Quiz (Substitution Cipher)

- Suppose a substitution cipher works over a set of symbols U with $|U| = 50$
- What is:
  - Its key space size?
  - (The lower bound of) its key size/length?

Shift and Caesar Ciphers

- **Shift cipher**: a type of substitution cipher
- Each letter in the plaintext is “shifted”
  a certain number of places down the alphabet
- Example (with a shift of **1**):
  - A would be replaced by B
  - B would become C
  - ….
  - Z would be replaced by ?
- **Caesar cipher**:
  - A shift cipher with a shift of **3**
  - There is no key, or 3 is the key
- *Question: is shift cipher easier to break than the (general) substitution cipher?*

Quiz (Shift Cipher)

- How about its:
  - Key space?
  - Key space size?
  - Key size (lower bound)?

1.2.2 Vigenere Cipher

Vigenere Cipher

- Substitution cipher is a **monoalphabetic cipher**:
  - The substitution is fixed for *each letter* of the alphabet
  - Different occurrences of a letter have **the same** mapped letter in ciphertext
- Vigenere Cipher is a **polyalphabetic cipher:**
  - Uses a ***keyword*** instead of a single shifting distance in shift cipher:

    a string of letters representing numbers based on their position in the alphabet
  - Example: “SOC” = 18, 14, 2
  - The keyword gets repeated for a longer plaintext
  - The keyword is used to select which shifting distance to be used for each letter of the plaintext
  - Example: “ATTACKATDAWN ” and “ABCD” → “AUVDCLCWDBYQ”
  - Tabula recta (see the next slide)

Vigenere Cipher: Tabula Recta

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

Source: Wikipedia

Security of Vigenere Cipher

- Vigenere cipher is an improvement over shift cipher
- Different occurrences of the same letter can have **different** mapped letters!
- Is it however secure against known-plaintext attack?
  → this is easy to answer: *no!*
- Is it secure against ciphertext-only attack??
  → trickier to answer: need to find *a good attack technique*
- Suppose we know $\boldsymbol{k}$ = the **length/period** of the keyword
- Observation: all letters of the plaintext whose index is i (mod k), for i=0..k-1, get shifted by the same key character
  See again: "ATTACKATDAWN " and "ABCD"
- Can we use our previous **frequency analysis** technique?
- Vigenere cipher turns into a monoalphabetic cipher *again*

Cryptanalysis of Vigenere Cipher

- Question: *How can we determine the period of the keyword?*

- **Kasiski method:** Babbage (1854) and Kasiski (1863)
- Repetitions in the ciphertext **give clues** to the period
- Having the same letter-block at a period apart results in the same letter-block in the ciphertext

- Example: WBLBXYLHRWBLWYH
- Possible keyword period: 9? 3?

- Hence, if we find repeated patterns/letter-blocks with distance m, guess the period k, where $k|m$

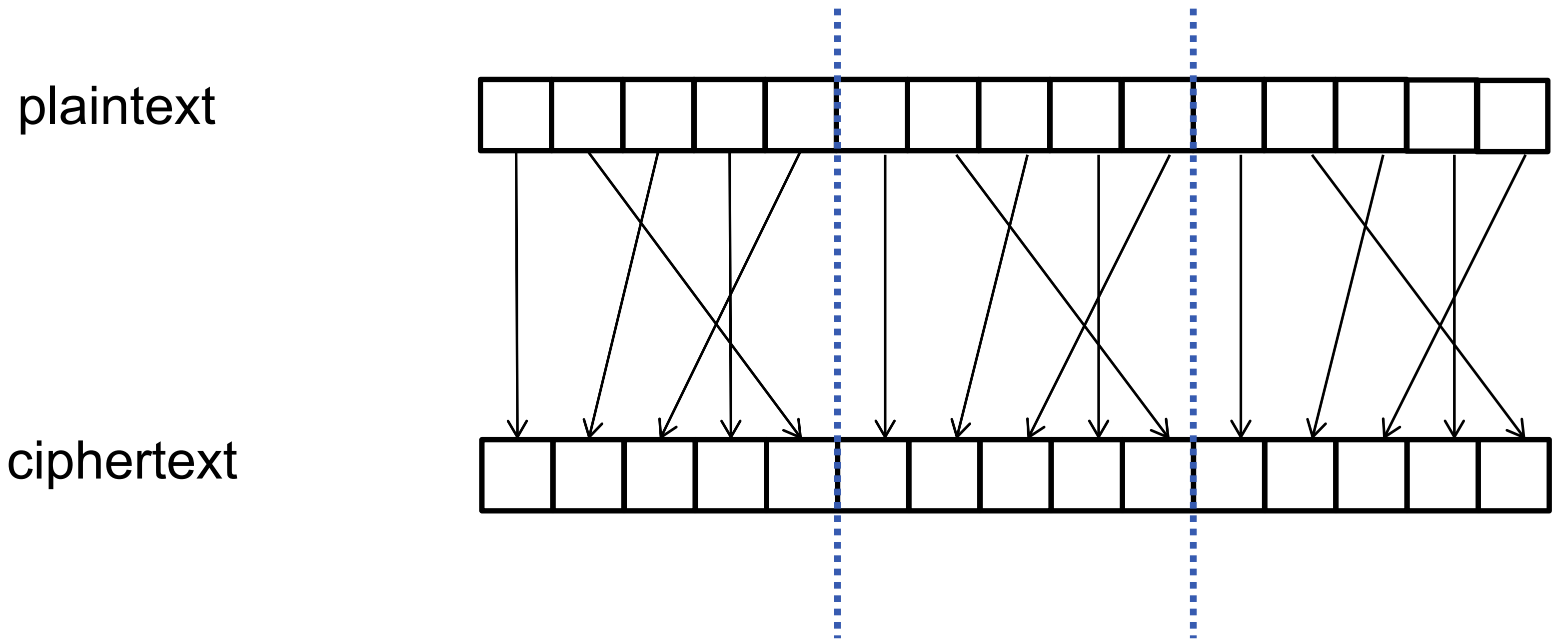
1.2.3 Permutation Cipher

Permutation Cipher

- Also known as **transposition** cipher
- It first groups the plaintext into blocks of $\boldsymbol{t}$ characters, and then applied a secret "permutation" to the characters in **each** block by shuffling the characters
- The key is the secret "permutation":
  an 1-1 onto function e from $\{1,2,..,t\}$ to $\{1,2,...,t\}$
- We can write the permutation $\boldsymbol{p}$ as a sequence
  $$\boldsymbol{p} = (p_1, p_2, p_3, \ldots, p_t),$$
  which shifts the character at position/index i within the block to the position p_i
- The block size $\boldsymbol{t}$ could be part of the key:
  $\boldsymbol{t}$ is also kept secret

