CS 419: Computer Security

Week 6: Cryptography
Part 1: Introduction
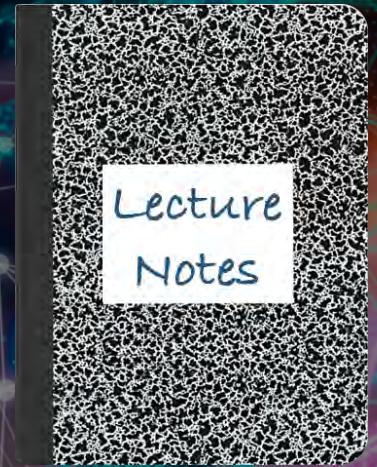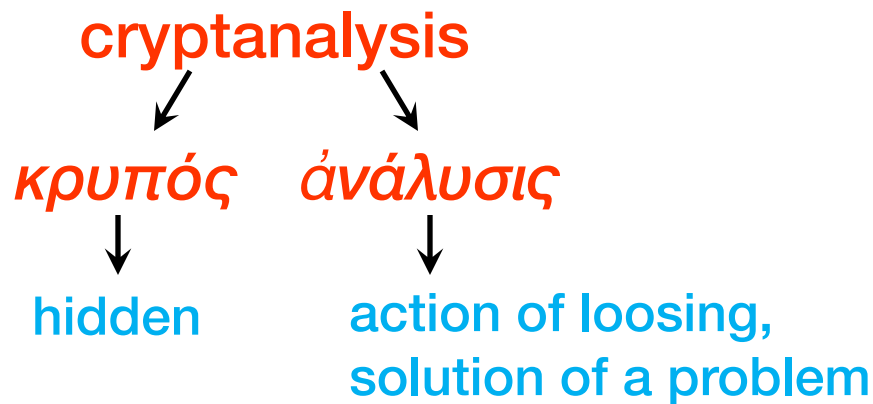
Lecture Notes

Paul Krzyzanowski

cryptography

κρυπός   γραφία

hidden   writing
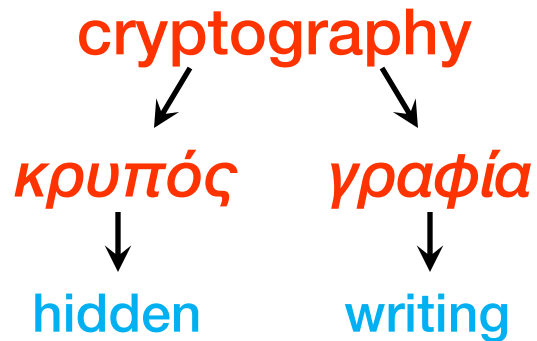
A secret manner of writing, … Generally, the art of writing or solving ciphers.

— *Oxford English Dictionary*

cryptanalysis

κρυπός   ἀνάλυσις

hidden   action of loosing,
solution of a problem

The analysis and decryption of encrypted text or information without prior knowledge of the keys.

— Oxford English Dictionary

cryptology

κρυπός   λογια

hidden   speaking
(knowledge)
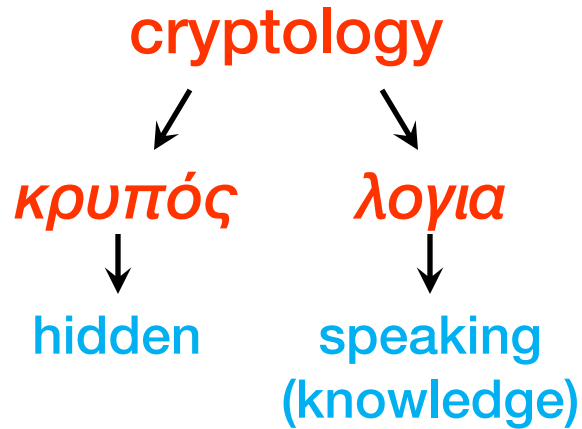
1967 D. Kahn, *Codebreakers* p. xvi, Cryptology is the science that embraces cryptography and cryptanalysis, but the term 'cryptology' sometimes loosely designates the entire dual field of both rendering signals secure and extracting information from them.

— *Oxford English Dictionary*

Cryptography ≠ Security

Cryptography may be a component of a secure system

Just adding cryptography may not make a system secure

Cryptography: what is it good for?

- **Confidentiality**
  - Others cannot read contents of the message

- **Authentication**
  - Determine origin of message

- **Integrity**
  - Verify that message has not been modified

- **Nonrepudiation**
  - Sender should not be able to falsely deny that a message was sent

Terms

Plaintext (cleartext) message P

Encryption $E(P)$

Produces Ciphertext, $C = E(P)$

Decryption, $P = D(C)$

Cipher = cryptographic algorithm

Restricted algorithm

Algorithm is kept secret

- **Vulnerable to:**
  - Leaking
  - Reverse engineering
    - HD DVD (Dec 2006) and Blu-Ray (Jan 2007)
    - Digital cellular encryption algorithms
    - DVD and DIVX video compression
    - Firewire protocol
    - Enigma cipher machine
    - Every NATO and Warsaw Pact algorithm during Cold War

- **Hard to validate its effectiveness (who will test it?)**

- **Useless once exposed – not a viable approach!**

Shared algorithms & secret keys
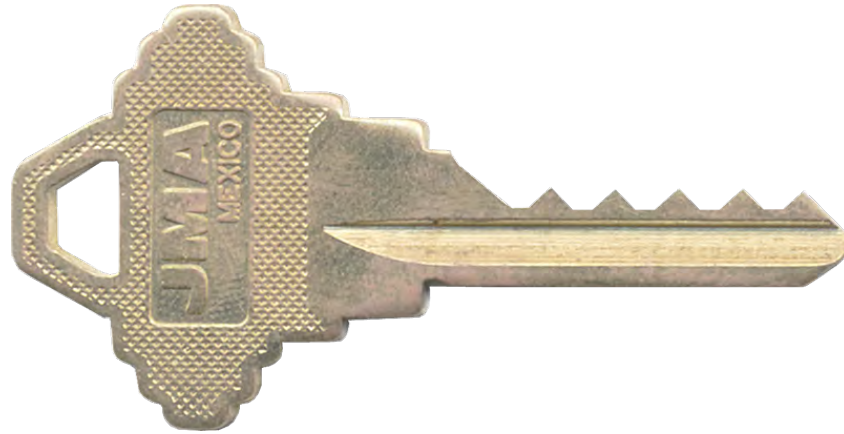
BTW, the above is a *bump key*. See http://en.wikipedia.org/wiki/Lock_bumping

Source: en.wikipedia.org/wiki/Pin_tumbler_lock
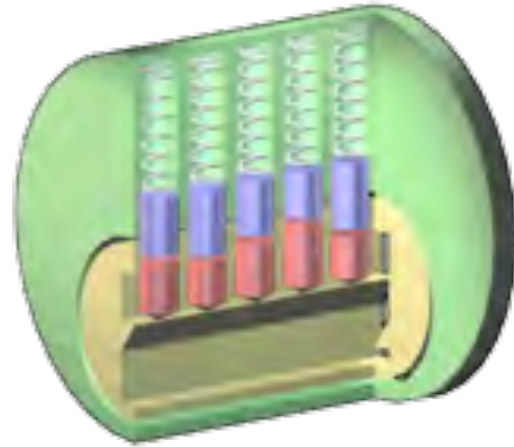
Source: en.wikipedia.org/wiki/Pin_tumbler_lock

The key & lock

- We understand how the mechanism works:
  - Strengths
  - Weaknesses

- Based on this understanding, we can assess how much to trust the key & lock
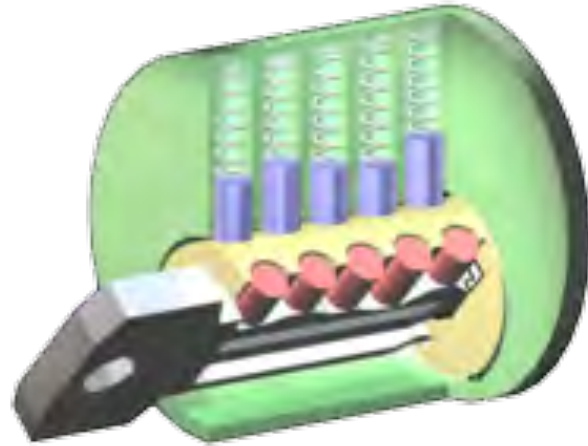
Source: en.wikipedia.org/wiki/Pin_tumbler_lock

Kerckhoffs's Principle (1883)

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

Security should rest entirely on the secrecy of the key

Properties of a good cryptosystem

1. Ciphertext should be indistinguishable from random values

2. Given ciphertext, there should be no way to extract the original plaintext or the key short of enumerating all possible keys (i.e., a *brute force attack*)

3. The keys should be large enough that a brute force attack is not feasible

Symmetric key ciphers

One shared secret key, *K*, for encryption & decryption

$$C = E_K(P)$$

$$P = D_K(C)$$

Classic Cryptosystems

Substitution Ciphers

Atbash (אתבש) – Ancient Hebrew cipher

MY CAT HAS FLEAS

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

NBXZGSZHUOVZH

Origin of the name:
א – alef (1st letter),
ת – tav (last letter),
ב – bet (2nd),
ש – shin (2nd to last)

c. 600 BCE

No information (key) needs to be conveyed!

ש	ת	ר	ק	צ	פ	ע	ס	נ	מ	ל	כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א
ש	ת	ר	ק	צ	פ	ע	ס	נ	מ	ל	כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א

"The art of understanding writing in cypher, and the writing of words in a peculiar way"

Kautiliya

– Phonetic substitution scheme used in India 400 BCE – 200 CE

– Short & long vowels are exchanged with consonants

a	ā	i	ī	u	ū	ṛ	ṝ	ḷ	Ḹ	e	ai	o	au	ṃ	ḥ	ñ	ś	ṣ	s	i	r	l	u
kh	g	gh	ṅ	ch	j	jh	ñ	ṭh	ḍ	ḍh	ṇ	th	d	dh	n	ph	b	bh	m	y	r	l	v

Cæsar cipher

Earliest documented military use of cryptography

- Julius Caesar c. 60 BCE

- Shift cipher: simple variant of a substitution cipher

- Each letter replaced by one n positions away modulo alphabet size
  - n = shift value = key

Cæsar cipher

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	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

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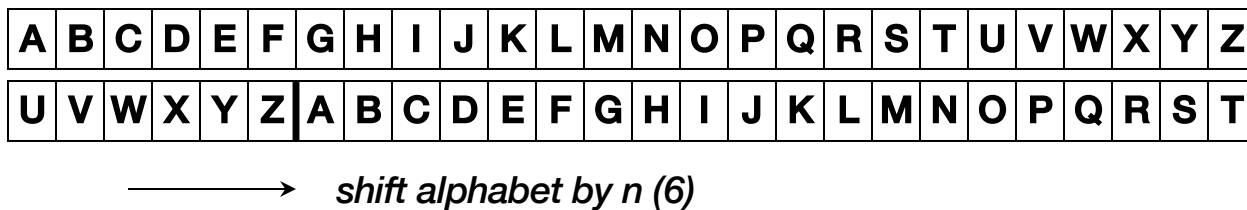
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

⟶ *shift alphabet by n (6)*

Cæsar cipher

MY CAT HAS FLEAS

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
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

Cæsar cipher

MY CAT HAS FLEAS

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

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

G

Cæsar cipher

MY CAT HAS FLEAS

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

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

GS

Cæsar cipher

MY CAT HAS FLEAS

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

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

GSW

Cæsar cipher

MY CAT HAS FLEAS

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

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

GSWU

Cæsar cipher

MY CAT HAS FLEAS

| 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 |

| 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 |

GSWUNBMUFZYUM

- Convey one piece of information for decryption: *shift value*

- Trivially easy to crack
  (25 possibilities for a 26-character alphabet)

