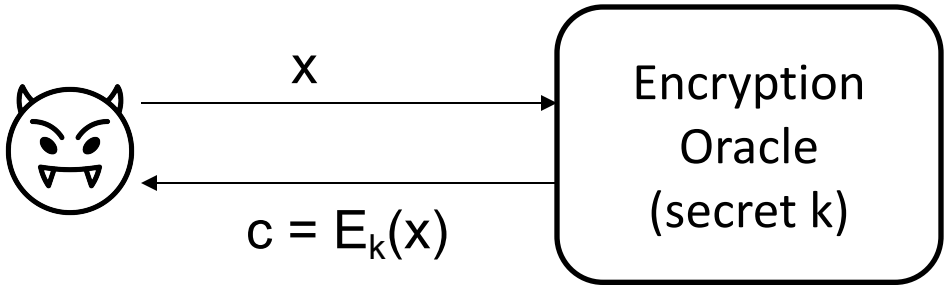
- Triple DES is still in used recently. (I do not know an example but won't be surprised that some systems still use it)

(3DES was the default encryption in Outlook 2007. See its help page: http://office.microsoft.com/en-sg/outlook-help/encrypt-messages-HP006369952.aspx)

1.4.2 Padding Oracle Attack (assignment 1)

Oracle in security analysis

- Recap that in security analysis, it is important to formulate (1) what information the attackers have; (2) attackers' goals.
- One type of information is obtained via a query-answer system, known as ***Oracle.*** The attackers can send in queries, and the ***Oracle*** will output the answer. E.g.
  - **Encryption Oracle.** On query a plaintext **x**, the oracle outputs the ciphertext $E_k(\mathbf{x})$ where the key ***k*** is a secret key.
  - **Decryption Oracle.** On query a ciphertext **c**, the oracle outputs the plaintext $D_k(\mathbf{x})$ where the key ***k*** is a secret key.
- An attacker can send multiple queries.



Padding Oracle attack

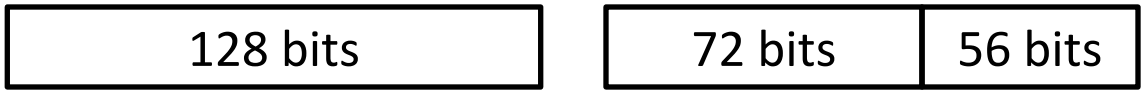
- The attacker have:
  - A ciphertext which include the iv: (iv, **c)**
  - Access to the Padding Oracle
- Attacker's goal
  - The plaintext of (iv, ***c*)**

- Note: the ciphertext is encrypted with a secret key *k*. The Padding Oracle knows *k.*

- Padding Oracle:
  - Query: A ciphertext**.** (the ciphertext is encrypted using *k*)
  - Output: YES, if the plaintext is in the correct "padding" format.
    NO, otherwise

Padding Format

- The block size of AES is 128 bits (or 16 bytes). Suppose the length of the plaintext is 200 bits, it will be fitted into two blocks, with the remaining 56 bits "padded" with some values.



- There are many ways to fill in the values. In any case, an important piece of information must be encoded: the number of padded bit. If this info is missing, the receiver will not know the length of the plaintext.

- Next slide give a "standard" on the padding format.

Padding - PKCS#7

- PKCS#7 is a padding standard.

Read https://en.wikipedia.org/wiki/Padding_(cryptography)#PKCS7

- The following example is self-explanatory.