Permutation Cipher

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



Cryptanalysis (Known-Plaintext Attack)

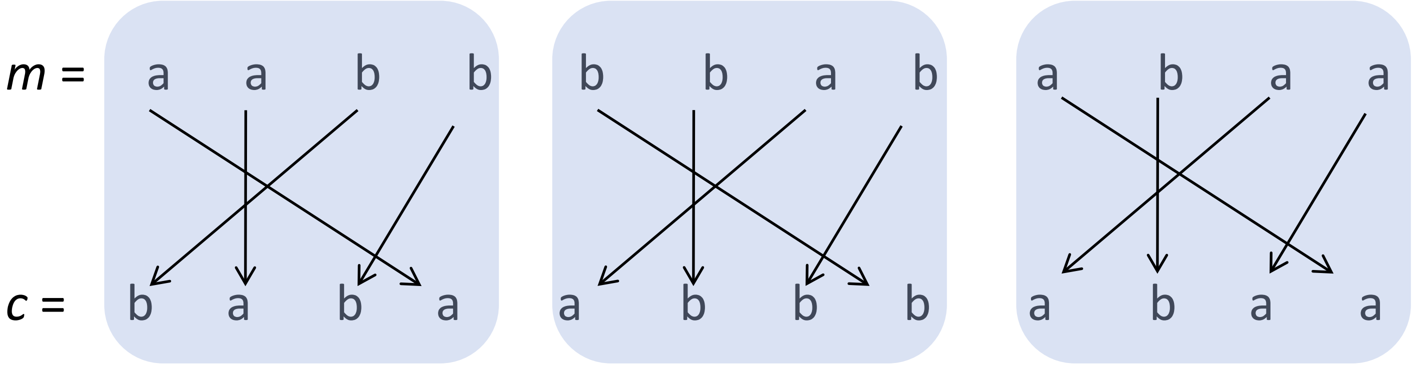
- Permutation cipher fails miserably under **known-plaintext attack**
- Given a plaintext and a ciphertext, it is very easy to determine the key
- Example:

$$m = \text{a a b b b b a b a b a a}$$

$$c = \text{b a b a a b b b a b a a}$$

- Question:
  - In the above, what is the block size *t?*
  - What is the permutation?

Solution



- How about **ciphertext-only attack**?
  Permutation cipher can be easily broken if the plaintext is an English text

1.2.4 One Time Pad

One-Time-Pad

- **Encryption**:

  Given an n-bit plaintext: $x_1x_2....x_n$ and n-bit key: $k_1k_2....k_n$,

  output the ciphertext:

  $$C = (x_1 \text{ xor } k_1)\ (x_2 \text{ xor } k_2)\ (x_3 \text{ xor } k_3)\ \ldots\ (x_n \text{ xor } k_n)$$

- **Decryption**:

  Given an n-bit ciphertext: $c_1c_2....c_n$ and n-bit key: $k_1k_2....k_n$,

  output the plaintext:

  $$X = (c_1 \text{ xor } k_1)\ (c_2 \text{ xor } c_2)\ (c_3 \text{ xor } k_3)\ldots(c_n \text{ xor } k_n)$$

- In short:

  Encryption: plaintext $\oplus$ key $\rightarrow$ ciphertext

  Decryption: ciphertext $\oplus$ key $\rightarrow$ plaintext

Xor (Exclusive Or) Operation

- Xor operation: $A \oplus B = (A+B) \bmod 2$
- Xor table:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

- 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 Example

encryption

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

decryption

- Why does its decryption process works?
- For any x, k: $(x \oplus k) \oplus k$

$$= x \oplus (k \oplus k) \qquad \text{[by associativity]}$$

$$= (x \oplus 0) \qquad \text{[by self-inverse]}$$

$$= x \qquad \text{[by identity element]}$$

Security of One-Time-Pad

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

- Note that it is not clear how to apply exhaustive search on one-time pad
- In fact, It can be shown that one-time-pad leaks **no information** of the plaintext, except for its length, even if the attacker has an arbitrary running time (unlimited computing power)
- Hence, it is sometime called "*unbreakable*" or has "*perfect secrecy*"* (provided that a "random" key is used *once*)

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

Security of One-Time-Pad (For Those Who're Curious)

Optional

- How can the one-time pad achieve *perfect secrecy*?
- If the key *k* is random, then the ciphertext *c* looks "*as random as the key*" to an attacker: the XOR of a *random* string with a fixed string yields a *random* string
- In other words:
  - Pr [k[0] = 0] = ½, giving Pr [x[0] = 0] = ½
  - Pr [k[1] = 0] = ½, giving Pr [x[1] = 0] = ½
  - …
- It is *impossible* to learn anything about the plaintext *x* given *c*, even for an attacker with unlimited computing power
- Hence, knowing the ciphertext gives *no information* about the plaintext except its length

Some Remarks

- Yes, one-time-pad can be shown to be "unbreakable"
- However, The key must be at least as long as the message, which is useless in many applications
- Nevertheless, it is practical in some scenarios

  (See http://ciphermachines.com/otp)

http://ciphermachines.com/otp

- Yet, also check "*The Venona Story*", where one-time-pads fails with repeated key ("*two key*" issue)

