

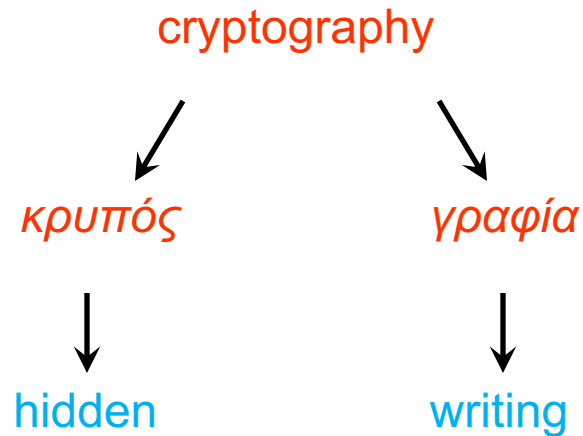
Computer Security

06. Cryptography

Paul Krzyzanowski

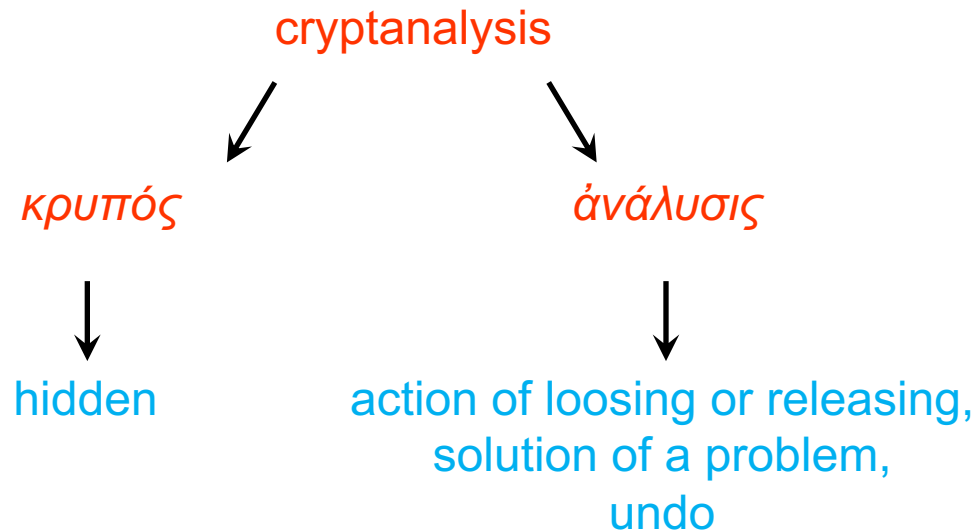
Rutgers University

Fall 2019



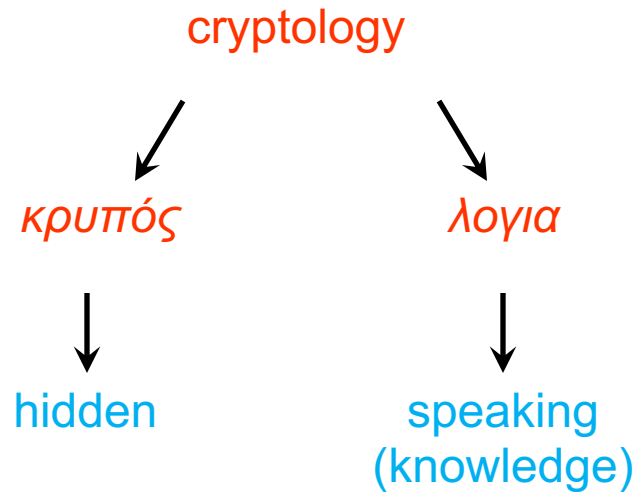
A secret manner of writing, ... Generally, the art of writing or solving ciphers.

— *Oxford English Dictionary*



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

— *Oxford English Dictionary*



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 \neq 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?