Monoalphabetic substitution cipher

MY CAT HAS FLEAS

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

IVSMXAMBQCLMB

Monoalphabetic = constant mapping between plaintext and ciphertext

General case: arbitrary mapping
(instead of a Cæsar cipher, where the letters are in an alphabetic sequence but shifted)

Both sides must have the same substitution alphabet

Monoalphabetic substitution cipher

Easy to decode: vulnerable to frequency analysis

Moby Dick
(1.2M chars)

Shakespeare
(55.8M chars)

e	12.300%	e	11.797%
o	7.282%	o	8.299%
d	4.015%	d	3.943%
b	1.773%	b	1.634%
x	0.108%	x	0.140%

*Frequency
distribution of letters*

Frequency Analysis

Letter frequencies

E: 12%

A, H, I, N, O, R, S, T: 6 – 9%

D, L: 4%

B, C, F, G, M, P, U, W, Y: 1.5 – 2.8%

J, K, Q, V, X, Z: < 1%

Common digrams:

TH (3.56%), HE (3.07%), IN (2.43%), ER (2.05%), AN, RE, …

Common trigrams

THE, ING, AND, HER, ERE, …

Polyalphabetic substitution ciphers

- **Designed to thwart frequency analysis techniques**
  - Different ciphertext symbols can represent the same plaintext symbol

    1 → *many* relationship between letter and substitution

- **Leon Battista Alberti: 1466**
  - Two disks
  - Line up predetermined letter on inner disk with outer disk
  - Plaintext on inner → ciphertext on outer
  - After *n* symbols, the disk is rotated to a new alignment
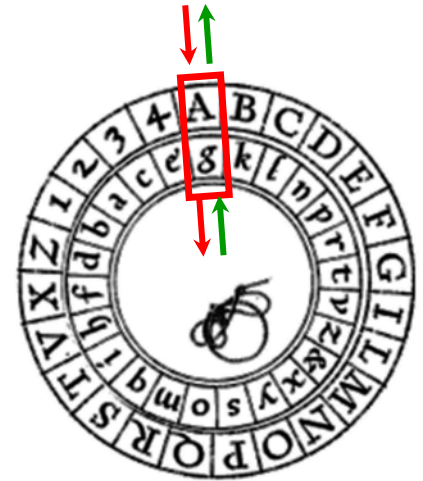
encrypt: A→g
decrypt: g→A

Image source: https://en.wikipedia.org/wiki/Alberti_cipher_disk

http://dabblesandbabbles.com/printable-secret-decoder-wheel/

Vigenère polyalphabetic cipher

Blaise de Vigenère, court of Henry III of France, 1518

No need for a disk: use table and key word to encipher a message

	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

Vigenère polyalphabetic cipher

- **Repeat keyword over text: (e.g., key=FACE)**

  Keystream: FA CEF ACE FACEF ....

  Plaintext: MY CAT HAS FLEAS

- **Encrypt**: find intersection:

  row = keystream letter
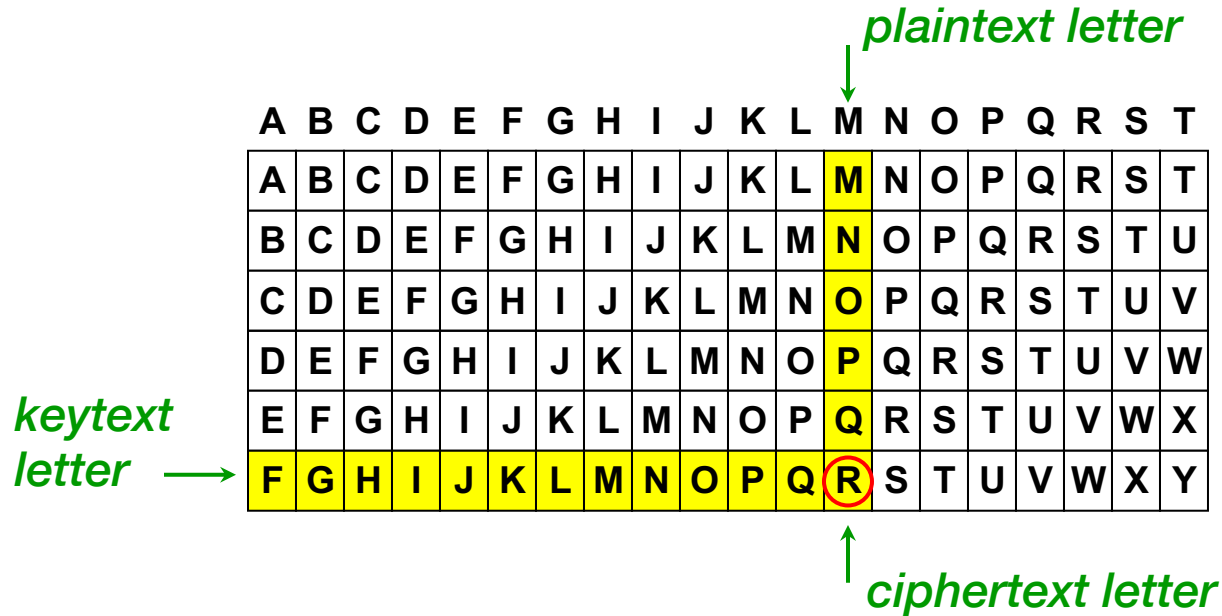 **column** = plaintext (message) letter

- **Decrypt**: find column

  – Row = keystream letter, search for ciphertext

  – Column heading = plaintext letter

- **Message is encrypted with as many substitution ciphers as there are unique letters in the keyword**

Vigenère polyalphabetic cipher

Vigenère polyalphabetic cipher

```
FA CEF ACE FACEF
MY CAT HAS FLEAS
```

R

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	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	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	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	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	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	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	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

Vigenère polyalphabetic cipher

```
FA CEF ACE FACEF
MY CAT HAS FLEAS
```

RY

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	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	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	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	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	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	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	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

Vigenère polyalphabetic cipher

```
FA CEF ACE FACEF
MY CAT HAS FLEAS
```
RY E

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	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	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	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	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	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	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	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

Vigenère polyalphabetic cipher

```
FA CEF ACE FACEF
MY CAT HAS FLEAS
```
RY EE

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	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	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	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	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	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	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	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

Vigenère polyalphabetic cipher

```
FA CEF ACE FACEF
MY CAT HAS FLEAS
```
```
RY EEY
```

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	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	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	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	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	**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	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	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

Vigenère polyalphabetic cipher

```
FA CEF ACE FACEF
MY CAT HAS FLEAS
```
───────────────────
```
RY EEY HCW KLGEX
```

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	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	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	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	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	**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	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	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

Vigenère polyalphabetic cipher

"The rebels reposed their major trust, however, in the Vigenère, sometimes using it in the form of a brass cipher disc. In theory, it was an excellent choice, for so far as the South knew the cipher was unbreakable. In practice, it proved a dismal failure. For one thing, transmission errors that added or subtracted a letter ... unmeshed the key from the cipher and caused no end of difficulty. Once Major Cunningham of General Kirby-Smith's staff tried for twelve hours to decipher a garbled message; he finally gave up in disgust and galloped around the Union flank to the sender to find out what it said."

http://rz1.razorpoint.com/index.html

Hard to break with long keys and small amounts of ciphertext *...*
… in the 1800s

Cryptanalysis of the Vigenère cipher

1. Determine key length

   - Count coincidences – identical sets of characters *n* characters apart
   - Key length is likely to be the separation with the maximum # of coincidences

2. Determine values of each character of the key

   - You know the length of the key – that's the # of Caesar ciphers you have
   - Do a frequency analysis of each position of the key

Digression: exclusive-or

AND (∧): clears bits

Truth table

$$1 \wedge 1 = 1$$
$$1 \wedge 0 = 0$$
$$0 \wedge 1 = 0$$
$$0 \wedge 0 = 0$$

AND clears bits

– AND 1 – keep the bit
– AND 0 – clear the bit

If you clear a bit, you will never know if it used to be a 0 or a 1

Boolean logic refresher: OR

OR (∨): sets bits

OR sets bits
- OR 1 – set the bit
- OR 0 – keep the bit

$$1 \vee 1 = 1$$
$$1 \vee 0 = 1$$
$$0 \vee 1 = 1$$
$$0 \vee 0 = 0$$

If you set a bit, you will never know if it used to be a 0 or a 1

XOR (⊕): flips bits

Truth table

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

XOR flips bits

- XOR 1 – flip the bit
- XOR 0 – keep the bit as it is

If you flip a bit, you can restore it by XORing it with 1 again

XOR in cryptography

We use XOR operations a lot in cryptography

They allow us to flip certain bits to encrypt and later unflip to decrypt

```
    0 1 1 1 0 0 0 0      0 1 1 0 1 0 1 1        plaintext = 0x70  0x6b
⊕   1 0 1 0 1 1 0 0      1 0 1 1 1 0 0 1         ⊕ key = 0xac  0xb9
   ─────────────────    ─────────────────
    1 1 0 1 1 1 0 0      1 1 0 1 0 0 1 0        ciphertext = 0xdc  0xd2
⊕   1 0 1 0 1 1 0 0      1 0 1 1 1 0 0 1         ⊕ key = 0xac  0xb9
   ─────────────────    ─────────────────
    0 1 1 1 0 0 0 0      0 1 1 0 1 0 1 1           result = 0x70  0x6b
```

plaintext recovered!

End of digression

One-time pad

Only provably secure encryption scheme

- Invented in 1917

- Large non-repeating set of *random* key letters originally written on a pad

```
CIHJT  UUHML  FRUGC  ZIBGD  BQPNI  PDNJG  LPLLP  YJYXM
DCXAC  JSJUK  BIOYT  MWQPX  DLIRC  BEXYK  VKIMB  TYIPE
UOLYQ  OKOXH  PIJKY  DRDBC  GEFZG  UACKD  RARCD  HBYRI
DZJYO  YKAIE  LIUYW  DFOHU  IOHZV  SRNDD  KPSSO  JMPQT
MHQHL  OHQQD  SMHNP  HHOHQ  GXRPJ  XBXIP  LLZAA  VCMOG
AWSSZ  YMFNI  ATMON  IXPBY  FOZLE  CVYSJ  XZGPU  CTFQY
HOVHU  OCJGU  QMWQV  OIGOR  BFHIZ  TYFDB  VBRMN  XNLZC
```

- Each key letter on the pad encrypts exactly one plaintext character
  - Encryption is addition of characters modulo *alphabet size* (26)