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

1.3 Definitions & Properties of Cryptosystems (More Formal)

Cryptosystem/Cipher Review

- More formally defined as **algorithms** (**G**, **E**, **D**) over **sets** (***K***, ***M***, ***C***):
  - ***K***: set of all keys (key space)
  - ***M***: set of all plaintexts (plaintext/message space)
  - ***C***: set of all ciphertexts (ciphertext space)
  - G: generates $k \in \boldsymbol{K}$, key-generation algorithm
  - E (Enc): $\boldsymbol{K}, \boldsymbol{M} \rightarrow \boldsymbol{C}$, encryption algorithm
  - D (Dec): $\boldsymbol{K}, \boldsymbol{C} \rightarrow \boldsymbol{M}$, decryption algorithm
- Requirements:
  - **Correctness**: For all $m \in \boldsymbol{M}$ and $k \in \boldsymbol{K}$ outputted by G, $D_k(E_k(m)) = m$
  - **Efficiency**: G, E & D are "fast" enough, i.e. polynomial time
  - **Security**: *How can we define it more clearly, and say better than "attackers cannot recover the secret key or plaintext"*

Security Guarantee: Perfect Secrecy (Intuitively)

Attacker's ***prior knowledge*** of the unknown plaintext *m* (before knowing *c*)

- **Perfect security/secrecy** ("absolute security"):
  - Informally: "regardless of any prior information that attackers (with ***unlimited computational power***) has about the plaintext, the ciphertext should leak ***no additional information*** about the plaintext
  - More formally: for all $m \in \boldsymbol{M}$ and all $c \in \boldsymbol{C}$ (with $\Pr[C=c]>0$)

    $$\Pr[M=m \mid C=c] = \Pr[M=m]$$
  - **Issue**: the key must be (at least) as long as the message itself → impractical in practice
  - *Important questions*:
    - Is it unnecessarily strong for practical usage?
    - Any security notion that is **more relaxed and practical**?

Optional

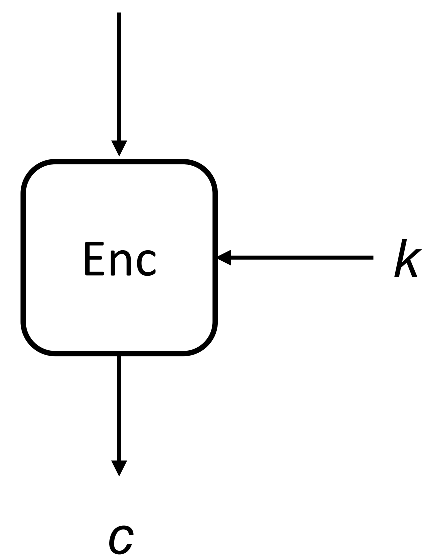
Attacker's ***updated knowledge*** of the unknown plaintext *m* after the attacker has seen the ciphertext *c*

So, the attacker gained ***no additional information*** at all!

A Good Analogy

- Bob gathers information from many sources, and believes that the movie *"Alice in Crypto Land"* has **42%** chance of winning.
  Based on that, Bob bet in some online site
- Let ***c*** be the ciphertext of the Oscar's winner movie, encrypted using some secret key
- Now, suppose Bob somehow obtains *c*.
  With the additional knowledge of *c*,
  can now Bob improve his guess from 42%?
  Should Bob change his bet?
- Let's assume that Bob have **extremely powerful machine** that can exhaustively search all keys.
  Can he improve the guess?
  - *If the encryption scheme used is AES, the answer is yes! How?*
  - *if the encryption scheme achieves perfect secrecy, the answer is no!*

m = "And the Oscar goes to xxxxxxxxx"



Security Guarantee: Computational Secrecy

- **Computational security/secrecy** ("hardness security"):
  - More relaxed and practical than perfect secrecy
  - Informally: it is still fine if a cipher leaks some information ***with tiny probability*** to attackers with ***bounded computational resources***

- Impacts:
  - Security **may fail** with tiny probability:
    - Very very small probability, e.g. 2^{-60} : probability of rare events
    - Usually considered "**negligible**" enough in practice
  - Against **computationally bounded** attackers:
    - If 1 key is tested per clock cycle,
      a supercomputer can check ~2^{80} keys/year
    - A supercomputer since Bing Bang can check ~$2^{\mathbf{112}}$ keys
    - Key space size of modern ciphers ≥ $2^{\mathbf{128}}$ keys
    - See also some calculation exercises in our Tutorials

Security Guarantee: Computational Secrecy

- Important question:
  can we *accept (live with)* computational secrecy in practice?
- Usually based on a known **hard problem**:
  serves as the **computational assumption**
- The computational assumption is very important:
  - It enables “proof” of security:
    the “proof” is relative to the assumption(s)
  - Some implications if the assumption is proven wrong/invalid
  - It allows for some meaningful comparison of different ciphers

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
  - 2^{31} times on average
- We can **quantify the security** of an encryption scheme based on the *length (number of bits) of the key*
- Example:
  - Comparing a scheme *A* with 64-bit keys and a scheme *B* with 54-bit keys, then scheme *A* is more secure w.r.t. exhaustive search

Exhaustive Search and Key Length (See Work Factor [PF2.3page91])

- ***Additional Note*:** 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 exhaustive search**
- For example, in the best known attack on a 2048-bit RSA, roughly 2^{112} searches are required:
  - Hence, we treat its security as equivalent to 112 bits
  - So, we say that the 2048-bit RSA has key strength of 112 bits

- Question: How **many bits** are considered "secure"?
- Answer: See Tutorial 1
- Also *read* NIST Recommended key length for AES http://www.keylength.com/en/4/

Security Analysis of a Cipher

- Security analysis of a cipher is typically formulated *with respect to*:
  - **Threat model**:
    - Assumptions about **attackers' capabilities**,
      i.e. amount of information available to the attacker,
      what the attacker can & can't do with its computing resources
    - Categorized into: *black-box models* and *gray-box models*
  - **Security guarantee/goal**:
    intended attacker's <u>***in***</u>**-capabilities**,
    i.e. what we do *not* want the attacker to accomplish

Black-Box Threat Models

- Attackers only see what **goes in and out** of a cipher
- Can be abstracted by ***query operations*** to the cipher
- A ***query*** *to a* cipher: the operation that sends an input value to a cipher and gets the output in return, without:
  - Exposing the details of the cipher
  - Without revealing the secret key
- ***An encryption query***:
  takes a plaintext and returns its ciphertext
- ***A decryption query***:
  takes a ciphertext and returns its plaintext

Black-Box Threat Models

Attackers' capabilities (from the weakest to the strongest):

- ***Ciphertext-only attackers (COA)***: can see the ciphertexts only
- ***Known-plaintext attackers (KPA)***: can observe ciphertexts, and do know the associated plaintexts
- *Chosen-plaintext attackers (CPA)*:
  - Can perform encryption queries for plaintexts of their choice, and observe the resulting ciphertexts
  - Controls access to the **encryption engine** (*but* not the key used) as an ***encryption oracle***, e.g. encrypted text sent from a user's terminal
- *Chosen-ciphertext attackers (CCA/CCA2)*:
  - Can perform both encryption and decryption queries
  - CPA + (partial/full) access to the decryption engine as a ***decryption oracle***

Optional

How do you visualize different types of attackers?