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

Terms

Plaintext (cleartext) message P

Encryption $E(P)$

Produces **Ciphertext**, $C = E(P)$

Decryption, $P = D(C)$

Cipher = cryptographic algorithm

Restricted cipher

Secret algorithm

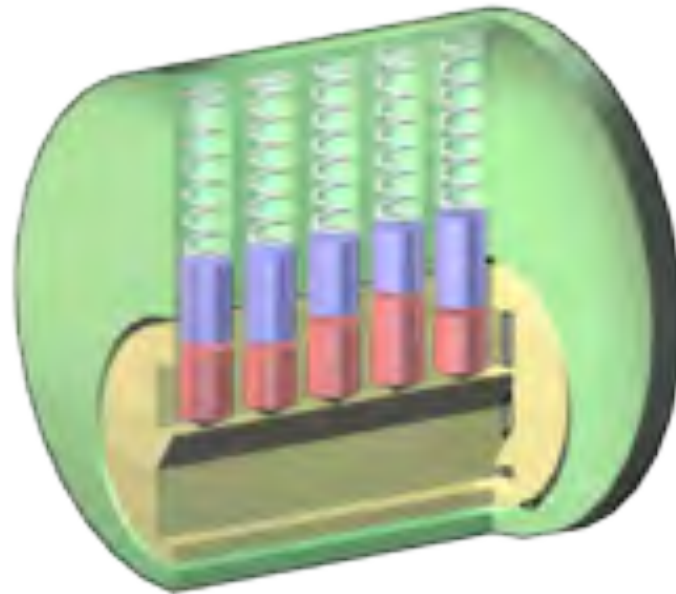
- Vulnerable to:
 - Leaking
 - Reverse engineering
 - HD DVD (Dec 2006) and Blu-Ray (Jan 2007)
 - RC4
 - All digital cellular encryption algorithms
 - DVD and DIVX video compression
 - Firewire
 - Enigma cipher machine
 - Every NATO and Warsaw Pact algorithm during Cold War
- Hard to validate its effectiveness (who will test it?)
- Not a viable approach!

Shared algorithms & secret keys

The key

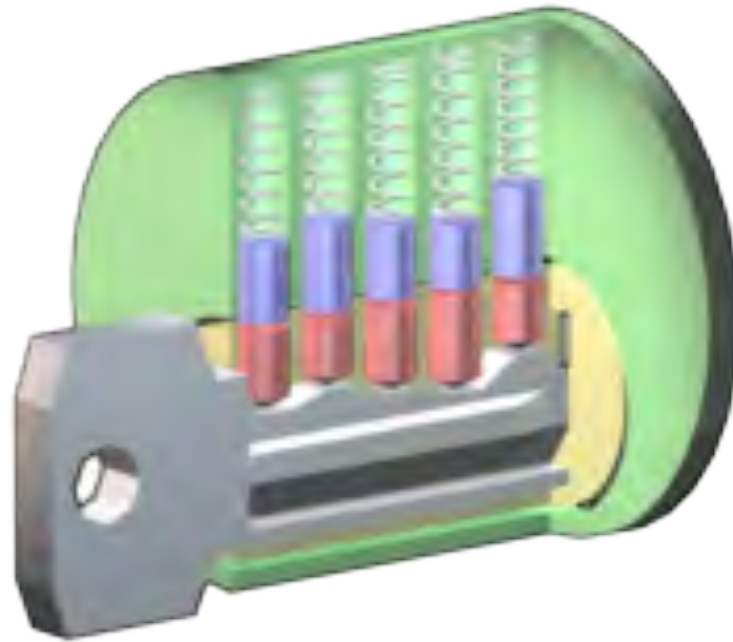


The lock



Source: en.wikipedia.org/wiki/Pin_tumbler_lock

The key & lock



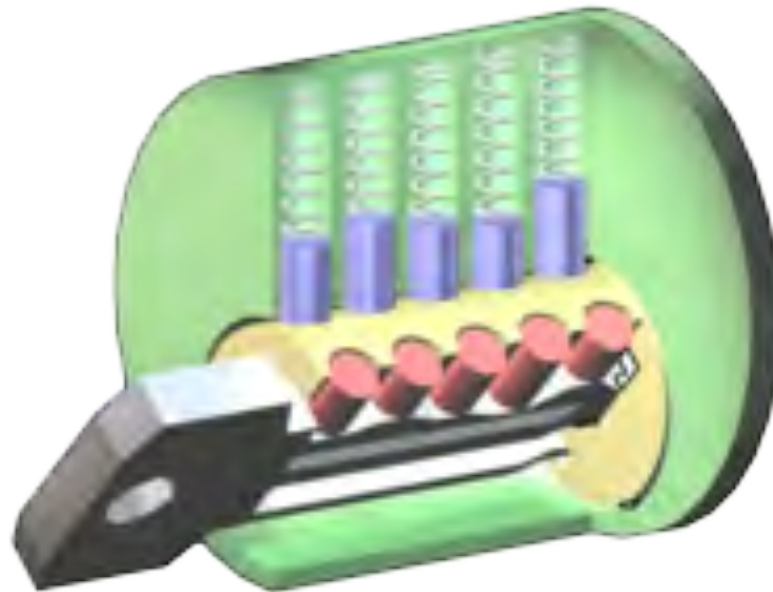
Source: en.wikipedia.org/wiki/Pin_tumbler_lock

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

Same shared secret key, K , for encryption & decryption

$$C = E_K(P)$$

$$P = D_K(C)$$

Classic Cryptosystems

Substitution Ciphers

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
 $n = \text{shift value} = \text{key}$

Substitution scheme used in India: 400 BCE – 200 CE

- Phonetic substitution

Last seen as ROT13 on Usenet & on-line forums to keep the reader from seeing offensive messages unwillingly