- Sender destroys pages that have been used

- Receiver maintains identical pad

https://en.wikipedia.org/wiki/File:OneTimePadExcerpt.agr.png

Some one-time pads

Letters-only one-time pad
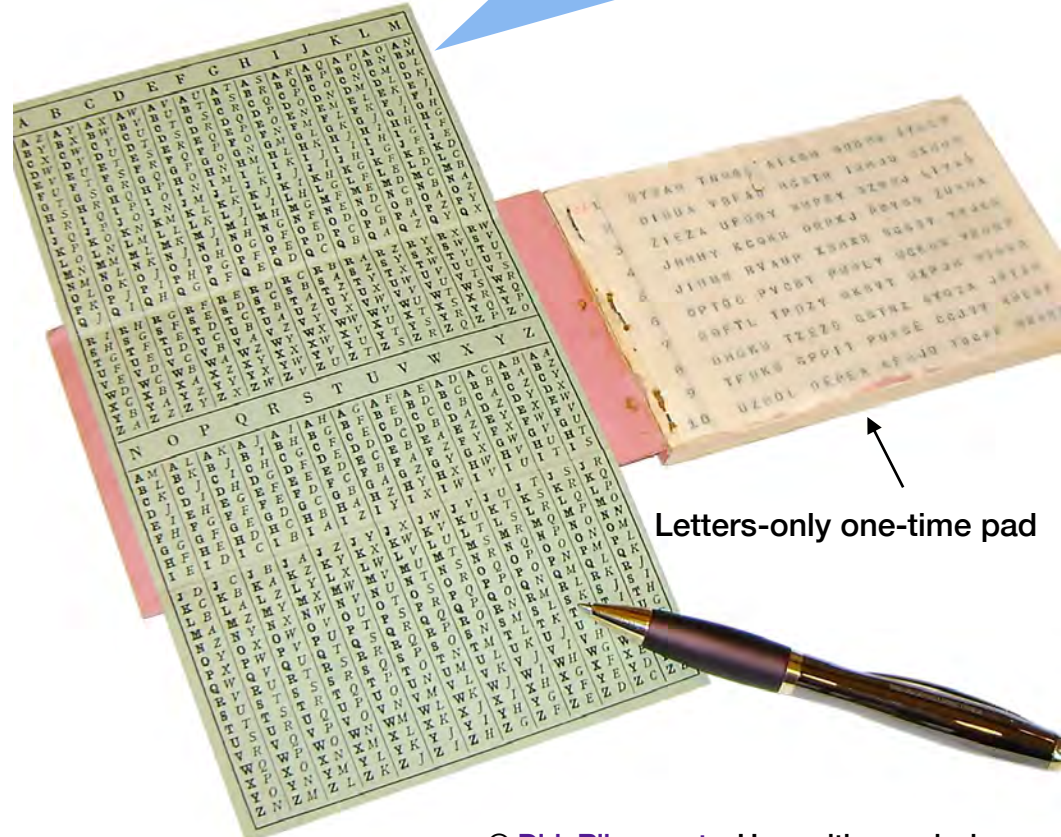
A Russian One-time pad, captured by MI5
Photo from ramnum.com. Used with permission

© Dirk Rijmenants. User with permission

If pad contains

KWXOPWMAELGHW...

and we want to encrypt

MY CAT HAS FLEAS

Ciphertext =

WUZOIDMSJWKHO

```
M + K mod 26 = W
Y + W mod 26 = U
C + X mod 26 = Z
A + O mod 26 = O
T + P mod 26 = I
H + W mod 26 = D
A + M mod 26 = M
S + A mod 26 = S
F + E mod 26 = J
L + L mod 26 = W
E + G mod 26 = K
A + H mod 26 = H
S + W mod 26 = O
```

The same ciphertext can decrypt to *anything* depending on the key!

Same ciphertext:

   WUZOIDMSJWKHO

With a pad containing:
   DNVLUXEACWVSQ**...**

Produces:

  THE DOG IS HAPPY

```
W – D mod 26 = T
U – N mod 26 = H
Z – V mod 26 = E
O – L mod 26 = D
I – U mod 26 = O
D – X mod 26 = G
M – E mod 26 = I
S – A mod 26 = S
J – C mod 26 = H
W – W mod 26 = A
K – V mod 26 = P
H – S mod 26 = P
O – Q mod 26 = Y
```

Can be extended to binary data

– Random key sequence as long as the message

– Exclusive-or key sequence with message

– Receiver has the same key sequence

```c
void onetimepad(void)
{
    FILE *if = fopen("intext", "r");
    FILE *kf = fopen("keytext", "r");
    FILE *of = fopen("outtext", "w");
    int c, k;

    while ((c = getc(if)) != EOF) {
        k = getc(kf);
        putc((c^k), of);
    }
    fclose(if); fclose(kf); fclose(of);
}
```

One-time pads: **perfect secrecy**

Perfect secrecy

– Ciphertext conveys no information about the content of plaintext
– *Achieved only if there are as many possible keys as plaintext*

Problems with one-time pads:

– Key needs to be as long as the message!

– Key storage and distribution can be problematic

– Keys have to be generated **randomly**
  • Cannot use pseudo-random number generator

– Cannot reuse key sequence

– Sender and receiver *must* remain synchronized (e.g., cannot lose any part of the message)

Random numbers

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"

– John von Neumann, 1951

- **Pseudo-random generators**
  - Linear feedback shift registers
  - Multiplicative lagged Fibonacci generators
  - Linear congruential generator

- **Obtain randomness from:**
  - Time between keystrokes
  - Various network/kernel events
  - Cosmic rays
  - Electrical noise
  - Other encrypted messages
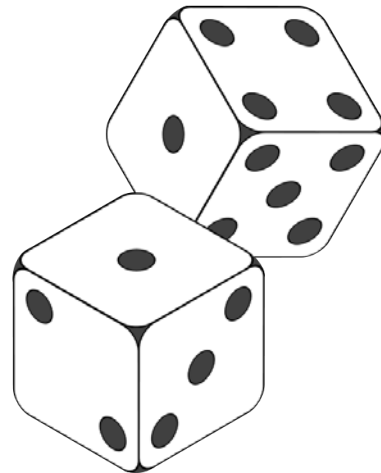
Image from Wikimedia Commons

Researchers use tiny magnetic swirls to generate true random numbers

PHYS.ORG

by Brown University
February 7, 2022

Whether for use in cybersecurity, gaming or scientific simulation, the world needs true random numbers, but generating them is harder than one might think. But a group of Brown University physicists has developed a technique that can potentially generate millions of random digits per second by harnessing the behavior of skyrmions—tiny magnetic anomalies that arise in certain two-dimensional materials.

Their research, published in Nature Communications, reveals previously unexplored dynamics of single skyrmions, the researchers say. Discovered around a half-decade ago, skyrmions have sparked interest in physics as a path toward next-generation computing devices that take advantage of the magnetic properties of particles—a field known as spintronics.
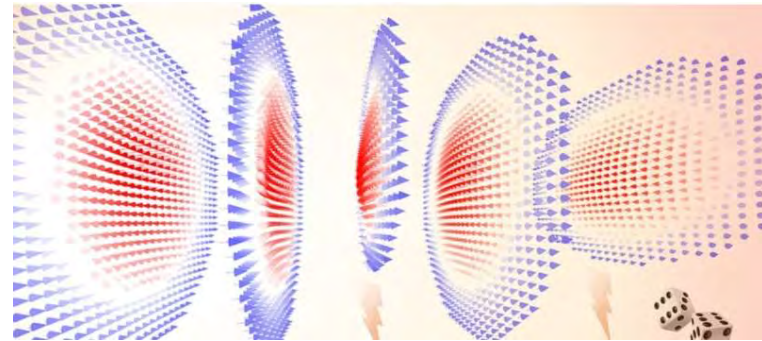
Magnetic swirls called skyrmions fluctuate randomly in size, a behavior that can be harnessed to generate true random numbers. Credit: Xiao lab / Brown University

https://phys.org/news/2022-02-tiny-magnetic-swirls-true-random.html

Stream ciphers

Key stream generator produces a sequence of **pseudo-random** bytes

Simulates a one-time pad
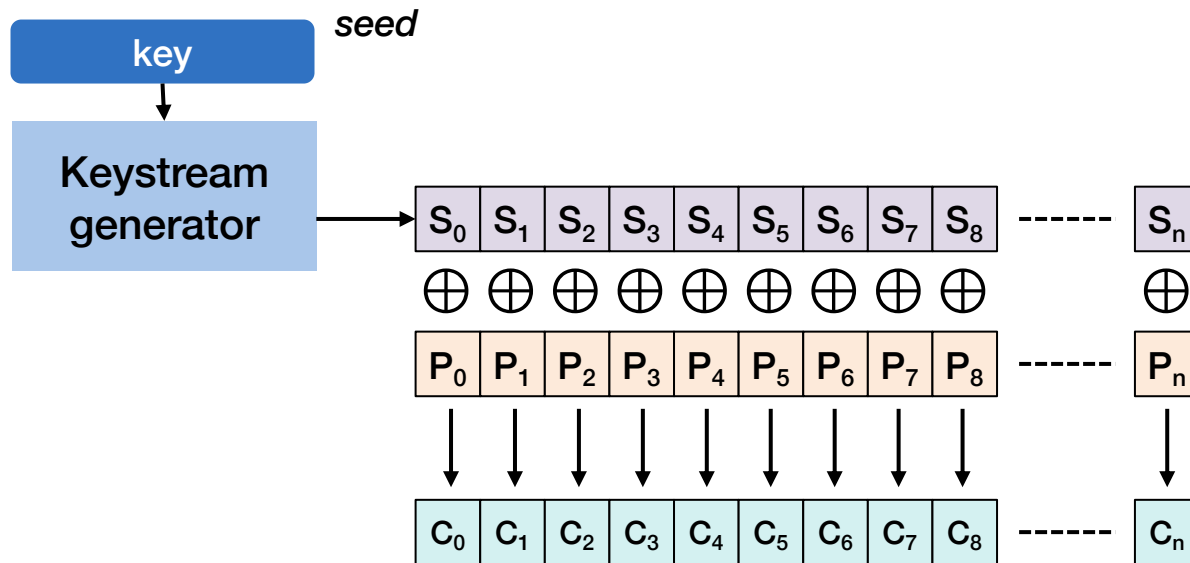
$$C_i = S_i \oplus P_i$$

Can never reuse a key

$$C = A \oplus K$$

$$C' = B \oplus K$$

$$C \oplus C' = A \oplus K \oplus B \oplus K = A \oplus B$$

Guess *A* to get *K* and see if *B* makes sense

Or… if you have **known plaintext** A and the corresponding ciphertext C, you can extract the key:

$$K = A \oplus C$$

Electro-mechanical cryptographic engines

Rotor machines

1920s: mechanical devices used for automating encryption

Rotor machine:

– Set of independently rotating cylinders (rotors) through which electrical pulses flow

– Each rotor has input & output pin for each letter of the alphabet
  • Each rotor implements a substitution cipher