Suppose the block size is 8 bytes, and the last block has 5 bytes (thus 3 extra bytes required), padding is done as follow:

DD DD DD DD DD DD DD DD	DD DD DD DD DD **03 03 03**

- In general, the padding are:

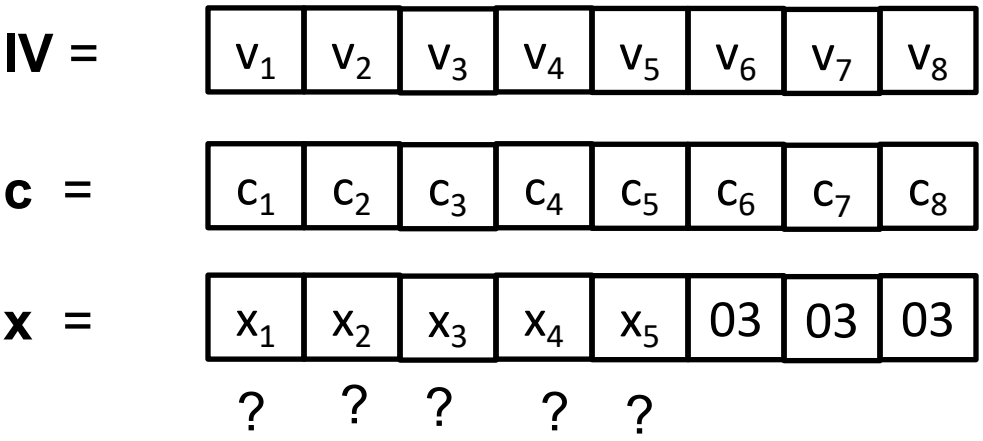
01
02 02
03 03 03
04 04 04 04 etc.

- If the last block is full, i.e. it has 8 bytes, an extra block of all zeros is added.

Padding oracle attack on AES CBC mode.

- AES CBC mode is not secure against padding oracle attack, (when padding is done with PKCS#7).

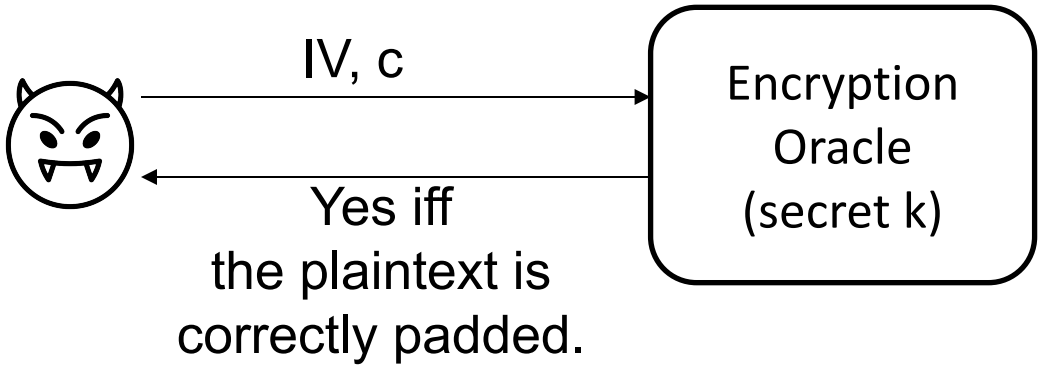
- Let us look at this example:
  - Attacker has (**IV** ‖ **c**), which is one block of IV, and 1 block of **c**. For convenience, let us assume that the attacker knows that the block is padded with 3 bytes, that is, the length of plaintext is 5 bytes.
  - The attacker wants to find the value of x_5.
  - The attacker can send any (iv, ciphertext) to the oracle, and the oracle will tell the attacker whether the decrypted plaintext is padded correctly.



Padding oracle attack on AES CBC mode.