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

GSWUN

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

GSWUNB

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

GSWUNBU

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

GSWUNBUM

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

GSWUNBUMZ

Cæsar cipher

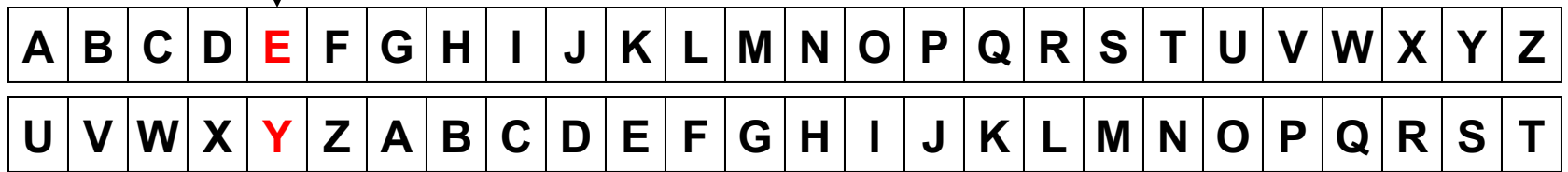
MY CAT HAS FLEAS

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

GSWUNBUMZF

Cæsar cipher

MY CAT HAS FLEAS

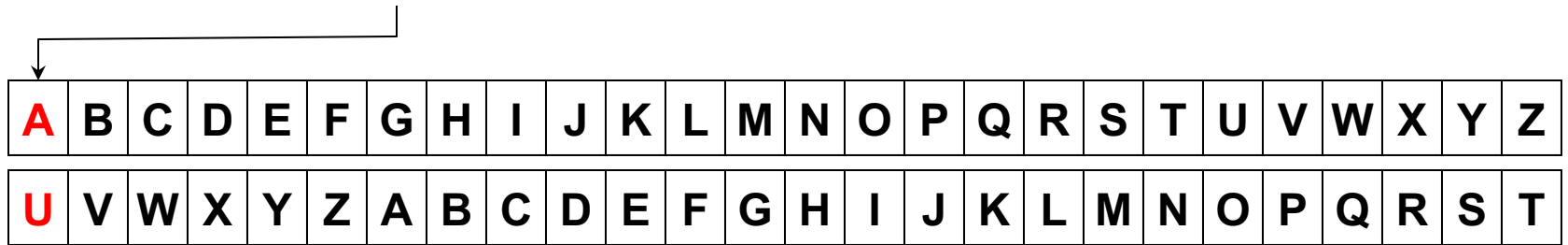


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

GSWUNBUMZFY

Cæsar cipher

MY CAT HAS FLEAS



GSWUNBUMZFYU

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

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)

Atbash (אתבש) – Ancient Hebrew variant

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

- c. 600 BCE
- No information (key) needs to be conveyed!

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

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 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%
o	7.282%
d	4.015%
b	1.773%
x	0.108%

e	11.797%
o	8.299%
d	3.943%
b	1.634%
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:

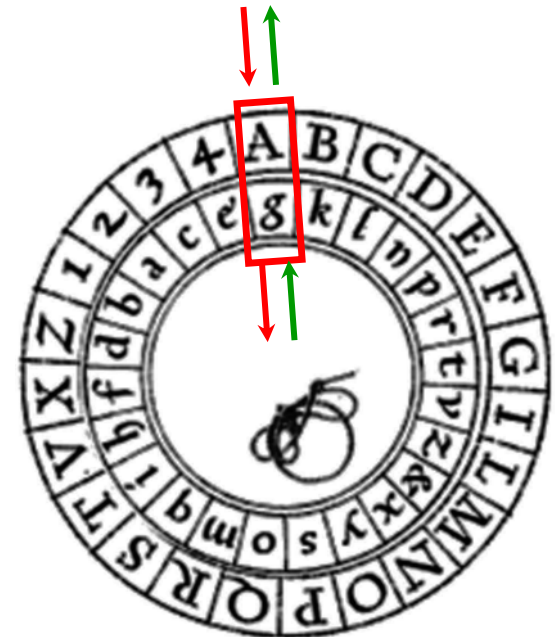
TH (3.56%), HE (3.07%), IN (2.43%), ER (2.05%), AN, RE, ...