– Output of each rotor is fed into the next rotor

– Together they implement a version of the Vigenère cipher

CS 419 © 2022 Paul Krzyzanowski
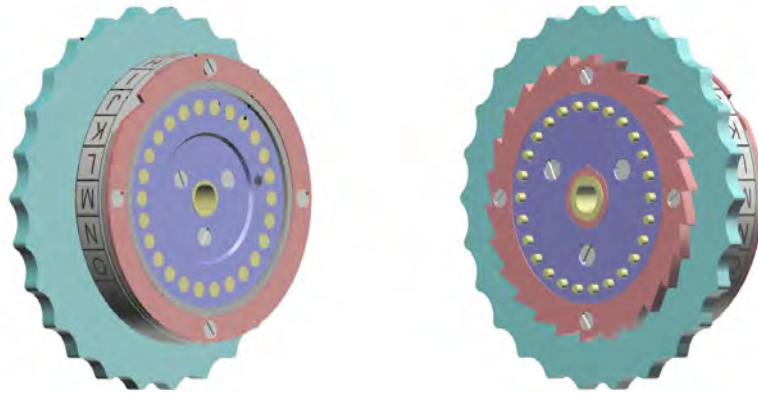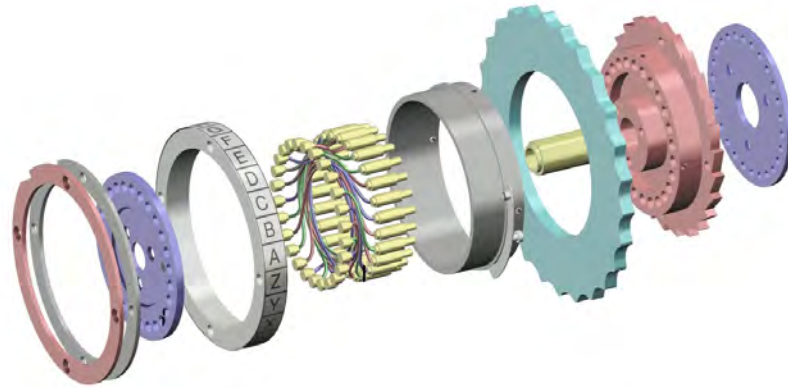
Simplest rotor machine: single cylinder
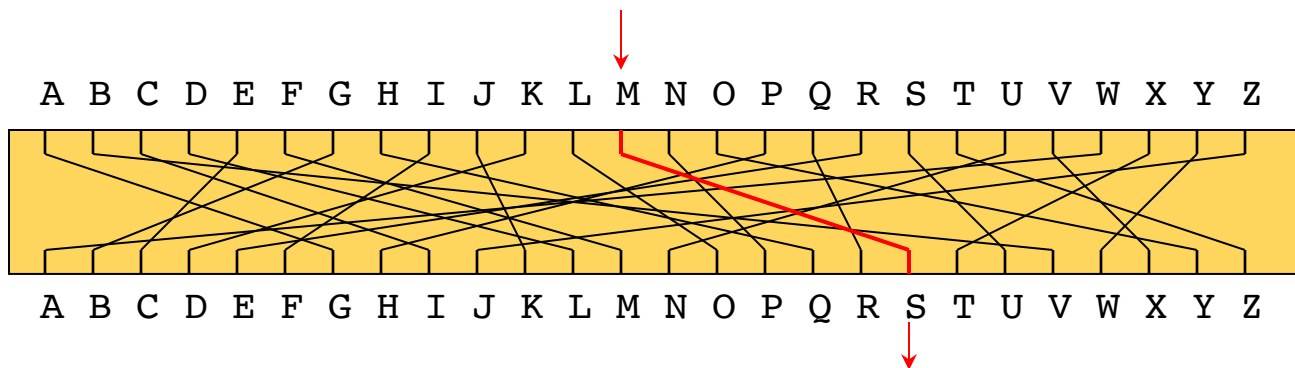
After a character is entered, the cylinder rotates one position

– Internal connections shifted by one

– Polyalphabetic substitution cipher with a period of 26

Single cylinder rotor machine
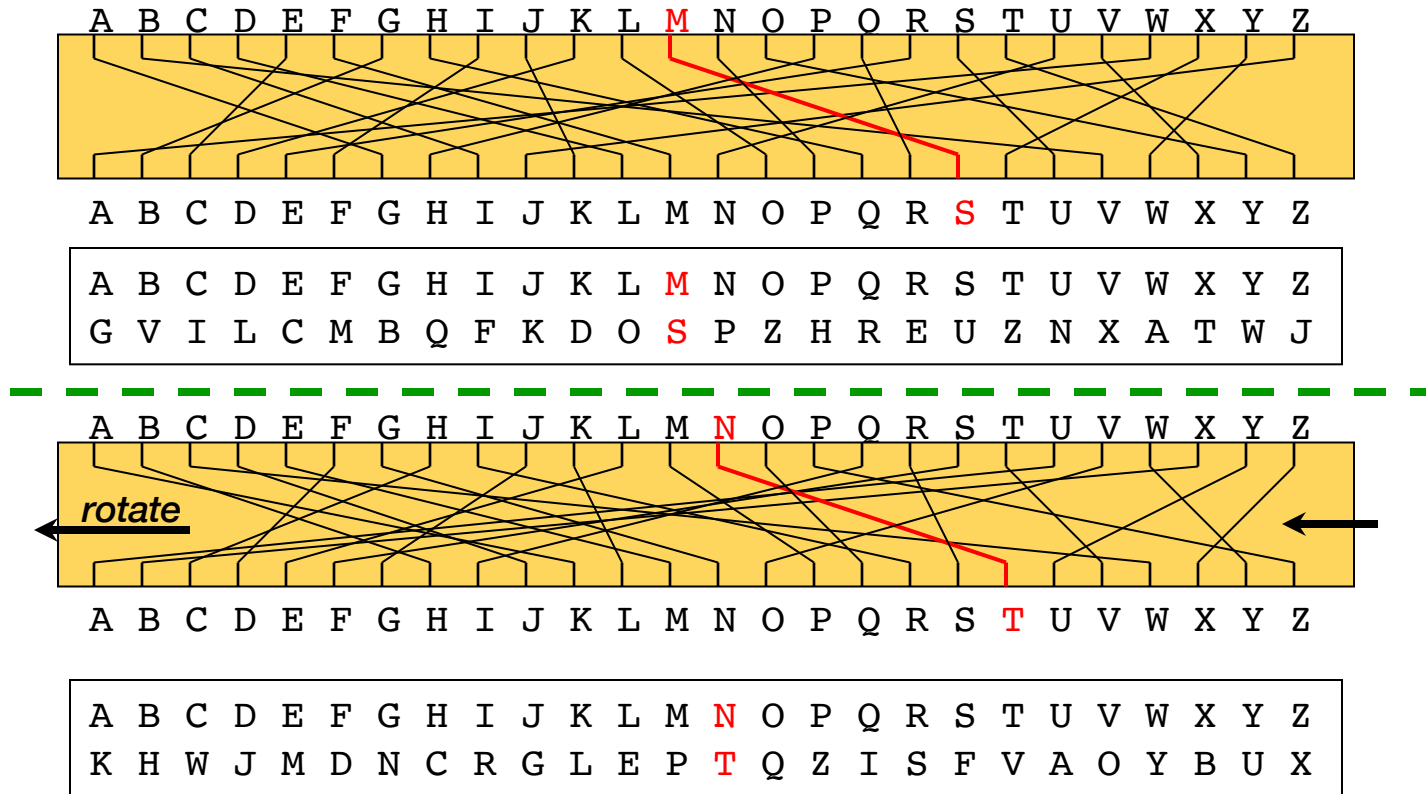
Multi-cylinder rotor machines

- **Single cylinder rotor machine**
  - Substitution cipher with a period = length of alphabet (e.g., 26)

- **Multi-cylinder rotor machine**
  - Feed output of one cylinder as input to the next one
  - First rotor advances after character is entered
  - Second rotor advances after a full period of the first
  - Polyalphabetic substitution cipher
    - Period = (length of alphabet)$^{\text{number of rotors}}$
    - 3 26-char cylinders $\Rightarrow 26^3 = 17,576$ substitution alphabets
    - 5 26-char cylinders $\Rightarrow 26^5 = 11,881,367$ substitution alphabets

Enigma

- **Enigma machine used in Germany during WWII**

- **Three rotor system**
  - $26^3 = 17,576$ possible rotor positions

- **Input data permuted via patch panel before sending to rotor engine**

- **Data from last rotor reflected back through rotors $\Rightarrow$ makes encryption symmetric**

- **Need to know initial settings of rotor**
  - setting was *f(date)* in a book of codes

- **Broken by group at Bletchley Park (Alan Turing)**

Enigma

Reflector

Rotors

Plugboard

Glowlamps
(results)

Keyboard
(input)
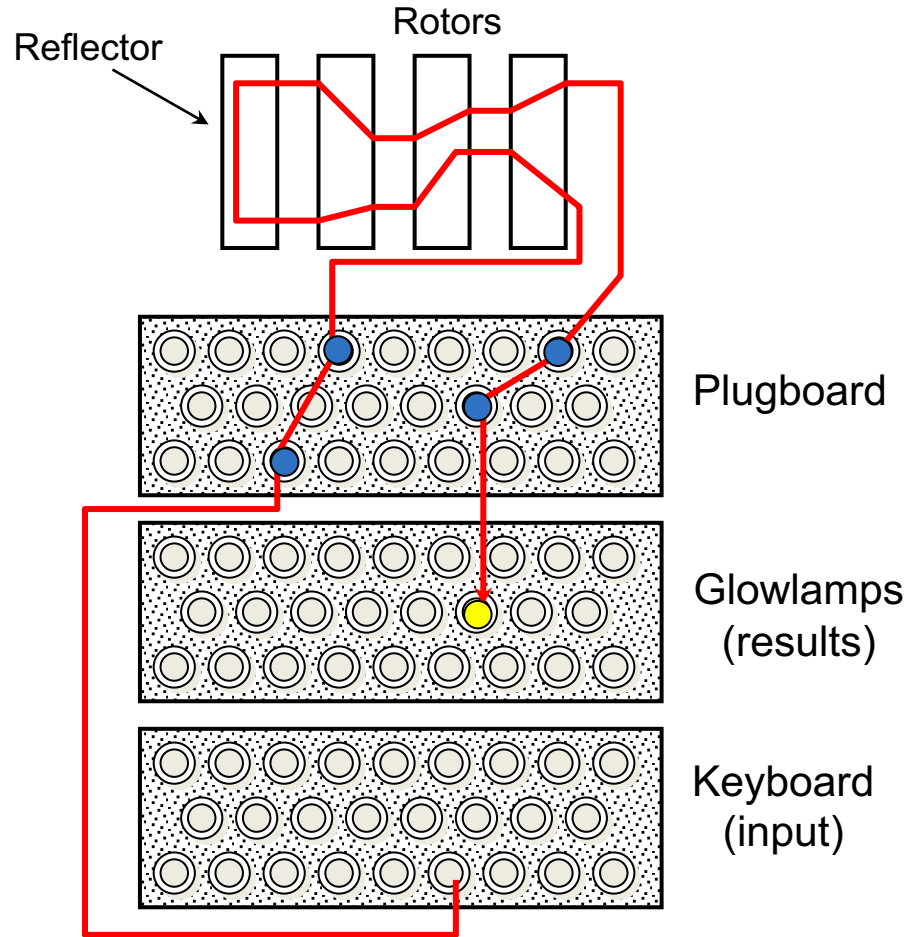
Transposition Ciphers

Transposition ciphers

- Permute letters in plaintext according to rules

- Knowledge of rules will allow message to be decrypted

- First mentioned in Greece in the 7th century BCE
  - **Scytale** (*rhymes with Italy*) = staff cipher

Secret = diameter of scytale

MYCATHASFLEAS

MHE

M
H
E

MYCATHASFLEAS

MHEYAACSS

MYCATHASFLEAS

CAT
SFL
Sxy

MHEYAACSSAFx

A
F
x

Pad out the text

Table version of scytale

– Enter data horizontally, read it vertically

– Secrecy is the width of the table

```
MYCATHASFLEAS    ➡    M Y C A
                      T H A S
                      F L E A
                      S x y z
```
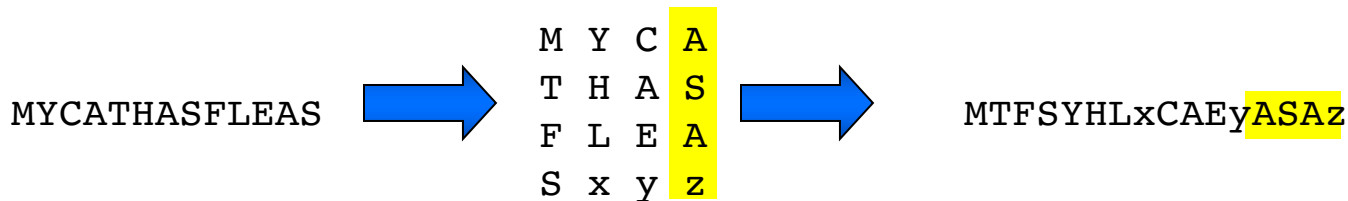
Table version of scytale