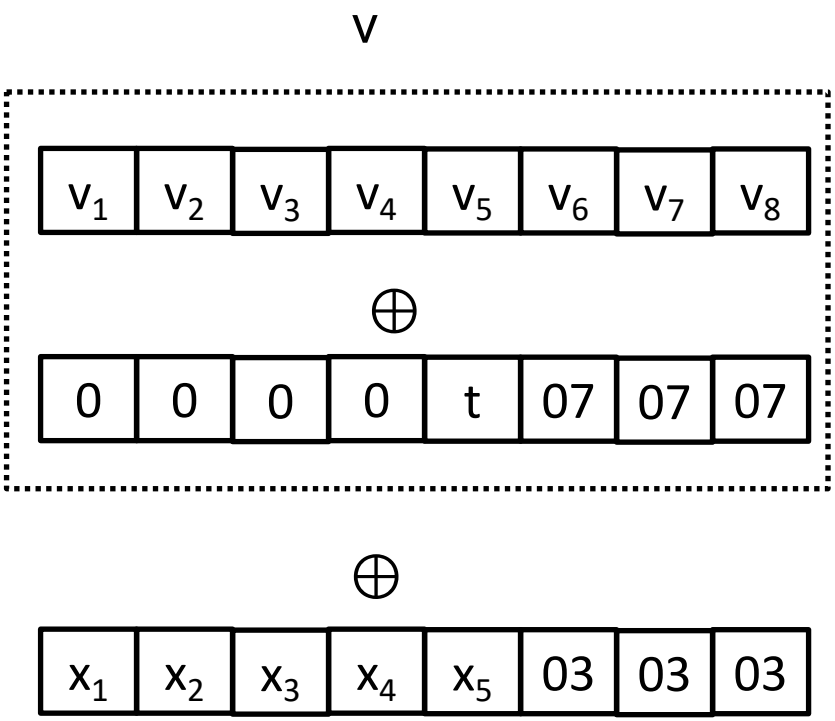
- This algorithm outputs the value of x_5.

1. For *t* = 0 to 255 // *binary representation*
2. Let **v** = IV ⊕ | 0 | 0 | 0 | 0 | t | 07 | 07 | 07 |
3. Sends the two-block query (**v** ‖ **c**) to ***Oracle***.
4. If ***Oracle*** gives **YES**, then outputs (04 ⊕ t)
5. End-for-loop

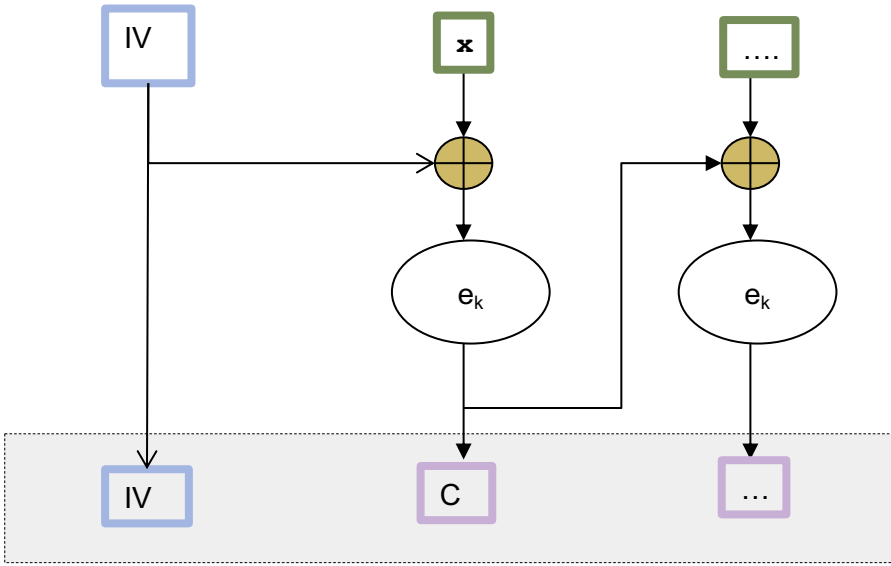


Why it works?

- Note: 07 ⊕ 03 = 04



- CBC mode



Remarks

- We can easily extend the algorithm to find all the plaintext.
- The algorithm need to know the plaintext's length. It is possible to determine the length (exercise).
- This attack is practical. There are protocols (*which are interactions between two or more entities*) between a client and server which performs this:

  If the client submits a ciphertext whose plaintext is not padded correctly, the server will reply with an error message.