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 substitute
- 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

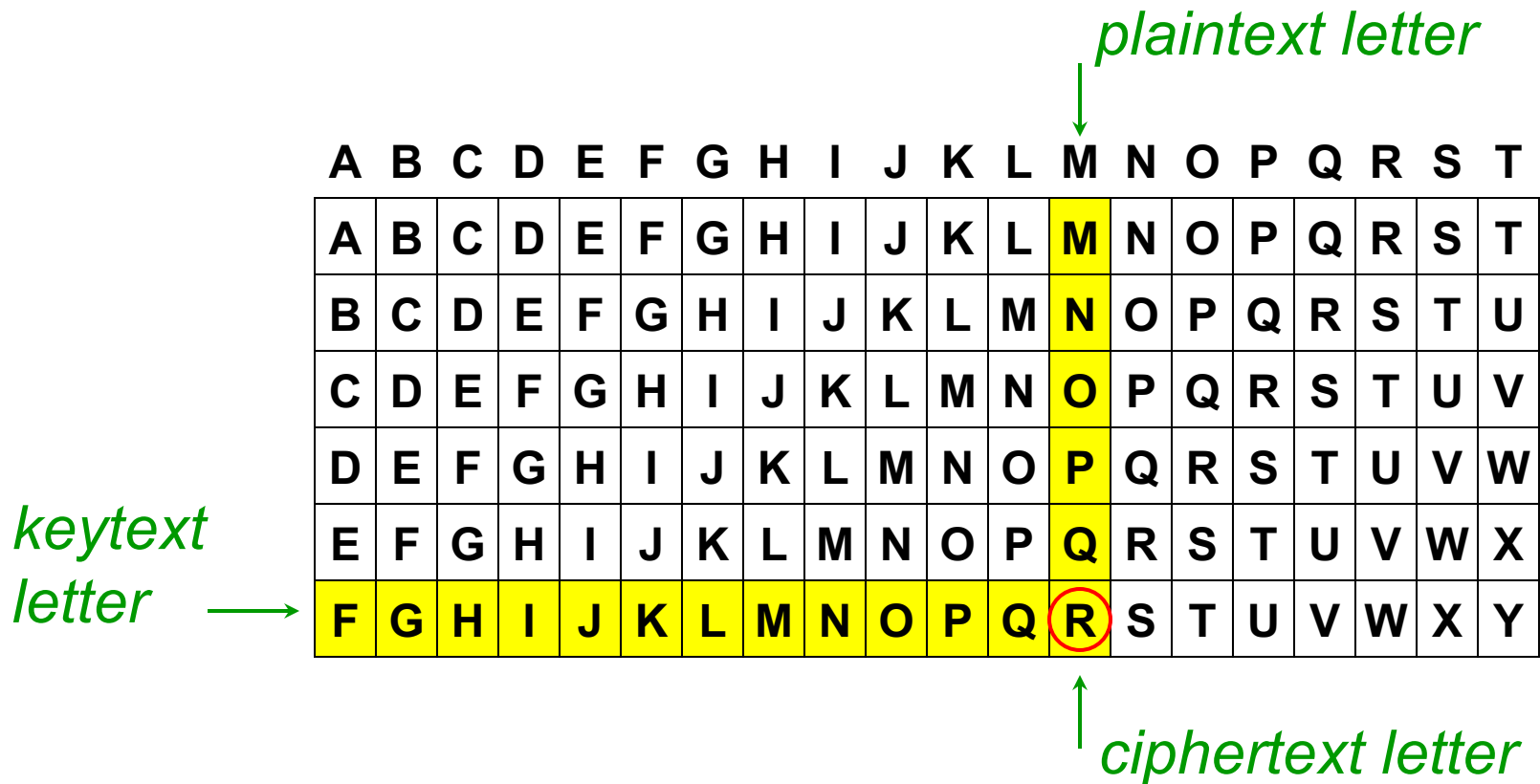
- No need for a disk: use table and key word to encipher a message
- Repeat keyword over text: (e.g. key=FACE)

FA CEF ACE FACEF
MY CAT HAS FLEAS

Running key

- **Encrypt:** find intersection:
row = keyword letter
column = plaintext letter
- **Decrypt:** column = keyword letter, search for intersection
= ciphertext 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 H

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 HC

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

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 **K**

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 KL

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 KL**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
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 KLGE

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 KLGE**X**

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>

Cryptoanalysis of the Vigenère cipher

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

Cryptoanalysis of the Vigenère cipher

1. Determine key length

- Count coincidences – identical sets of characters n characters apart

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.

One-time pad

Only provably secure encryption scheme

- Invented in 1917
- Large non-repeating set of *random* key letters originally written on a pad
- 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

One-time pad

If pad contains

KWXOPWMAELGHW...

and we want to encrypt

MY CAT HAS FLEAS

Ciphertext =

WUZOIDMSJWKHO

$$M + K \bmod 26 = W$$

$$Y + W \bmod 26 = U$$

$$C + X \bmod 26 = Z$$

$$A + O \bmod 26 = O$$

$$T + P \bmod 26 = I$$

$$H + W \bmod 26 = D$$

$$A + M \bmod 26 = M$$

$$S + A \bmod 26 = S$$

$$F + E \bmod 26 = J$$

$$L + L \bmod 26 = W$$

$$E + G \bmod 26 = K$$

$$A + H \bmod 26 = H$$

$$S + W \bmod 26 = O$$

One-time pad

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

Same ciphertext:

WUZOIDMSJWKHO

With a pad containing:

KWXOPWMAELGHW...