Black-Box Threat Models: Comparison

Attackers Model	Observe Ciphertexts	Know the Corresponding Plaintexts	Can Perform/ Influence Encryption Queries	Can Perform/ Influence Decryption Queries
Ciphertext-Only Attackers	Yes	No	No (passive)	No (passive)
Known-Plaintext Attackers	Yes	Yes	No (passive)	No (passive)
Chosen-Plaintext Attackers	Yes	Yes	Yes	No
Chosen-Ciphertext Attackers	Yes	Yes	Yes	Yes

Gray-Box Models

- The attacker has **access to** a cipher's implementation
- More realistic than black-box models for smart cards, embedded systems, virtualized systems
- ***Side-channel attacks***:
  - *A **side channel***: a source of information that depends on the implementation of the cipher
  - Side-channel attackers simply **observe/measure** analog characteristics of a cipher's implementation,
    but don't alter its integrity → *non-invasive*
  - **Typical side channels**: the execution time, power consumption, electromagnetic emanations, acoustic noise, …
- ***Invasive attacks***:
  - More powerful than side-channel attacks
  - More expensive than side-channel attacks: sophisticated equipment is required, e.g. high-resolution microscopes, a chemical lab, …

Attacker’s In-capabilities

- What we do not want the attacker to accomplish
- The flip side: **attacker’s goals**
- If an attacker wants to find the key, we call this goal as ***total break***
- Alternatively, the attacker may be satisfied with *a* ***partial break*****:**
  - The attacker may just want to **recover the plaintext from a ciphertext** (*but* not interested in knowing the secret key); or
  - The attacker may want to determine some **coarse information** about the plaintext (e.g., whether the plaintext is a JPEG image or a C program)
- The *least* the attacker can accomplish is ***indistinguishability of ciphertext***: with some “non-negligible” probability >½, the attacker can **distinguish** the ciphertexts of a given plaintext (e.g., “Y”) from the ciphertexts of another plaintext (e.g., “N”)

1.4 Modern Ciphers: Stream Cipher

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

Symmetric-key based modern ciphers:

stream cipher and block cipher

Modern Ciphers

- Designs of modern ciphers take into considerations of known-plaintext-attack, frequency analysis, and other known attacks
- Examples: DES (Data Encryption Standard, 1977)
  RC4 (Rivest's Cipher 4, 1987)
  A5/1 (used in GSM, 1987)
  AES (Advanced Encryption Standard, 2001)
  A5/3 or KASUMI (3G), SNOW 3G (LTE)
- 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 pages on RC4, A5/1 and DES give quite good descriptions. AES is believed to be secure, and classified as "Type 1" by NSA https://en.wikipedia.org/wiki/NSA_cryptography#Type_1_Product.)