- Now if an attacker obtained a ciphertext, the attacker can carry out the protocol with the server so as to get the plaintext.

The attack illustrates:

- The notion of *Oracle*.

- There are situations where seemingly useless information can be useful. So, leakage of any “bit” of information should be avoided.

1.5 Cryptography Pitfalls

A secure encryption scheme can be vulnerable if not implemented properly. This section gives some examples.

1.5.1 – Wrong choice of IV, reusing one-time-pad

1.5.2 – Random number is predictable

1.5.3 – Modify existing or make your own encryption scheme

1.5.4 – Reliance on Obscurity: Kerckhoff's principle.

(to be studied later)

– Using encryption for the wrong purpose (e.g. using encryption scheme to ensure message integrity)

– Side Channel Attack

1.5.1 Wrong choices of IV. Reusing one-time-pad

Wrong choices of IV

Some applications overlooked IV generation. As a result, under some situations, the same IV is reused.

- E.g. To encrypt a file F, the IV is derived from the filename. It is quite common to have files with the same filename.

  (Read **Schneier on Security, Microsoft RC4 Flaw.**

  https://www.schneier.com/blog/archives/2005/01/microsoft_rc4_f.html

  http://eprint.iacr.org/2005/007.pdf)

- E.g. When using AES under the "CBC mode", the IV has to be unpredictable to prevent a certain type of attack. (So, it is vulnerable to choose IV as 1,2,3,....)

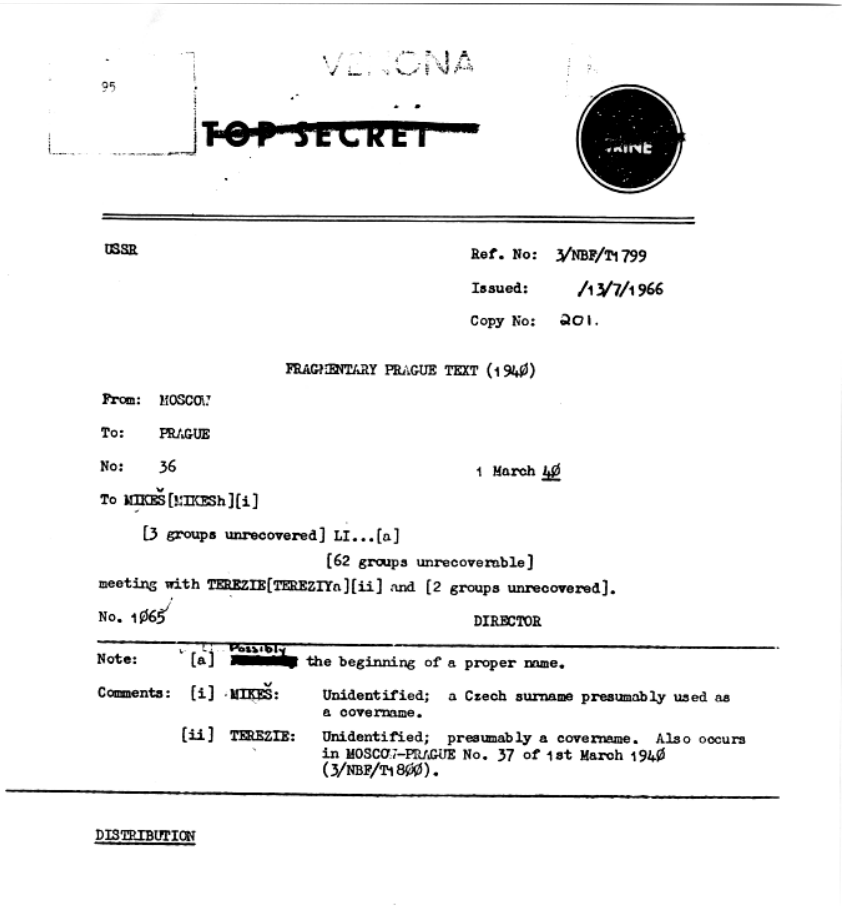
  The well-known BEAST attack exploits this.