Produces:

THE DOG IS HAPPY

$$W - D \pmod{26} = T$$

$$U - N \pmod{26} = H$$

$$Z - V \pmod{26} = E$$

$$O - L \pmod{26} = D$$

$$I - U \pmod{26} = O$$

$$D - X \pmod{26} = G$$

$$M - E \pmod{26} = I$$

$$S - A \pmod{26} = S$$

$$J - C \pmod{26} = H$$

$$W - W \pmod{26} = A$$

$$K - V \pmod{26} = P$$

$$H - S \pmod{26} = P$$

$$O - Q \pmod{26} = Y$$

One-time pads in computers

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

One-time pad – C code

```
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 can be problematic
 - May need to store a lot of data
- 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)

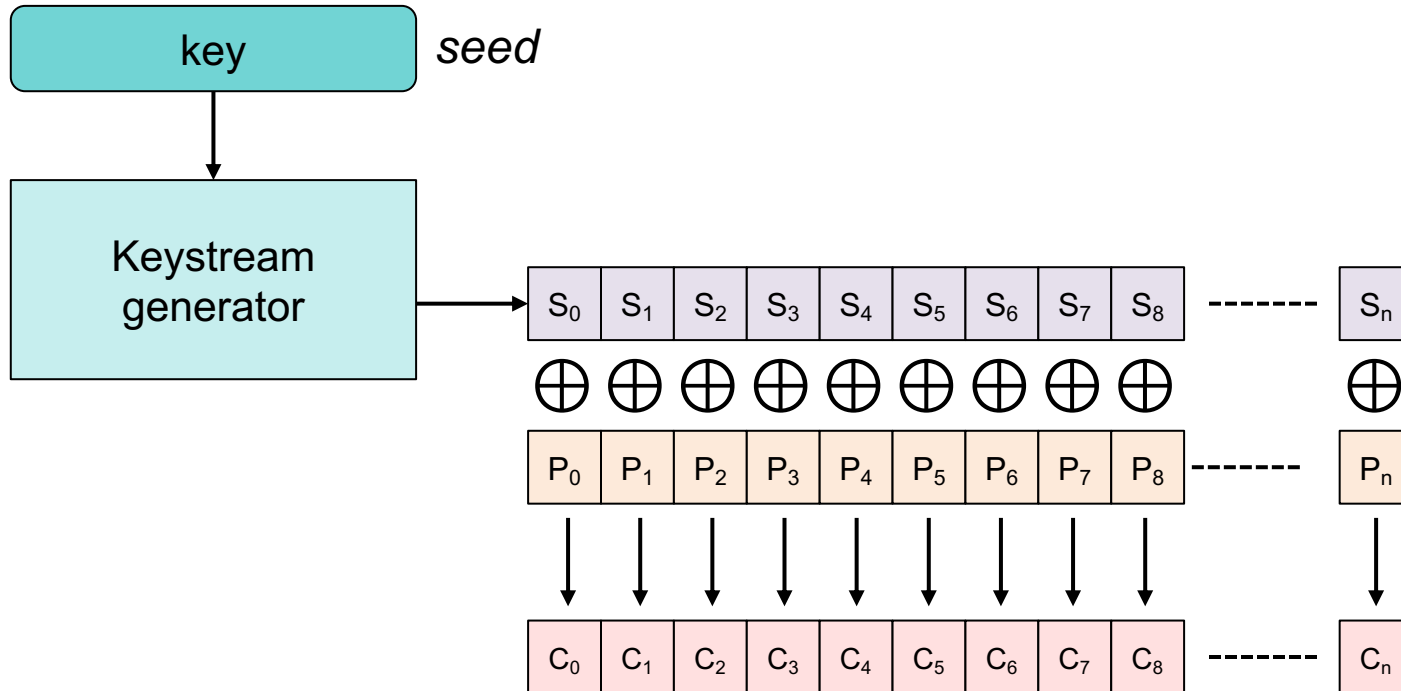
Random numbers

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