Illustration of a Block Cipher

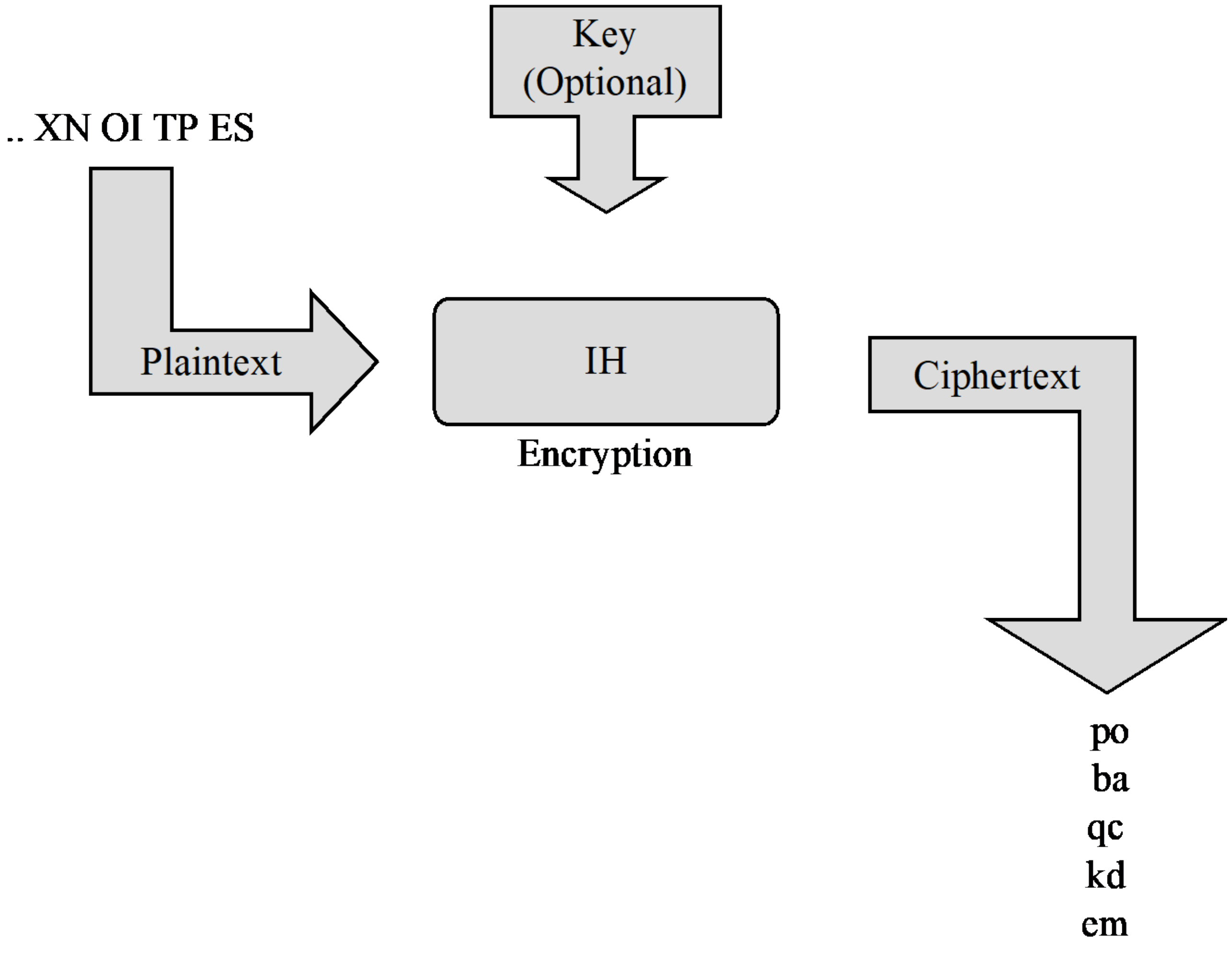
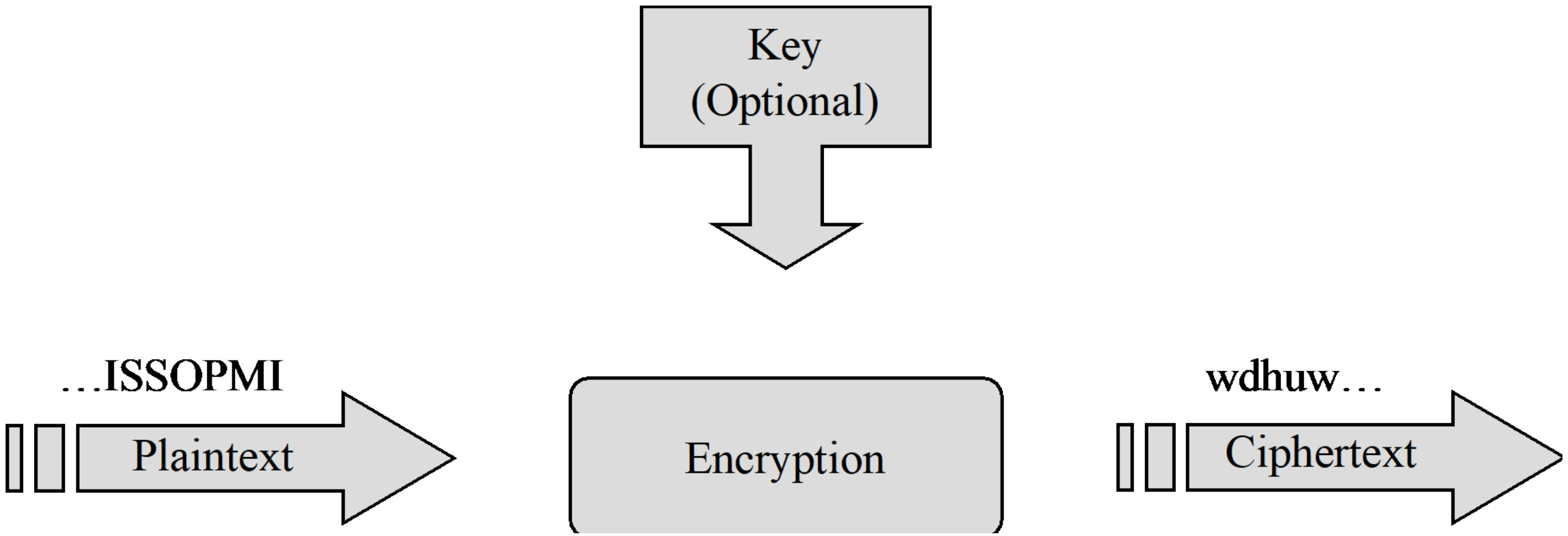


Illustration of a Stream Cipher



Stream vs Block Ciphers

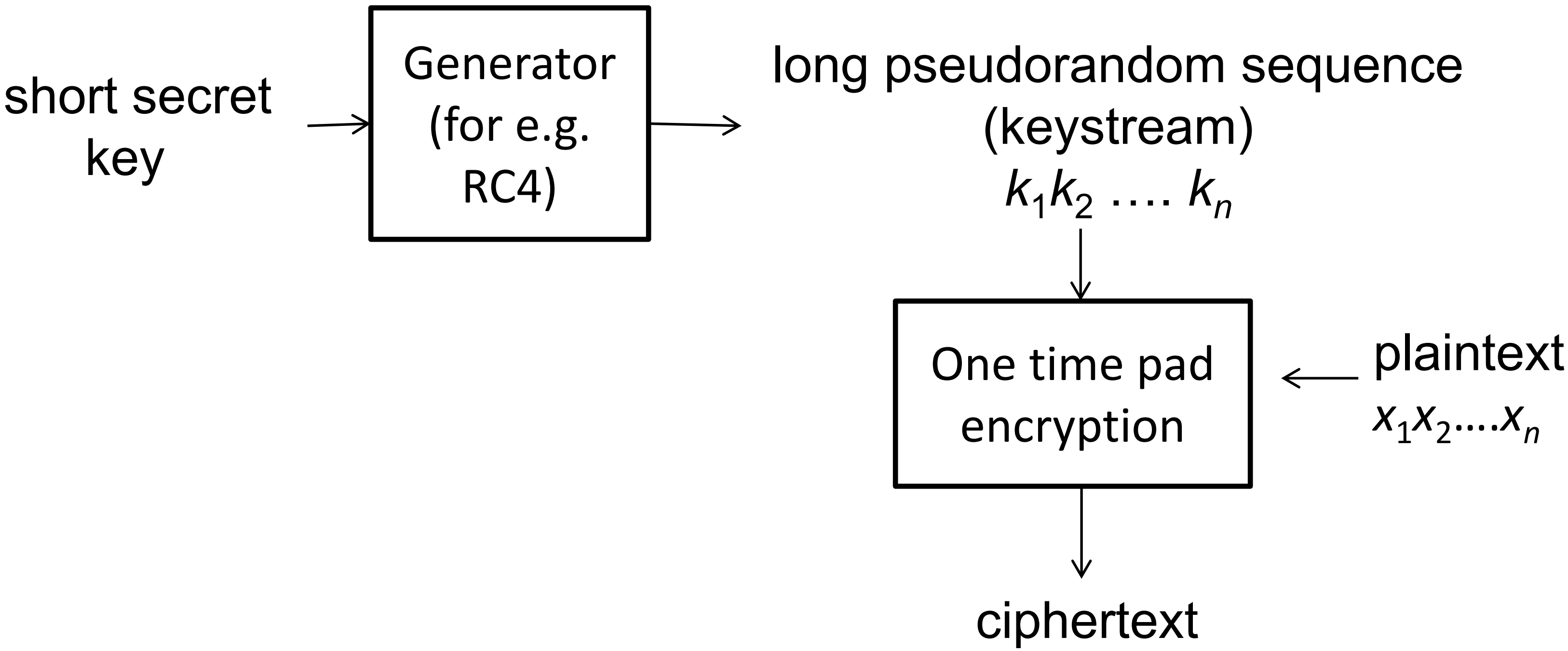
	Stream	Block
Advantages	• Speed of transformation • Low error propagation	• High diffusion • Immunity to insertion of symbol
Disadvantages	• Low diffusion • Susceptibility to malicious insertions and modifications	• Slowness of encryption • Padding • Error propagation