– Enter data horizontally, read it vertically
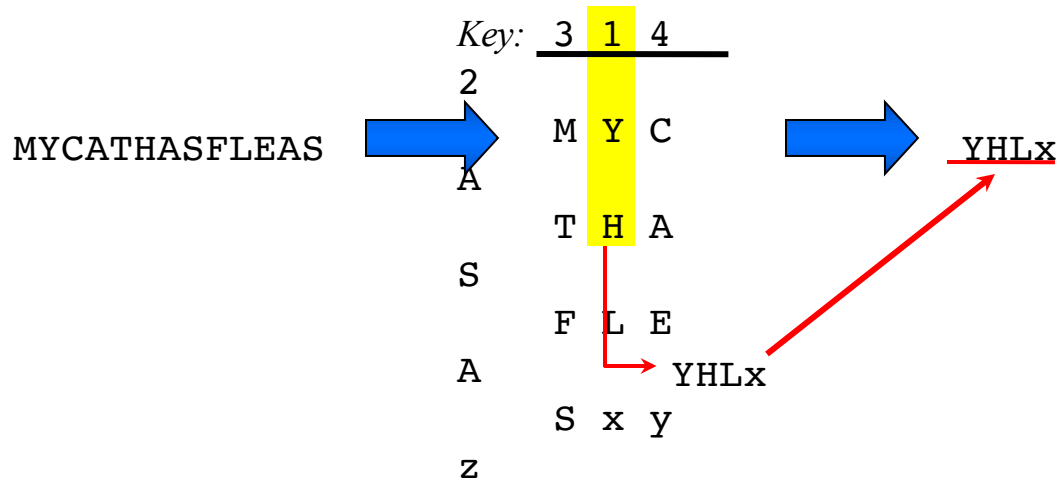
– Secrecy is the width of the table

```
          M Y C A
MYCATHASFLEAS    →    T H A S    →    MTFS
          F L E A
          S x y z
```

Table version of scytale

- Enter data horizontally, read it vertically
- Secrecy is the width of the table

Table version of scytale

– Enter data horizontally, read it vertically

– Secrecy is the width of the table

```
                          M Y C A
MYCATHASFLEAS    →        T H A S      →     MTFSYHLxCAEy
                          F L E A
                          S x y z
```

Table version of scytale

– Enter data horizontally, read it vertically

– Secrecy is the width of the table

```
MYCATHASFLEAS    →    M Y C A    →    MTFSYHLxCAEyASAz
                      T H A S
                      F L E A
                      S x y z
```
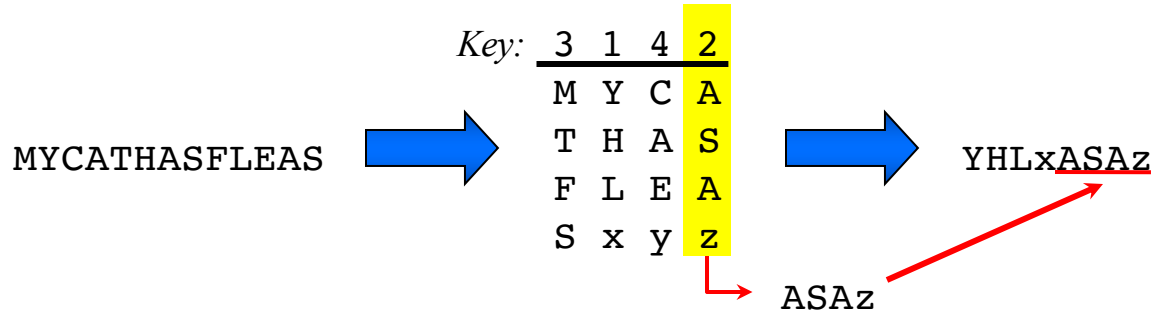
Columnar transposition cipher

- Permute letters in plaintext according to **key**
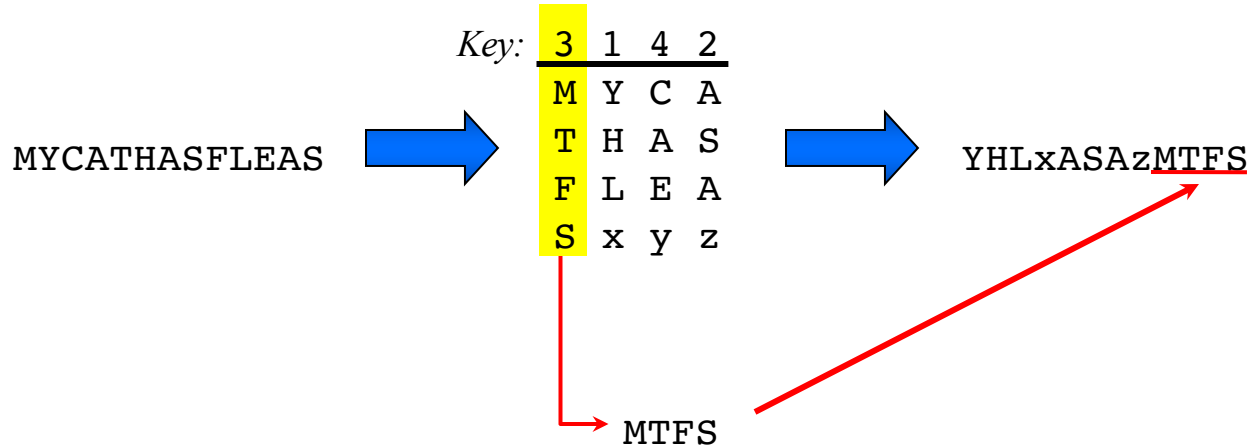- Read down columns, sorting by key

Key:

3	1	4	2
M	Y	C	A
T	H	A	S
F	L	E	A
S	x	y	z

MYCATHASFLEAS ➡️

Columnar transposition cipher

– Permute letters in plaintext according to **key**
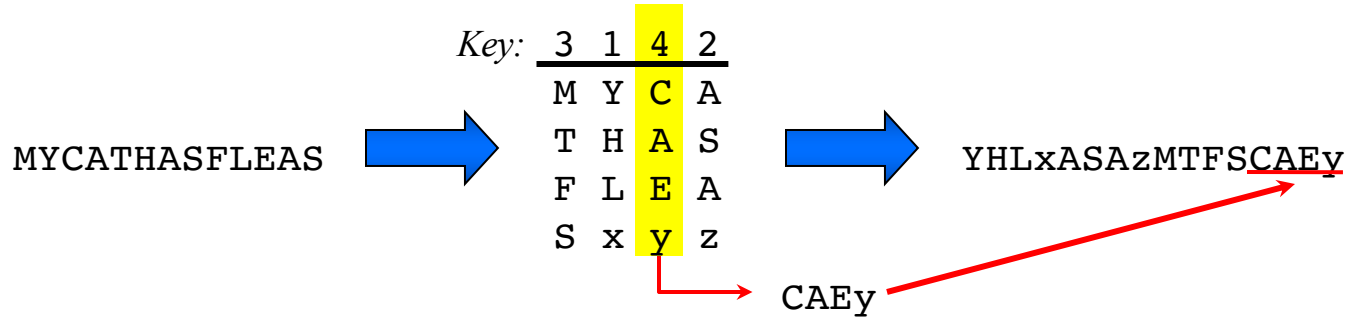– Read down columns, sorting by key

Key: 3 1 4

2

MYCATHASFLEAS → M Y C

A

T H A

S

F L E

A   YHLx

S x y

z

YHLx

Columnar transposition cipher

- Permute letters in plaintext according to **key**
- Read down columns, sorting by key

MYCATHASFLEAS

Key:

3	1	4	2
M	Y	C	A
T	H	A	S
F	L	E	A
S	x	y	z

ASAz

YHLxASAz

Columnar transposition cipher

– Permute letters in plaintext according to **key**
– Read down columns, sorting by key

Key: 3 1 4 2

MYCATHASFLEAS ➡️

3	1	4	2
M	Y	C	A
T	H	A	S
F	L	E	A
S	x	y	z

➡️ YHLxASAzMTFS

MTFS

Columnar transposition cipher

- Permute letters in plaintext according to **key**
- Read down columns, sorting by key

MYCATHASFLEAS ➡️

Key:
3	1	4	2
M	Y	C	A
T	H	A	S
F	L	E	A
S	x	y	z

CAEy

➡️ YHLxASAzMTFSCAEy

Columnar transposition cipher

- Permute letters in plaintext according to **key**
- Read down columns, sorting by key

	Key:	3	1	4	2
		M	Y	C	A
MYCATHASFLEAS	→	T	H	A	S	→ YHLxASAzMTFSCAEy
		F	L	E	A
		S	x	y	z
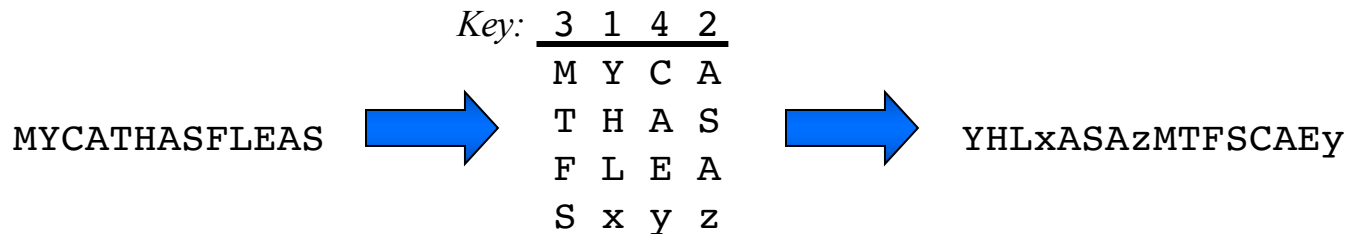
Transposition cipher

- **Not vulnerable to frequency analysis**

- **Scytale trivial to attack**
  - Make all possible matrices that would fit the ciphertext
  - Write ciphertext across rows
  - See if the columns contain legible content

- **Scrambled columns make it a bit harder**
  - Need to permute columns of matrices

Combined ciphers

- **Combine transposition with substitution ciphers**
  - German ADFGVX cipher (WWI)

- **Can be troublesome to implement**
  - Requires memory
  - Requires block processing (these are block ciphers)

- **Difficult with pencil-and-paper cryptography**

Computer Cryptography

Block ciphers

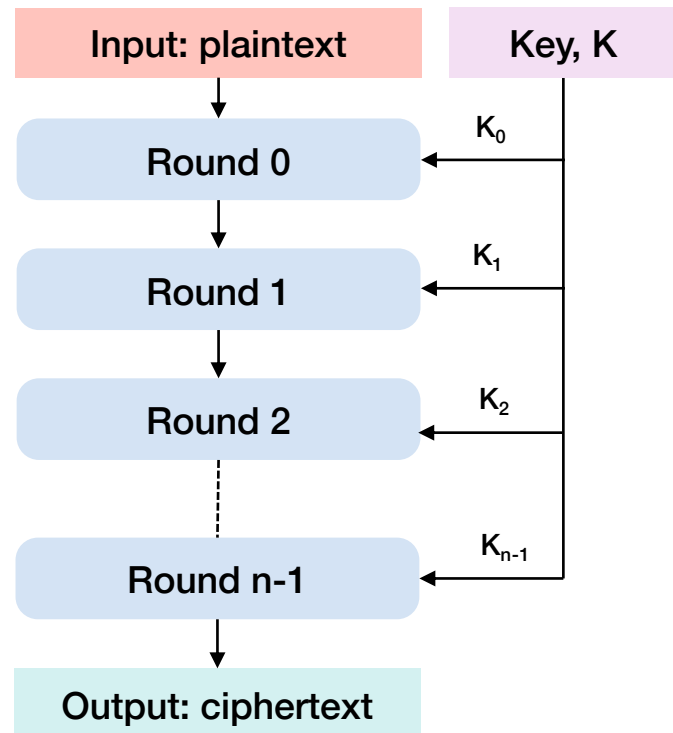## Block ciphers dominate computer cryptography