  (optional: http://resources.infosecinstitute.com/ssl-attacks/)

Reusing one-time-pad

- The Verona project is a classic example on such failure.

(optional: https://www.nsa.gov/about/_files/cryptologic_heritage/publications/coldwar/venona_story.pdf)



VENONA

95

TOP SECRET

USSR

Ref. No: 3/NBE/T1799

Issued: /13/7/1966

Copy No: 201.

FRAGMENTARY PRAGUE TEXT (1940)

From: MOSCOW

To: PRAGUE

No: 36 1 March 40

To MIKEŠ[MIKESh][i]

[3 groups unrecovered] LI...[a]

[62 groups unrecoverable]

meeting with TEREZIE[TEREZIYa][ii] and [2 groups unrecovered].

No. 1065 DIRECTOR

Note: [a] Possibly [redacted] the beginning of a proper name.

Comments: [i] MIKEŠ: Unidentified; a Czech surname presumably used as a covername.

[ii] TEREZIE: Unidentified; presumably a covername. Also occurs in MOSCOW-PRAGUE No. 37 of 1st March 1940 (3/NBE/T1800).

DISTRIBUTION

1.5.2 Predictable secret generation

(tutorial)

- **Scenario 1**:
  - You are coding a program for a simulation system, for e.g. to simulate road traffic.
  - In the program, you need a sequence of random numbers, for e.g. to decide the speed of the cars.
  - How to get these random numbers?

- **Scenario 2**:
  - You are coding a program for a security system.
  - In the program, you need a random number. For e.g. you need to generate a random number as a temporary secret key.
  - How to get these random numbers?

to be discussed in tutorial

- In Java, what is the different between the following?
  - java.util.Random
  - java.security.SecureRandom

- In C, what is the different between using the following

```
#include <time.h>
#include <stdlib.h>
      srand(time(NULL));
      int r = rand();
```

and a complicated version below?

```
int byte_count = 64;
char data[64];
FILE *fp;
    fp = fopen("/dev/urandom", "r");
    fread(&data, 1, byte_count, fp);
fclose(fp);
```

1.5.3 Designing your own cipher

- Don’t design your own crypto, or even make slight modification to existing scheme... unless you has in-depth knowledge on the topic.

- “Don’t roll your own crypto”

read http://security.stackexchange.com/questions/2202/lessons-learned-and-misconceptions-regarding-encryption-and-cryptology/2210#2210

1.5.4 Reliance on Obscurity: Kerckhnoffs's Principle

Kerckhoffs's principle

A system should be secure even if everything about the system, except the secret key, is public knowledge.

Security through Obscurity

To hide the design of the system in order to achieve security.

Examples (against obscurity)

- RC4 was introduced in 1987 and its algorithm was a trade secret. In 1994, a description of its algorithm was anonymously posted in a mailing group. http://en.wikipedia.org/wiki/RC4

- MIFARE Classic, a contactless smartcard widely used in Europe. It uses a set of proprietary protocols/algorithms. However, they are reverse-engineered in 2007. It turns out that the encryption algorithms are already known to be weak (using only 48bits) and breakable.

  http://en.wikipedia.org/wiki/MIFARE

  Presentation (video) by the researcher who reversed-engineered it:

  (optional) http://www.youtube.com/watch?gl=SG&hl=en-GB&v=QJyxUvMGLr0

  (the algo was revealed at 14:00)

Examples (for obscurity)

- It is not advisable to reveal the computer network structure and settings (for example, location of firewall and the firewall rules), although these are not “secrets”, and many users within the organization may already know the settings.

- Although the algorithm has to be made public, it is not advisable to publish the actual program used in a smart-card. By publishing the program/code, advisory may able to identity implementation flaw that was previously unaware of, or carry out side-channel attacks. Nevertheless, a sophisticated advisory may able to reverse-engineer and obtain the program/code.