Stream Cipher

- Stream cipher is inspired by one-time-pad: "pseudo/simulated OTP"
- 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***
- The pseudorandom sequence is sometimes also called ***keystream***

Stream Cipher

- Visually:



Pseudorandom Sequence

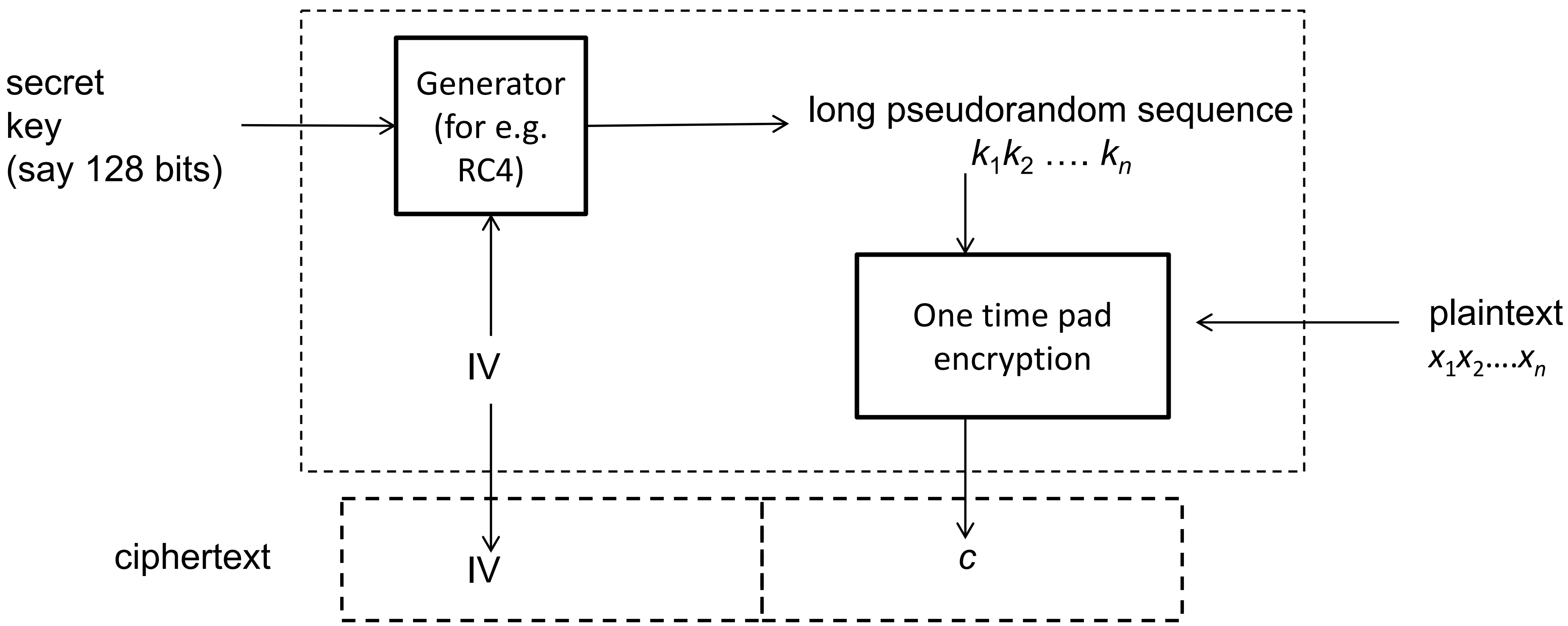
Optional

- Pseudorandomness:
  - An importance building block (primitive) for encryption
  - An important concept in cryptography as well
- Randomness:
  - A property of a **distribution** over elements in a set
- Pseudorandom:
  - (Informally) it **cannot be distinguished** from random/uniform
- Non-cryptographic pseudorandom:
  - Must pass various statistical tests
  - It is still **insufficient** in an adversarial setting
- **Cryptographic (cryptographically-secure)** pseudorandom:
  - Must pass all efficient statistical tests
  - It is also "unpredictable"

Pseudorandom Generator

- The security of a stream cipher relies on the used *generator*: technically known as *pseudorandom generator* (**PRG**) or *pseudorandom number generator* (**PRNG**)
- A function **PRG**: $\{0,1\}^s \rightarrow \{0,1\}^n$, with $n >> s$
- Properties:
  - PRG maps the **seed space** to the **output space** ***deterministically*** and efficiently
  - A selected seed must be random
  - The output “looks random” (i.e. cryptographically secure PR)
- It deterministically expands a short, random seed into a longer, pseudorandom output
- It is useful when we have a “small” no of true/good random bits, and wants a lot of “random-looking” bits
- **Issue**: the same seed generates the same keystream (***two-key issue in one-time pad)*** !