Encrypt a fixed number of bits at a time

Output blocksize (usually) == input blocksize

```
Plaintext (n bits)
        |
        v
Key (m bits) ------> Block cipher
                          |
                          v
                   Ciphertext (n bits)
```
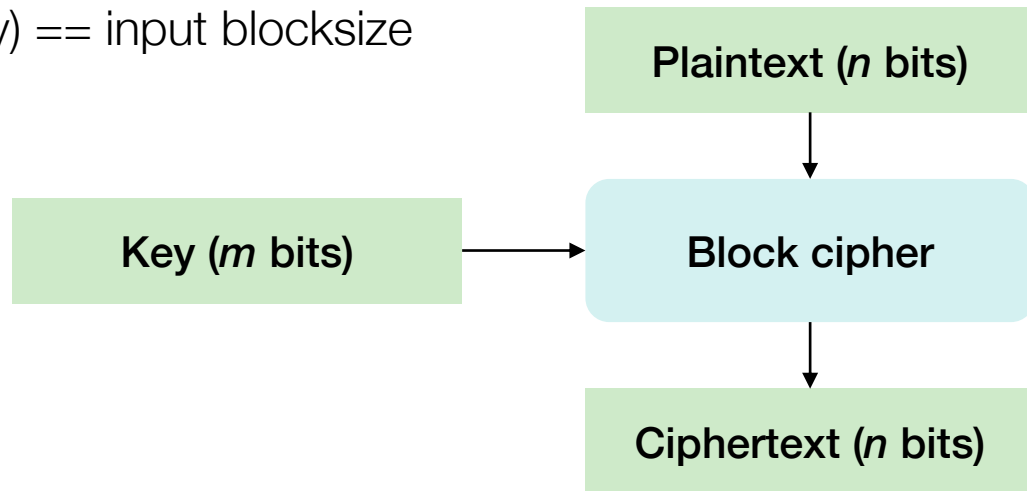
Block ciphers

- **Block ciphers encrypt a _block_ of plaintext at a time**

- **DES & AES are two popular block ciphers**
  - DES: 64-bit blocks
  - AES: 128-bit blocks

- **Block ciphers are usually iterative ciphers**
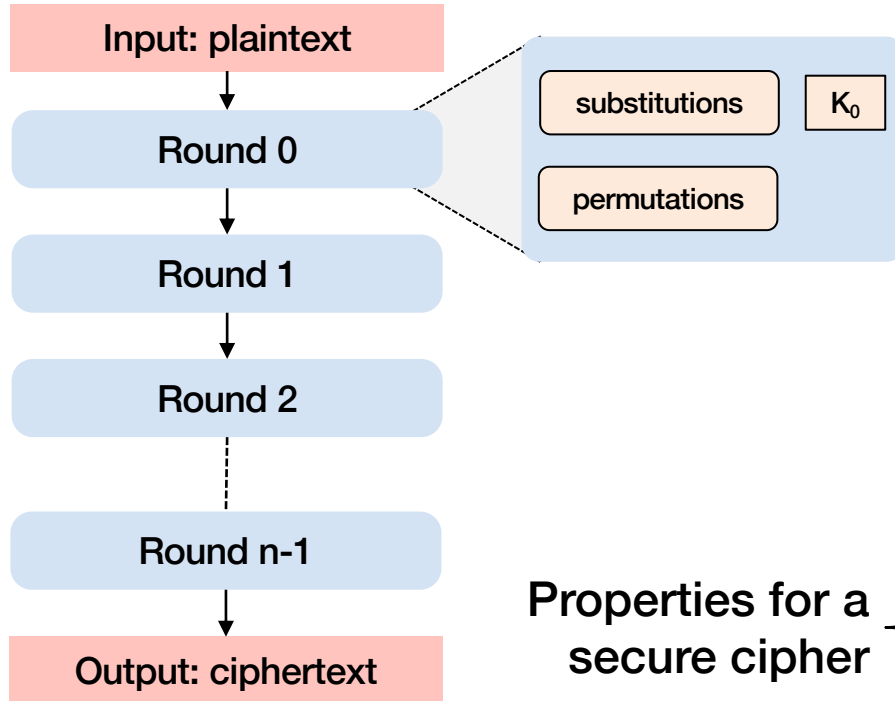 - The encryption process is an iteration through several _round_ operations

Input: plaintext	Key, K

Round 0 — K_0

Round 1 — K_1

Round 2 — K_2

Round n-1 — K_{n-1}

Output: ciphertext

Structure of block ciphers

- **Multiple rounds of combining the plaintext with the key**

- **Optional:**
  - Convert key to an internal subkey – different for each round

- **DES: 16 rounds**

- **AES: 10-14 rounds, depending on key length**

Sounds easy … but is difficult to design

Block cipher rounds

Each round consists of substitutions & permutations = SP Network

Input: plaintext

↓

Round 0 → substitutions | K_0
permutations

↓

Round 1

↓

Round 2

⋮

Round n-1

↓

Output: ciphertext

Properties for a secure cipher

<u>Substitution</u> = **S-box**
- Table lookup
- Converts a small block of input to a block of output

<u>Permutation</u>
- Scrambles the bits in a prescribed order

<u>Key application per round</u>
- Subkey, K_n, per round derived from the key
- Can drive behavior of s-boxes
- May be XORed with the output of each round

<u>Create *Confusion* & *Diffusion*</u>
- **Confusion**: no direct correlation between a bit of the key and resulting ciphertext
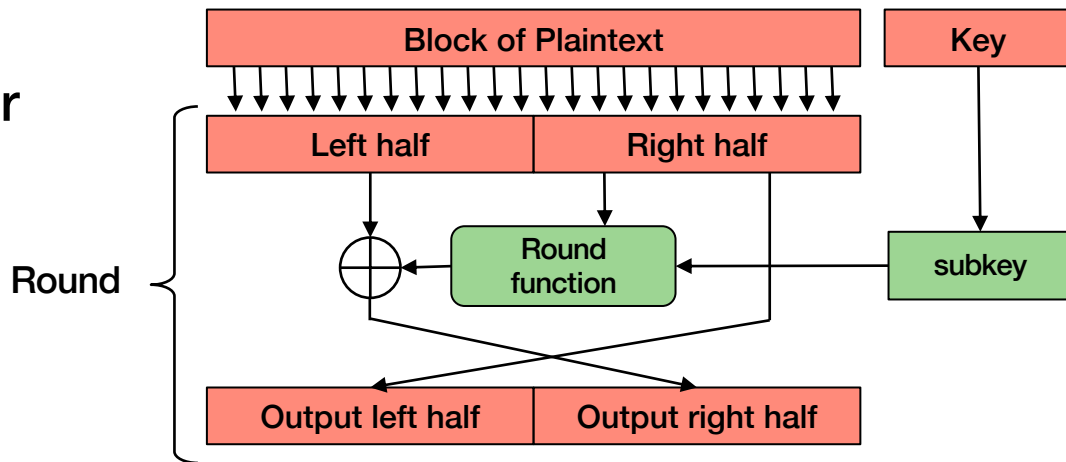- **Diffusion**: Changing one bit of input should change, on average, ½ of output bits
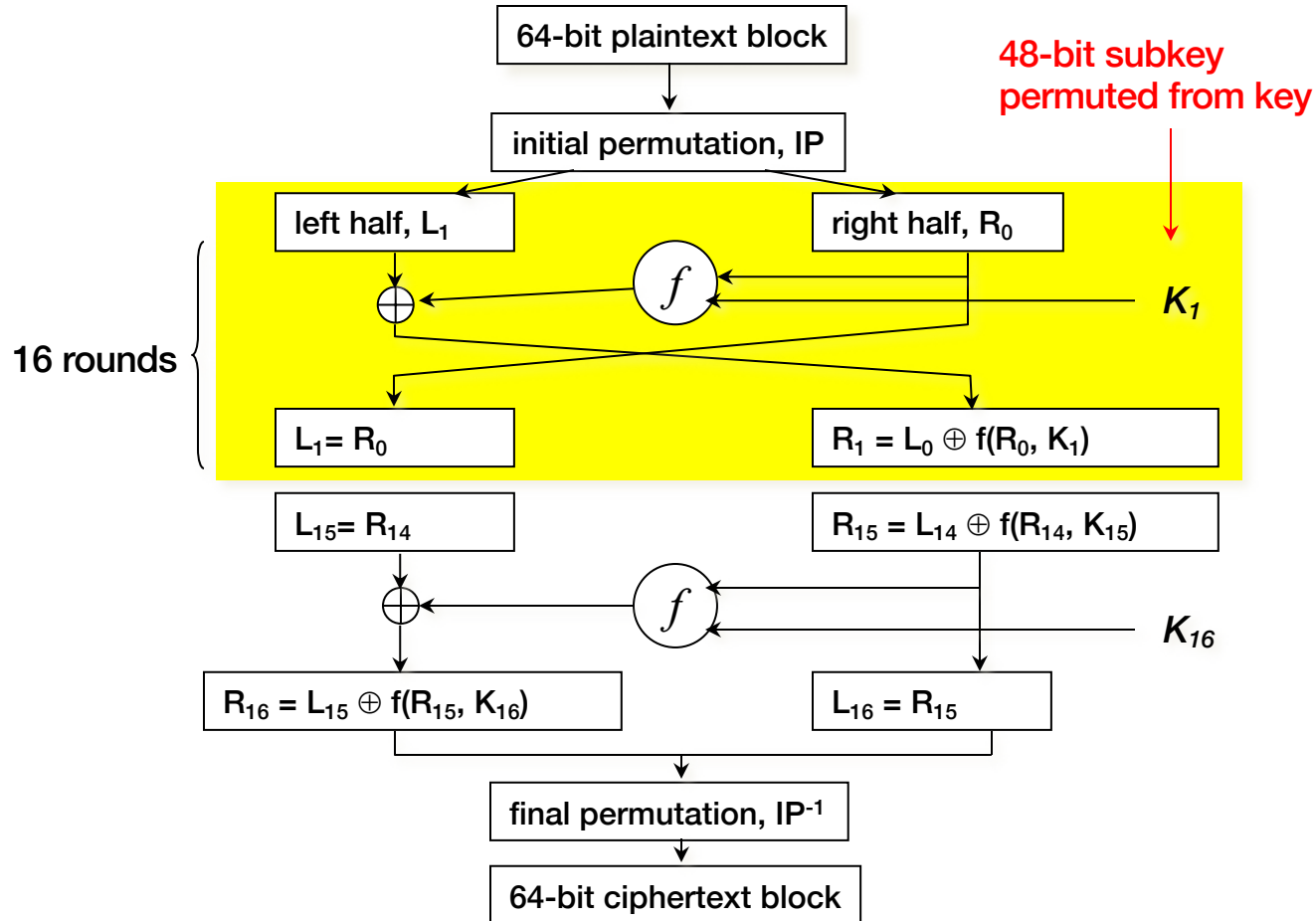
DES