- Usernames are not secrets. However, it is not advisable to publish all the usernames.

- In general, obscurity can be used as one layer of a ***defense in depth*** strategy. It could deter or discourage novice attackers, but ineffective against attackers with high skill and motivation.

see

http://technet.microsoft.com/en-us/magazine/2008.06.obscurity.aspx

http://en.wikipedia.org/wiki/Security_through_obscurity

In this module, we always assume that the attackers know the algorithms.

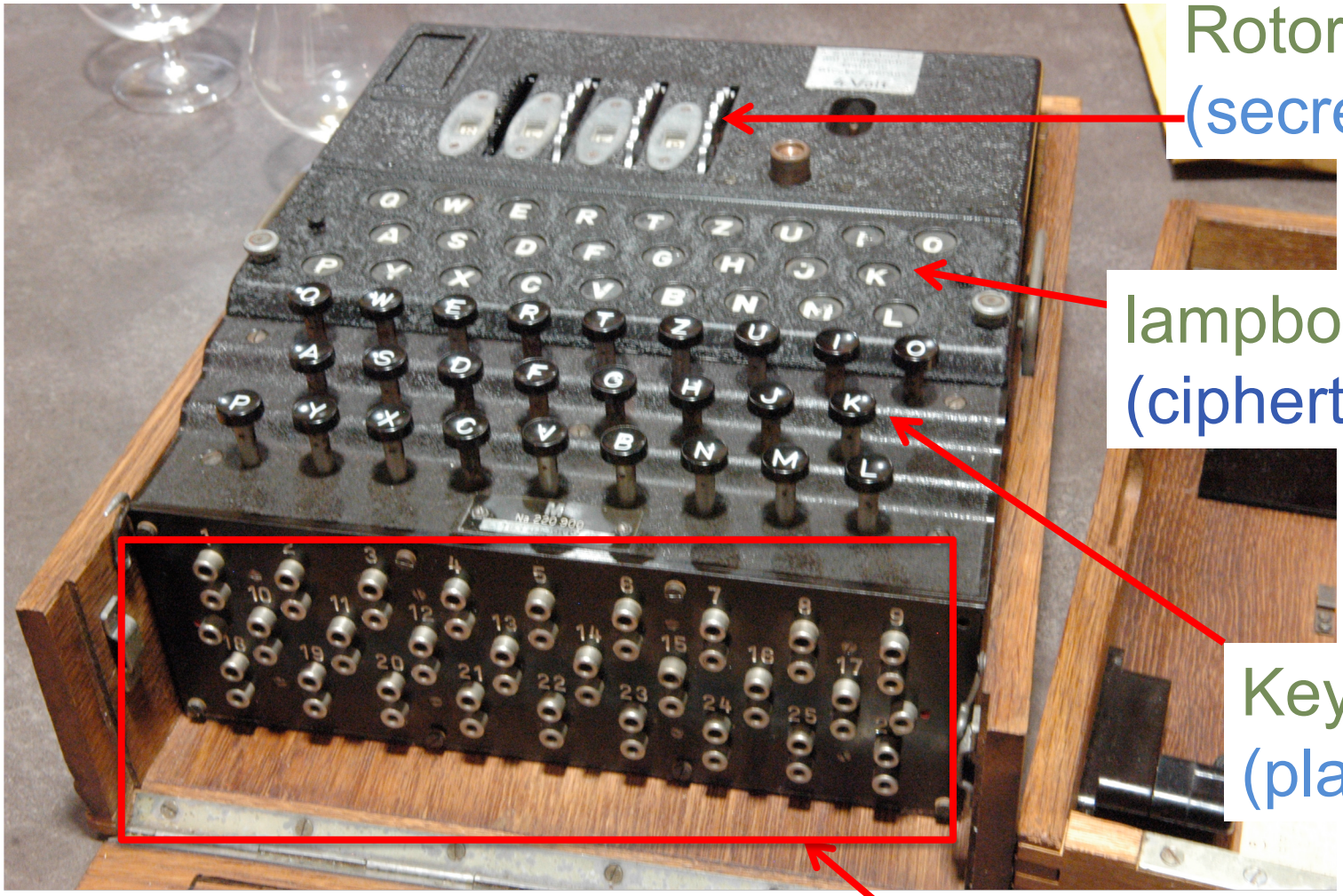