Stream Cipher with Initial Value (IV)



Important question: What if IV is omitted in the ciphertext?

Stream Cipher with Initial Value (IV)

- Stream cipher typically has an *Initial Value or Initialization Vector* (IV)
- The IV can be randomly chosen or from a counter
- A long pseudorandom sequence is generated from the secret key together with the IV: the generator gets **randomized**!
- The final **ciphertext contains the IV**, followed by the output of the one-time-pad encryption
- ***Note***: In some documents, the term “ciphertext” does not include the IV. In any case, the IV **must** appear in clear, and every entity can see it.
- 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

Example

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 generate an IV, say: IV = 0 0 0 1

Step 2: From the short key and IV, generate a 15-bit sequence

K = 0 1 1 0 1 0 1 0 0 1 0 0 1 1 0

Step 3: Output IV, followed by K xor X

0 0 0 1	0 1 1 0 1 1 0 1 1 0 0 0 1 1 0

Decryption:

Given the short key and the ciphertext with the IV

Step 1: Extract the IV from the ciphertext

Step 2: From the short key and IV, generate the long sequence K

Step 3: Perform xor to get the plaintext

Why IV?

- Suppose there isn't an IV
(or set the IV to be always a string of 0's)
- Consider the situation where the same key is used to encrypt two different plaintexts:

$$X = x_1\, x_2\, x_3\, x_4\, x_5 \quad \text{and}$$
$$Y = y_1\, y_2\, y_3\, y_4\, y_5$$

- Further, suppose that an attacker eavesdrops and obtains the two corresponding ciphertexts U, V

Why IV?

- The attacker can now compute:

$$U \oplus V = (X \oplus K) \oplus (Y \oplus K)$$

- By associative and commutative properties of xor:

$$(X \oplus Y) \oplus (K \oplus K) = X \oplus Y$$

- So, from ciphertexts U and V, the attackers can obtain information about $X \oplus Y$, i.e. the following sequence:

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

- ***Question***: What's the big deal about revealing $X \oplus Y$?

What's the big deal about revealing $X \oplus Y$?

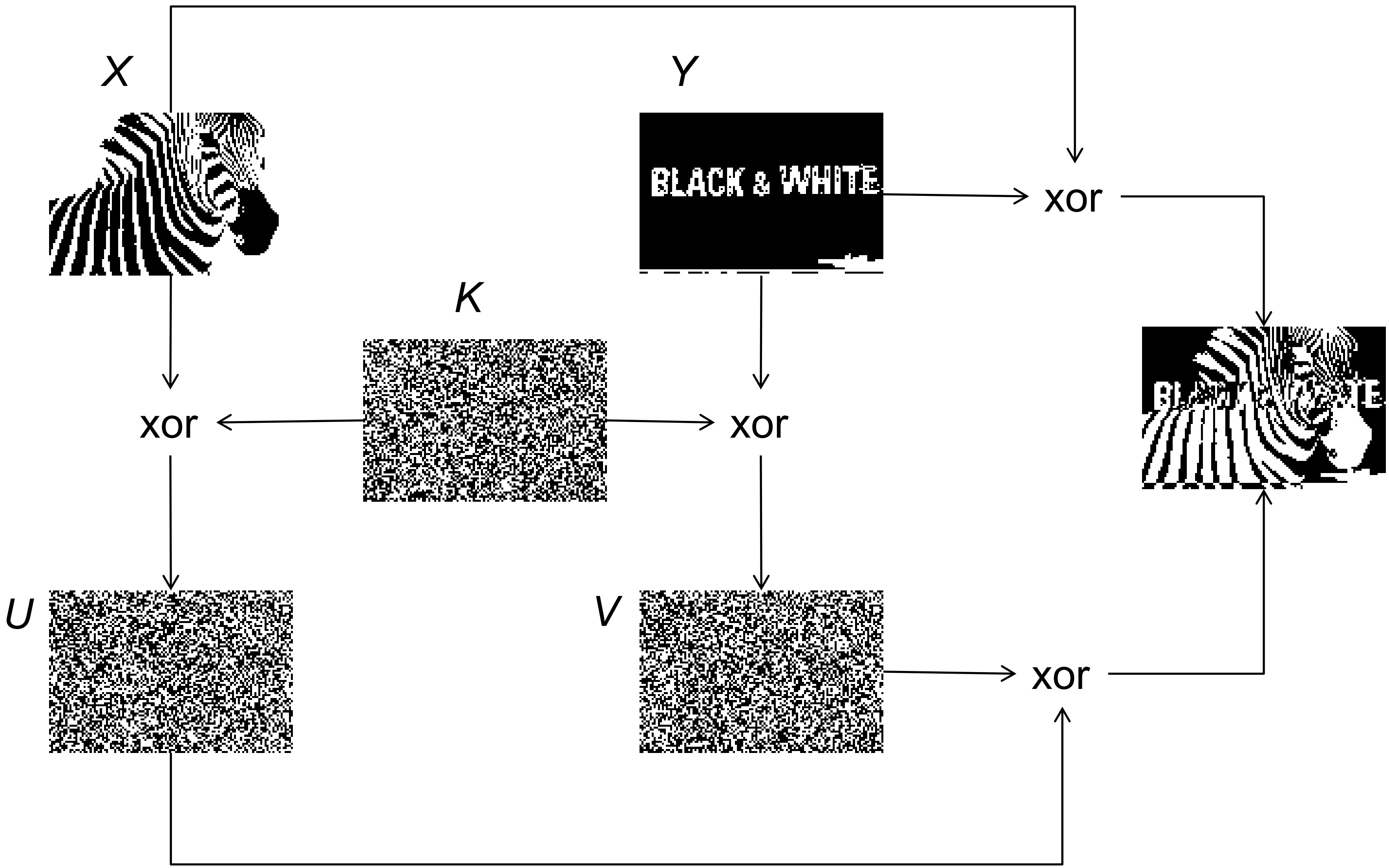
- Suppose X is a 80x120 image of an animal.
  Each pixel is either black (0) or white (1).
  The image can be represented as a (80x120)-bit sequence, where each bit corresponds to a pixel.