- John vonNeumann
- 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

Stream ciphers

Key stream generator produces a sequence of pseudo-random bytes



$$C_i = S_i \oplus P_i$$

Stream ciphers

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 and see if B makes sense

Or... if you have **known plaintext** and the corresponding ciphertext $\{ A, C \}$, you can extract the key:

$$K = A \oplus C$$

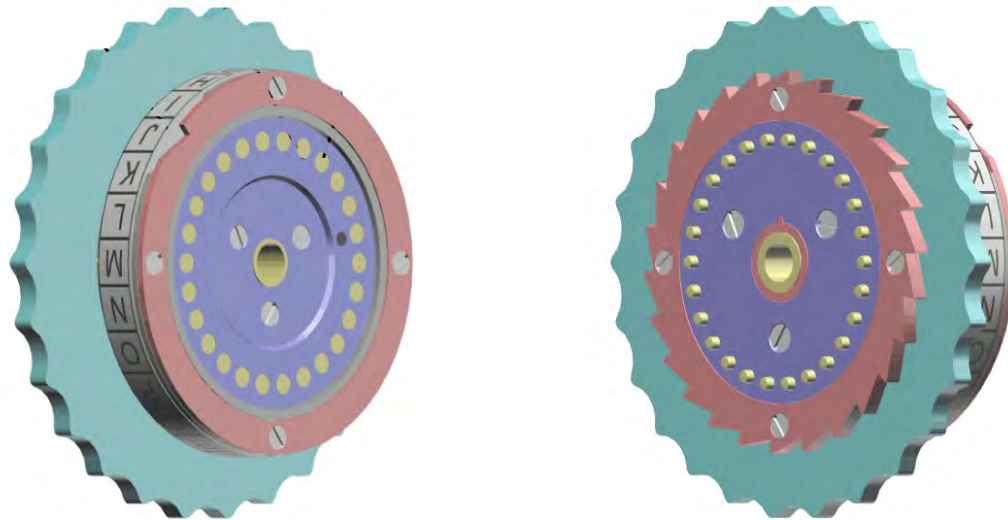
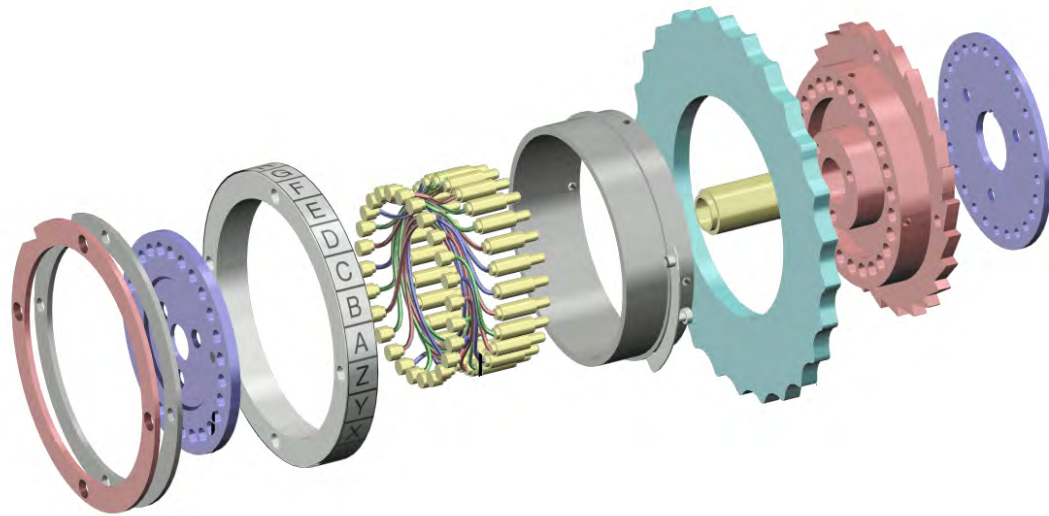
Electro-mechanical cryptographic engines

Rotor machines

1920s: mechanical devices used for automating encryption

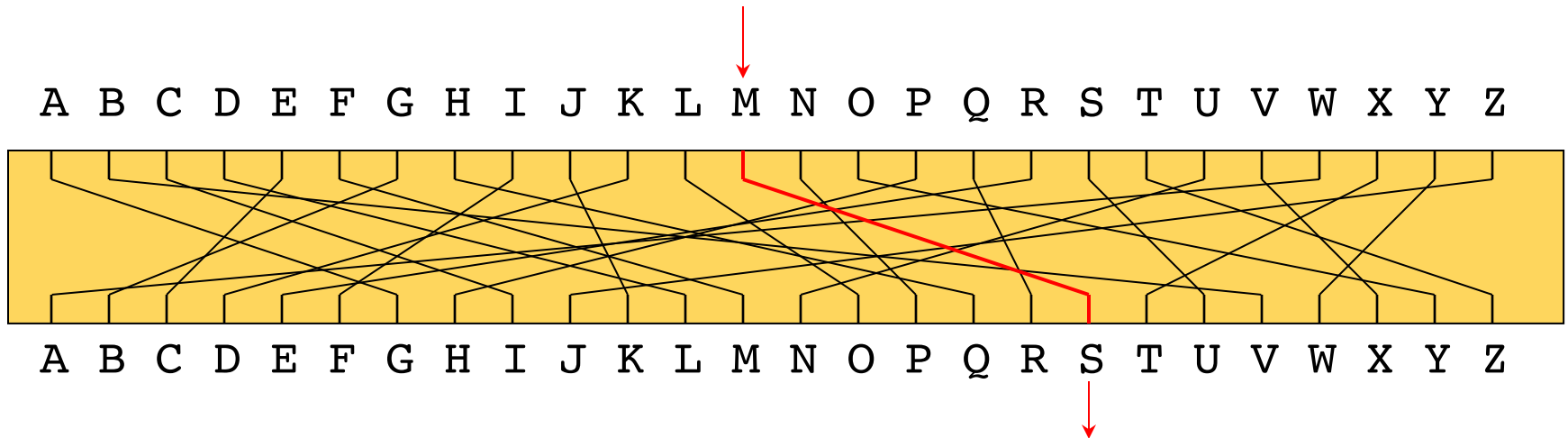
Rotor machine:

- 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



Rotor machines