1.6 Some historical Facts

Cryptography: History

- Cryptography is closely related to warfare and can be traced back to ancient Greece. Its role became significant when information is sent over-the-air. Cryptanalysis is one of the driving forces to the invention of computer (e.g. Colossus computer https://en.wikipedia.org/wiki/Colossus_computer).

- WWII: Famous encryption machines include the Enigma, and the Bombe (that helped to break Engima).

Enigma



Rotors
(secret key)

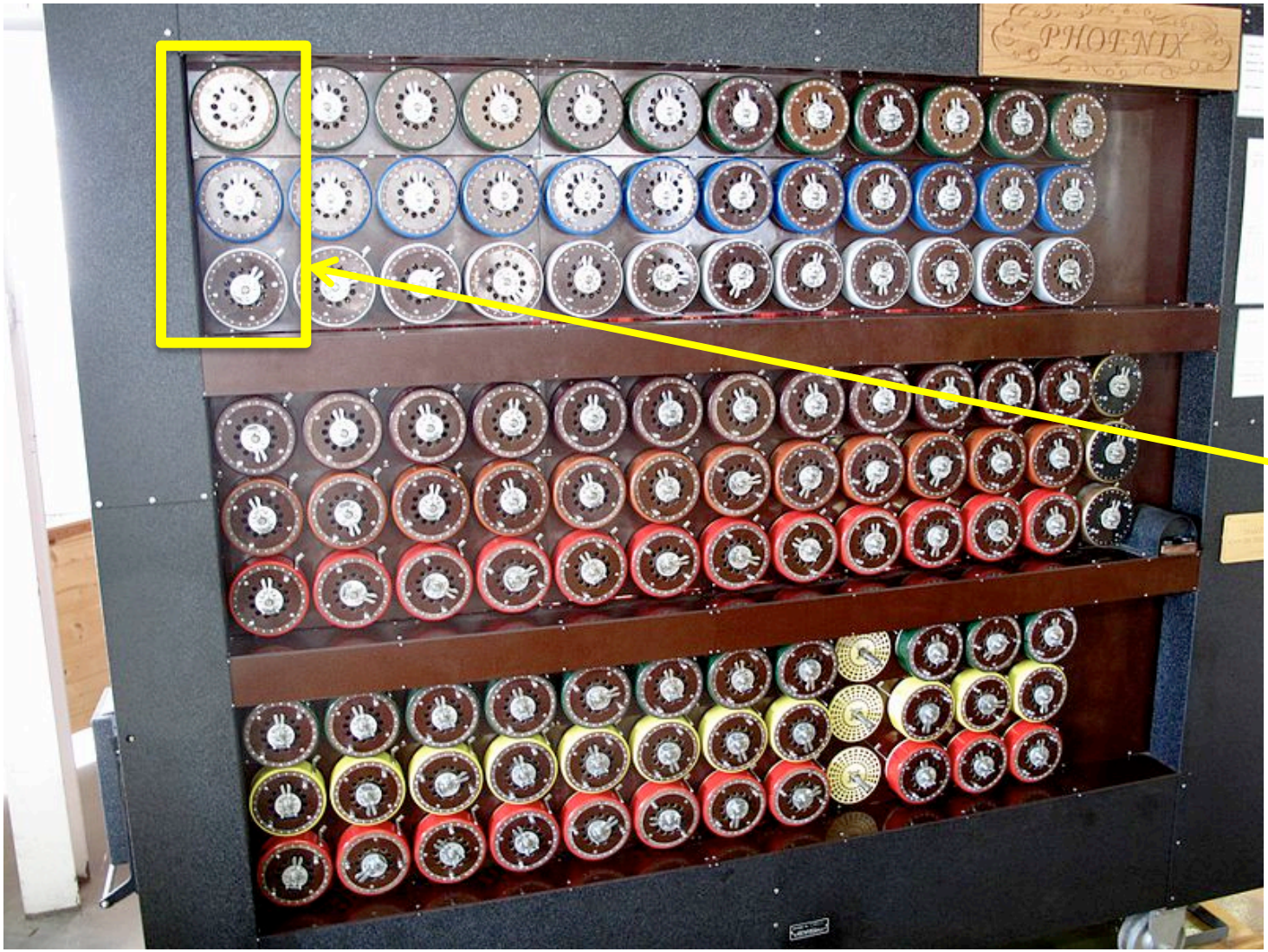
lampboard
(ciphertext)