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

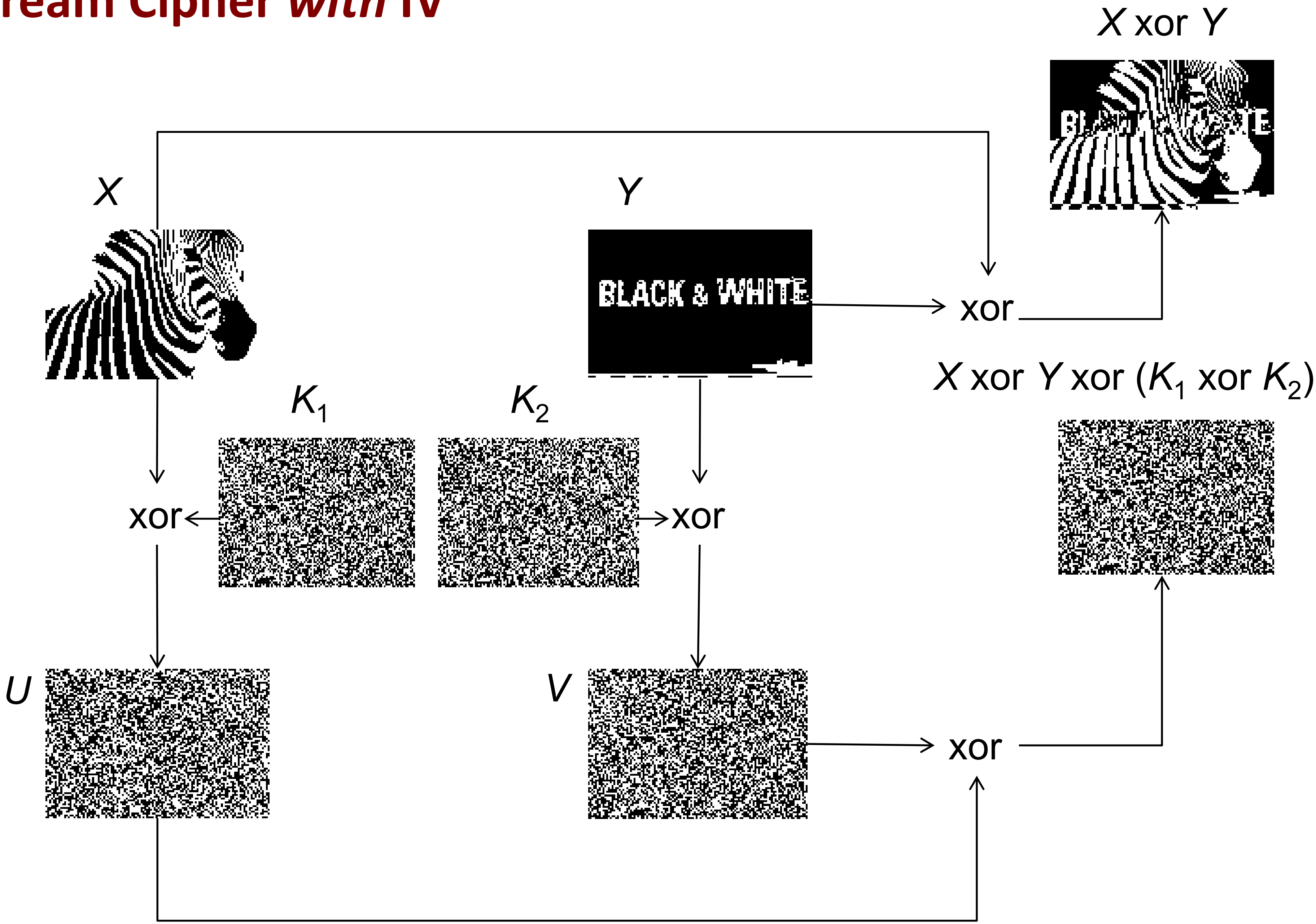
- Here is $X \oplus Y$:

- So, what is X, Y?

Stream Cipher *Without* IV



Stream Cipher *with* IV



Note: On this slide, the ciphertexts/keys are “artist’s impression”

Role of IV

- During 2 different processes of encryption of the same plaintext:
  - The IVs are likely to be **different**
  - The **2 pseudorandom sequences** are thus likely to be different
  - The **ciphertexts** are also likely to be different
- The IV makes an encryption "probabilistic"
- Furthermore, the xor-ing of two ciphertexts would ***not*** cancel the effect of the pseudorandom sequences, and reveal information of the plaintexts!
- Note that IV is not only used in stream cipher, but also adopted in other ciphers

[***Note***: In many documents, the description of ciphers (such as AES, RSA) does not include the IV, and thus doesn't specify how the IV is to be chosen.
Typically, the IV is mentioned during discussion of implementation.
E.g. for AES, the IV is only included in the **"mode of operation"** of AES.]

Sample Stream Ciphers

- Broken stream ciphers (do not use them):
  - RC4 (more later)
  - Content Scramble System (CSS): https://en.wikipedia.org/wiki/Content_Scramble_System
- Examples of modern stream ciphers:
  - Salsa20 & ChaCha: https://en.wikipedia.org/wiki/Salsa20
  - Sosemanuk: https://en.wikipedia.org/wiki/SOSEMANUK
  - See **eStream project** organized by the EU ECRYPT network: https://en.wikipedia.org/wiki/ESTREAM

LumiNUS Forum Challenge: Just for a *Good Fun*

- "**Breaking substitution cipher**" challenge in LumiNUS forum
- You'll be given a ciphertext
- Do break the substitution cipher by finding the correct corresponding plaintext
- You can also refer to the uploaded **"Self-Exploration Activity 1"**

- The ***first person*** who can post the correct plaintext will get **3 (three) extra marks** for assignment (capped at 25)!
- The challenge will be posted **this evening (20 Aug)** at **~9pm**