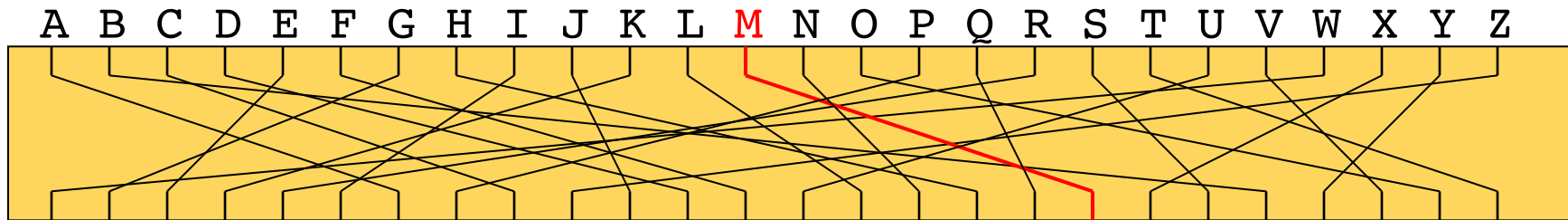
Simplest rotor machine: single cylinder



After a character is entered, the cylinder rotates one position

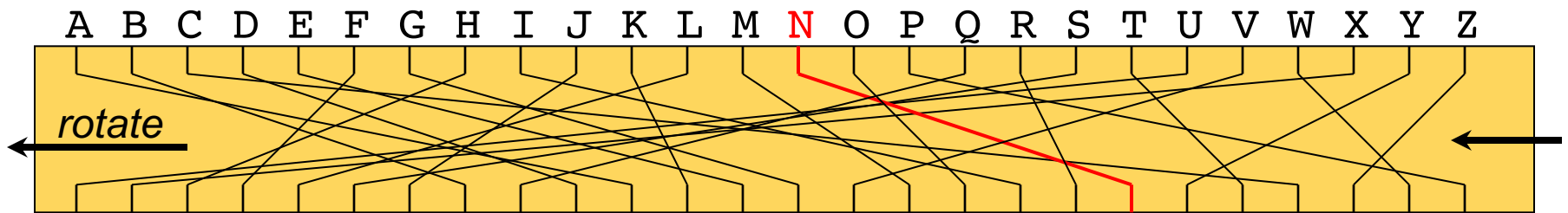
- Internal combinations shifted by one
- Polyalphabetic substitution cipher with a period of 26

Single cylinder rotor machine



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



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

Multi-cylinder rotor machines

Single cylinder rotor machine

- Substitution cipher with a period = length of alphabet (e.g., 26)

Multi-cylinder rotor machine

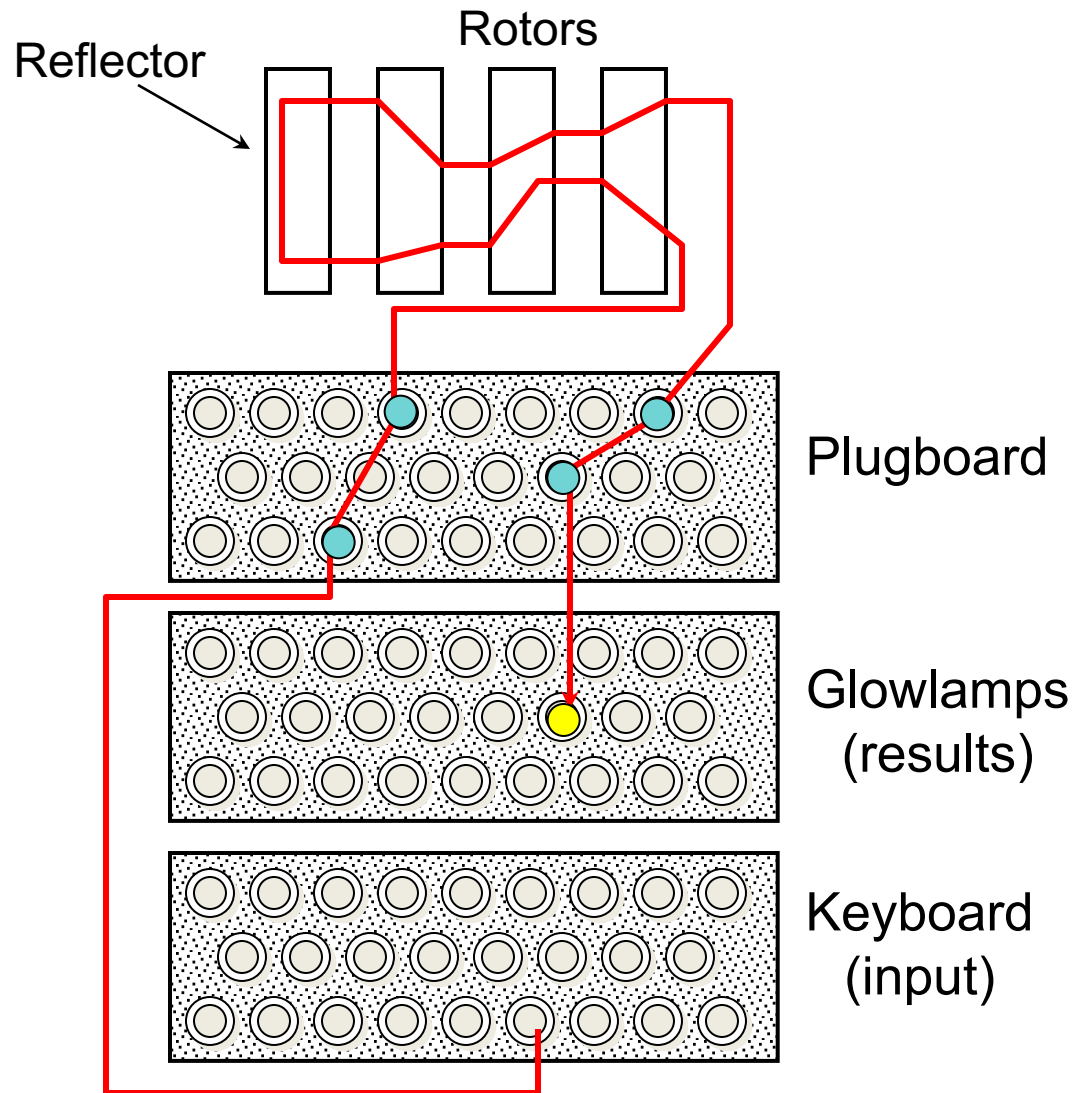
- Feed output of one cylinder as input to the next one
- First rotor advances after character is entered