Keyboard
(plaintext)

plugboard
(secret key)

http://www.enigma-replica.com/Glens_Enigma.JPG

Working rebuilt bombe at Bletchley Park museum.



simulates the 3 rotors in one Enigma machine

http://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma#Crib-based_decryption

Modern Ciphers

DES

- 1977: DES (Data Encryption Standard), 56 bits.

During cold war, cryptography, in particular DES was considered as "Munition", and subjected to export control. (Currently, export of certain cryptography products is still controlled by US.)

Read the section on Singapore in http://www.cryptolaw.org/cls2.htm

- 1998: A DES key broken in 56 hours.

  Triple DES (112 bits) is still in used.

- 2001: AES (Advanced Encryption Standard). NIST. 128, 192, 256 bits.

RC4

- Designed by Ron Rivest (RSA Security) in 1987.
- Initially a trade secret. Algorithm leaked in 1994.
- Used in WEP (for wifi) in 1999. WEP implementation has 40 or 104-bit key. WEP was widely popular.
- 2001: a weakness in how WEP adopts RC4 is published by Fluhrer, Mantin, Shamir.
- 2005: a group from FBI demonstrated the attack.
- Industry switched to WPA2. (with WPA as an intermediate solution).

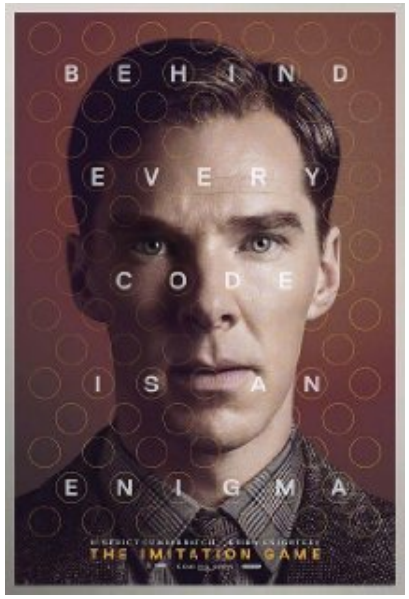
Question:

Bob encrypted a .mp3 music file using winzip, which uses the 256-bit key AES. He choose a 6-digit number as password. Winzip will generate the 256-bit key from the 6-digit password using a "hash" function, say SHA1.

Alice obtained the ciphertext. Alice also knew that Bob used a 6-digit password.

Given a "guess" of the 256-bit key, Alice can determine whether the key can successfully decrypted the file. How many guesses Alice needed to make in order to get the video?

Movie about encryption



The Imitation Game.

During World War II, mathematician Alan Turing tries to crack Enigma with help from fellow mathematicians.

http://www.imdb.com/title/tt2084970/

U-571

Fictional plot on how a U-boat was captured. Actual U-boat captured: U-110, U-505, U-570, U-744, U-1024.

Prize of capturing a u-boat instead of sinking it…. Engima!