Data Encryption Standard

– Adopted as a federal standard in 1976

- **Block cipher, 64-bit blocks, 56-bit key**

- **Substitution followed by a permutation**
  - Transposition and XORs based on a subkey derived from the key
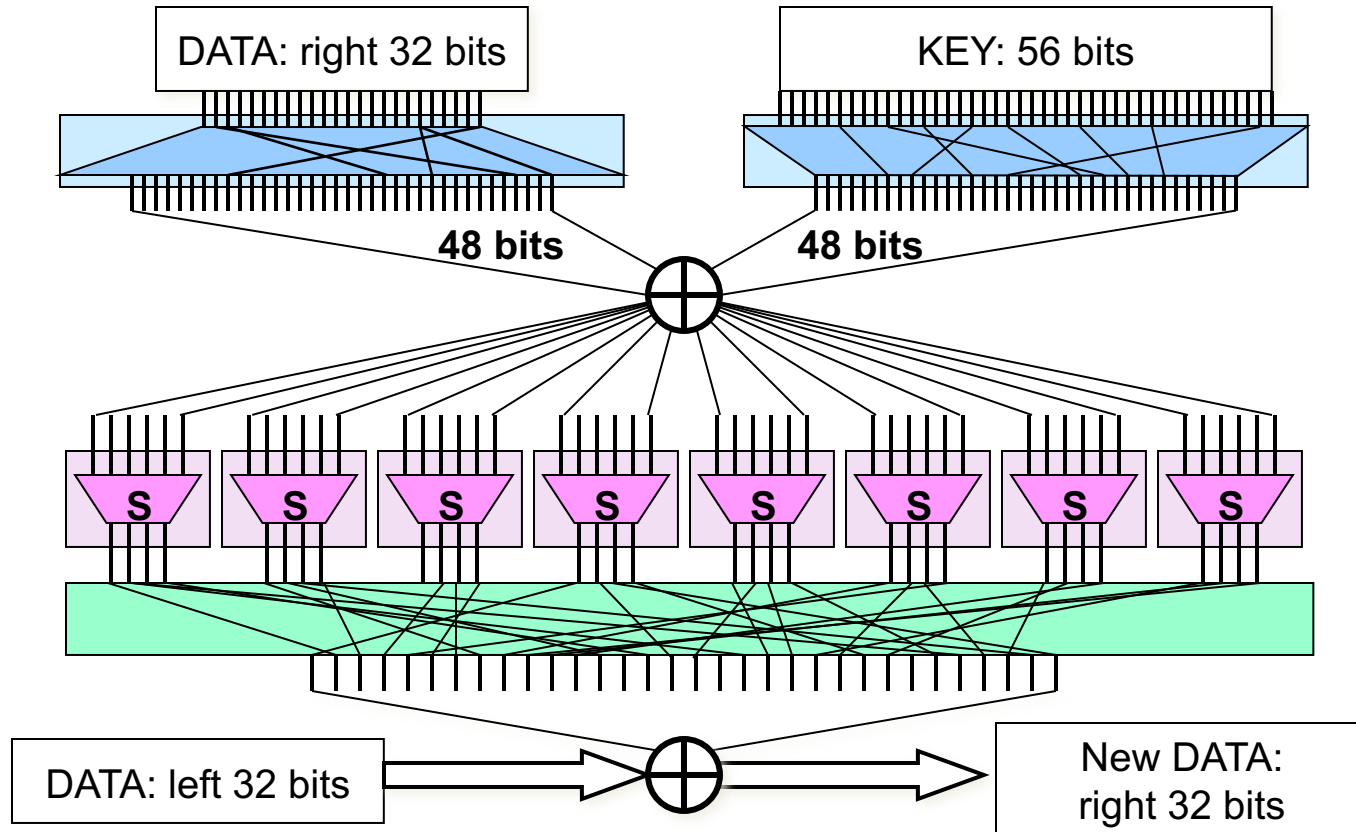 - 16 rounds

- DES is a type of **Feistel cipher**, which is a form of a **block cipher**

- **Plaintext block is split in two**
  - *Round* function applied to one half of the block
  - Output of the round function is XORed with other half of the block
  - Halves are swapped

- AES is *<u>not</u>* a Feistel cipher
  - Uses the entire block

Block of Plaintext		Key
Left half	Right half	

Round { (Round function) ⊕ subkey

| Output left half | Output right half |

DES

64-bit plaintext block

48-bit subkey
permuted from key

initial permutation, IP

left half, L_1

right half, R_0

f

K_1

16 rounds

$L_1 = R_0$

$R_1 = L_0 \oplus f(R_0, K_1)$

$L_{15} = R_{14}$

$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$

f

K_{16}

$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$

$L_{16} = R_{15}$

final permutation, IP^{-1}

64-bit ciphertext block

DES: S-boxes

- **After compressed key is XORed with expanded block**
  - 48-bit result moves to substitution operation via eight substitution boxes (s-boxes)
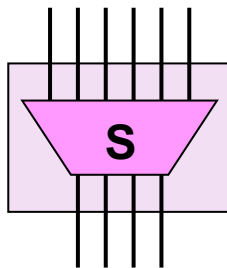
- **Each S-box has**
  - 6-bit input
  - 4-bit output

- **48 bits divided into eight 6-bit sub-blocks**

- **Each block is operated by a separate S-box**

- **Net result: 48-bit input generates 32-bit output**

- **S-boxes are key components of DES's security**

S-boxes are used in symmetric block ciphers to add _confusion_: hide the relationship of any ciphertext from any plaintext & key bits.

Implemented as a table lookup

Is DES secure?

- **56-bit key makes DES relatively weak**
  - $2^{56} = 7.2 \times 10^{16}$ keys
  - Brute-force attack

- **By the late 1990's:**
  - DES cracker machines built to crack DES keys in a few hours
  - DES Deep Crack: 90 billion keys/second
  - Distributed.net: test 250 billion keys/second

- **2000s < 1 day**
  - 2006: COPACOBANA: Custom FPGA-based DES cracker for < $10,000
  - 2012: cloud-based service
    crack MS-CHAPv2 authentication (which uses DES) on sale for $20 vs. $200

https://boingboing.net/2012/09/24/exhaust-all-of-des-and-crack-a.html

The power of 2

- Adding one extra bit to a key doubles the search space

- Suppose it takes 1 second to search through all keys with a 20-bit key

key length	number of keys	search time
20 bits	1,048,576	1 second
21 bits	2,097,152	2 seconds
32 bits	4.3×10^9	~ 1 hour
56 bits	7.2×10^{16}	2,178 years
64 bits	1.8×10^{19}	> 557,000 years
256 bits	1.2×10^{77}	3.5×10^{63} years

Distributed & custom hardware efforts typically allow us to test between 1 and >100 billion 64-bit (e.g., RC5) keys per second

Can double encryption work for DES?

Useless if we could find a key K such that:

$$E_K(P) = E_{K2}(E_{K1}(P))$$

This does not hold for DES (luckily!)

Double DES

Vulnerable to *meet-in-the-middle* attack

If we know some pair (*P, C*), then:

[1] Encrypt *P* for all 2^{56} values of K_1

[2] Decrypt *C* for all 2^{56} values of K_2

For each match where [1] == [2]

- Test the two keys against another *P, C* pair
- If match, you are assured that you have the key
- The complexity is 2×2^{56} rather than $2^{2 \times 56}$

Triple DES key lengths

Triple DES with two 56-bit keys (112-bit key):

$$C = E_{K1}(D_{K2}(E_{K1}(P)))$$

Triple DES with three 56-bit keys (168-bit key):

$$C = E_{K3}(D_{K2}(E_{K1}(P)))$$

Decryption used in middle step for compatibility with DES ($K_1=K_2=K_3$)

$$C = E_K(D_K(E_K(P))) \equiv C = E_{K1}(P)$$

DES Disadvantages

- **DES has been shown to have some weaknesses**
  - Key can be recovered using 2^{47} chosen plaintexts or 2^{43} known plaintexts
  - *Note that this is not a practical amount of data to get for a real attack!*

- **Short block size (8 bytes = 2^8 = 64 bits)**

- **The real weakness of DES is its 56-bit key**
  - Exhaustive search requires 2^{55} iterations on average

- **3DES solves the key size problem: we can have keys up to 168 bits**
  - Differential & linear cryptanalysis is not effective here: the three layers of encryption use 48 rounds instead of 16 making it infeasible to reconstruct s-box activity

- **But DES is relatively slow**
  - It was designed with hardware encryption in mind: and 3DES is 3x slower than DES

AES (Advanced Encryption Standard)

- **Block cipher: 128-bit blocks**
  - DES used 64-bit blocks

- **Successor to DES as a standard encryption algorithm**
  - DES: 56-bit key
  - AES: 128, 192, or 256-bit keys

From NIST:

> *Assuming that one could build a machine that could recover a DES key in a second (i.e., try 2^{56} keys per second), then it would take that machine approximately 149 trillion years to crack a 128-bit AES key. To put that into perspective, the universe is believed to be less than 20 billion years old.*

https://www.nist.gov/news-events/news/2001/12/commerce-secretary-announces-new-standard-global-information-security

AES (Advanced Encryption Standard)

- **Iterative cipher, just like most other block ciphers**
  - Each round is a set of substitutions & permutations

- **Variable number of rounds**
  - DES always used 16 rounds
  - AES:
    - 10 rounds: 128-bit key
    - 12 rounds: 192-bit key
    - 14 rounds: 256-bit key
  - A **subkey** ("round key") derived from the key is computed for each round
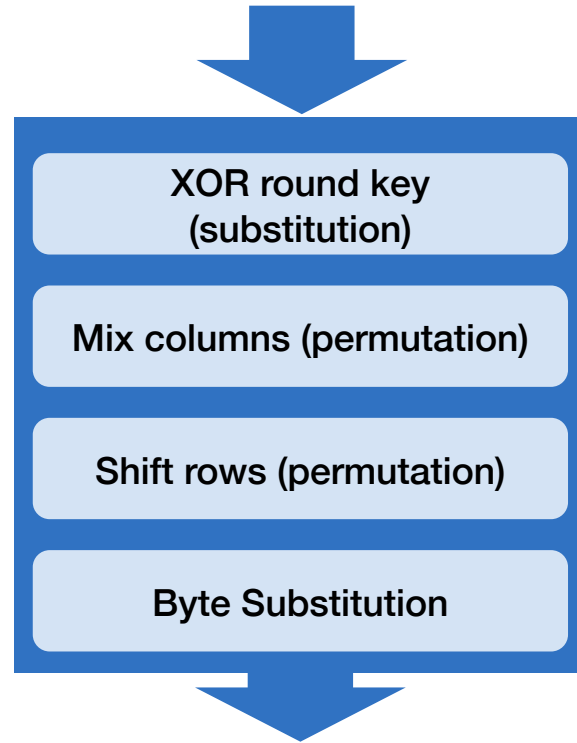   - DES used this too

Each AES Round

- **Step 1: Byte Substitution (s-boxes)**
  - Substitute 16 input bytes by looking each one up in a table (S-box)
  - Result is a 4x4 matrix

- **Step 2: Shift rows**
  - Each row is shifted to the left (wrapping around to the right)
  - 1st row not shifted; 2nd row shifted 1 position to the left; 3rd row shifted 2 positions; 4th row shifted three positions

- **Step 3: Mix columns**
  - 4 bytes in each column are transformed
  - This creates a new 4x4 matrix

- **Step 4: XOR round key**
  - XOR the 128 bits of the round key with the 16 bytes of the matrix in step 3
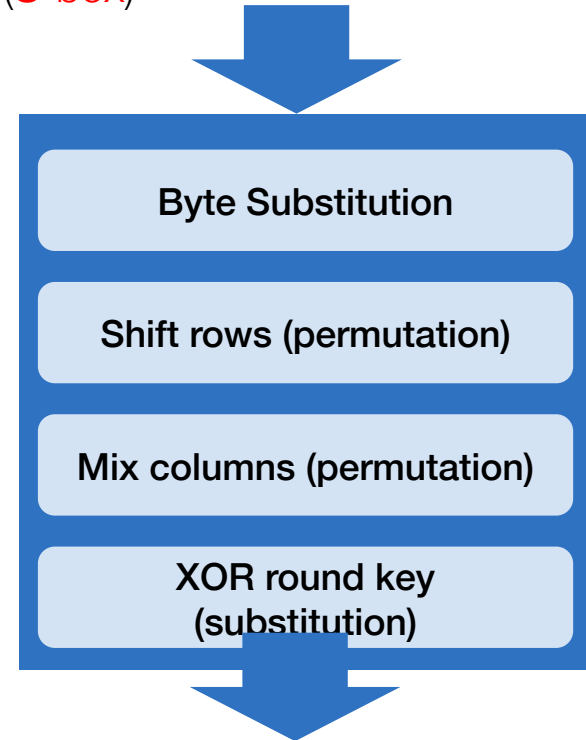
| Byte Substitution |
| Shift rows (permutation) |
| Mix columns (permutation) |
| XOR round key (substitution) |

Same rounds
… but in reverse order

XOR round key
(substitution)

Mix columns (permutation)

Shift rows (permutation)

Byte Substitution

AES Advantages

- **Larger block size**: 128 bits vs 64 bits

- **Larger & varying key sizes**: 128, 192, and 256 bits
  - 128 bits is complex enough to prevent brute-force searches

- **No significant academic attacks beyond brute force search**
  - Resistant against linear cryptanalysis thanks to bigger S-boxes
    - S-box = lookup table that adds non-linearity to a set of bits via transposition & flipping
  - DES: 6-bit inputs & 4-bit outputs
  - AES: 8-bit inputs & 8-bit outputs

- **Typically 5-10x faster in software than 3DES**

- **Attacks have been found**
  - This does **not** mean that AES is insecure!

- **Because of the attacks:**
  - AES-128 has computational complexity of $2^{126.1}$ (~126 bits)
  - AES-192 has computational complexity of $2^{189.7}$ (~190 bits)
  - AES-256 has computational complexity of $2^{254.9}$ (~255 bits)

- **Increasing AES security**
  - The security of AES can be increased by increasing the number of rounds in the algorithm
  - However, AES-128 still has a sufficient safety margin to make exhaustive search attacks impractical

Popular symmetric algorithms

AES (Advanced Encryption Standard)	• FIPS standard since 2002 • 128, 192, or 256-bit keys; operates on 128-bit blocks • By far the most widely used symmetric encryption algorithm
DES (Data Encryption Standard)	• FIPS standard from1976-2002 • 56-bit key; operates on 64-bit (8-byte) blocks • Triple DES recommended since 1999 (112 or 168 bits) • Not actively used anymore; AES is better by any measure
Blowfish	• Key length from 23-448 bits; 64-bit blocks • Optimized for 32-bit CPUs
Twofish	• Successor to Blowfish; key length from 128, 192, 256 bits; 128-bit blocks • Competed against AES for standardization
IDEA (International Data Encryption Algorithm)	• 128-bit keys; operates on 64-bit blocks • More secure than DES but AES is faster • Competed against AES for standardization

Cryptanalysis

Cryptographic attacks

- **Chosen plaintext**
  - Attacker can create plaintext and see the corresponding ciphertext

- **Known plaintext**
  - Attacker has access to both plaintext & ciphertext but doesn't get to choose the text

- **Ciphertext-only**
  - The attacker only sees ciphertext
  - Popular in movies but rarely practical in real life

Differential Cryptanalysis

Examine how changes in input affect changes in output

- **Discover where a cipher exhibits non-random behavior**
  - These properties can be used to extract the secret key
  - Applied to block ciphers, stream ciphers, and hash functions (functions that flip & move bits vs. mathematical operations)

- **Chosen plaintext attack is normally used**
  - Attacker must be able to choose the plaintext and see the corresponding cipher text

Differential Cryptanalysis

- **Provide plaintext with known differences**
  - See how those differences appear in the ciphertext

- **The properties depend on the key and the s-boxes in the algorithm**

- **Do this with lots and lots of known *plaintext-ciphertext* sets**

- **Statistical differences, if found, may allow a key to be recovered faster than with a brute-force search**
  - You may deduce that certain keys are not worth trying

Linear Cryptanalysis

Create a predictive approximation of inputs to outputs

- **Instead of looking for differences, linear cryptanalysis attempts to come up with a linear formula (e.g., a bunch of *xor* operations) that connects certain input bits, output bits, and key bits with a probability higher than random**
  - Goal is to approximate the behavior of s-boxes

- **Part 1: construct linear equations**
  - Find high correlations

- **Part 2: guess key bits**
  - Guess enough bits so that a brute force attack becomes feasible

Linear Cryptanalysis

It will _not_ recreate the working of the cipher

- You just hope to find non-random behavior that gives you insight on what bits of the key might matter

- **Works better than differential cryptanalysis for known plaintext**

  - Differential cryptanalysis works best with chosen plaintext

- **Linear & differential cryptanalysis will rarely recover a key but may be able to reduce the number of keys that need to be searched**

Cipher modes

CS 419 © 2022 Paul Krzyzanowski

Not a good idea to use block ciphers directly

- **Streams of data are broken into *k*-byte blocks**
  - Each block encrypted separately
  - This is called **Electronic Codebook** (**ECB**)

- **Problems**
  1. Same plaintext results in identical encrypted blocks
     Enemy can build up a code book of plaintext/ciphertext matches
 2. Attacker can add/delete/replace blocks

P_0	P_1	P_2	P_3	P_4	-----------------
C_0	C_1	C'_2	C_3	C_4	-----------------
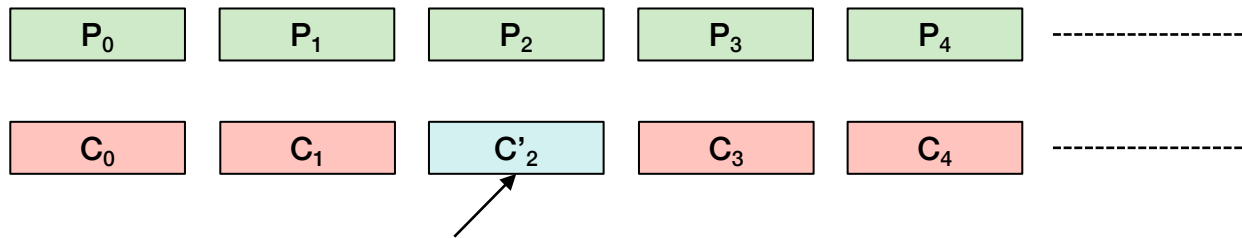
Intruder can replace blocks (e.g., with ciphertext from previous messages)

Cipher Block Chaining (CBC) mode

- **Random initialization vector (IV) = bunch of *k* random bits**
  - Non-secret: both parties need to know this

- **Exclusive-or with first plaintext block – then encrypt the block**

- **Take exclusive-or of the result with the next plaintext block**
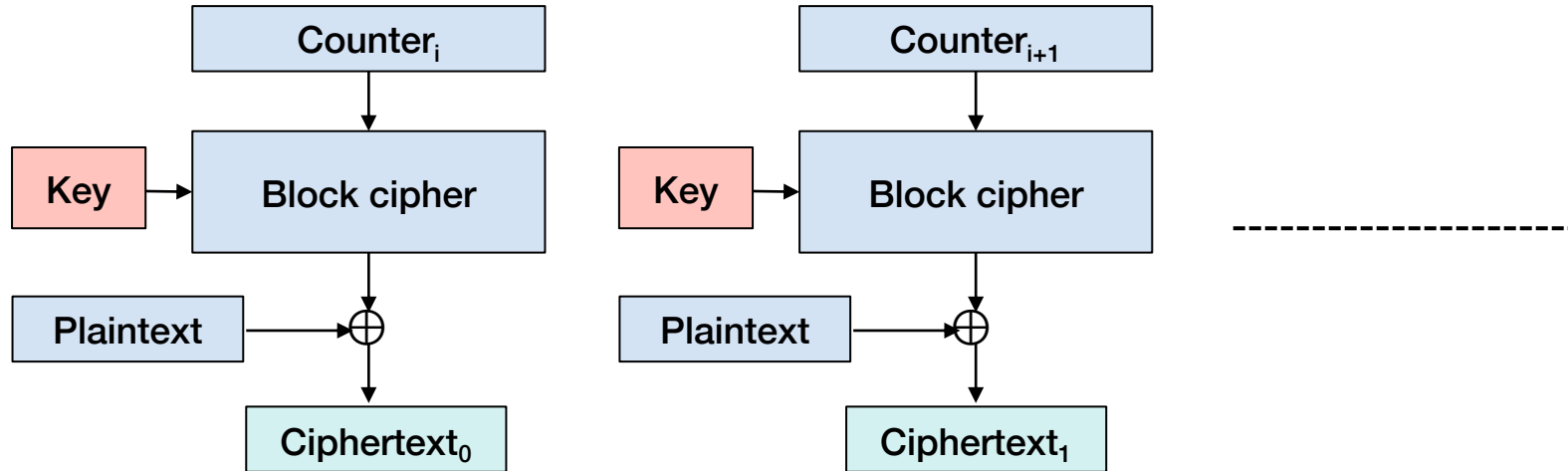
$$c_i = E_K(m_i) \oplus c_{i-1}$$
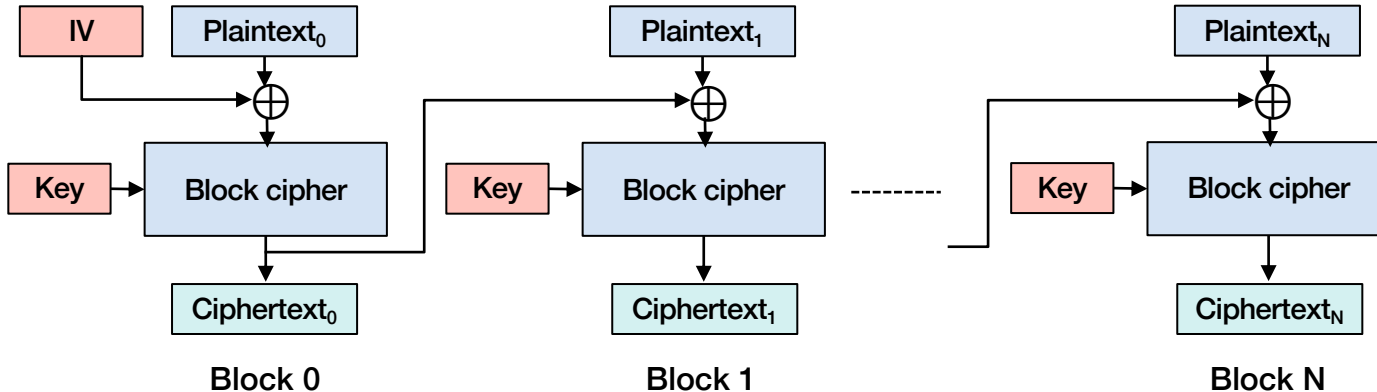
CBC Observations

- **Identical plaintext does not produce the same ciphertext**
  - Unless all previous blocks are identical

- **Each block is a function of all previous blocks**

- **An attacker can still cause data corruption**

Block encryption: Counter (CTR) mode

- **Random starting counter = bunch of *k* random bits, just like IV**
  - Any function producing a non-repeating sequence
    (an incrementing number is a common function)

- **Encrypt the counter with the key**

- **Exclusive-or result with plaintext block**

The End