- Second rotor advances after a full period of the first
- Polyalphabetic substitution cipher
 - Period = (length of alphabet)^{number of rotors}
 - 3 26-char cylinders $\Rightarrow 26^3 = 17,576$ 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
⇒ *makes encryption symmetric*
- Need to know initial settings of rotor
 - setting was $f(\text{date})$ in a book of codes
- Broken by group at Bletchley Park (Alan Turing)



Enigma



Transposition Ciphers

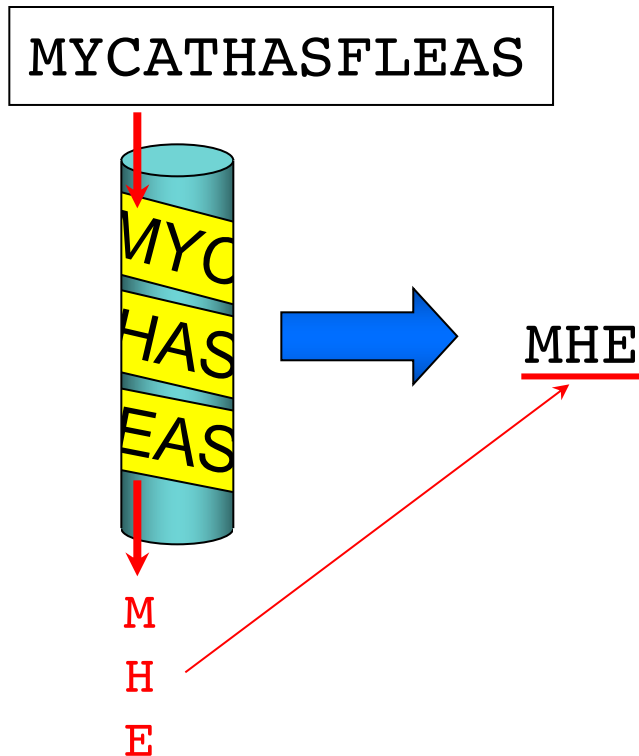
Transposition ciphers

- Permute letters in plaintext according to rules
- Knowledge of rules will allow message to be decrypted
- First mentioned in Greece in the 7th century BC
 - **Scytale** (*rhymes with Italy*) = staff cipher

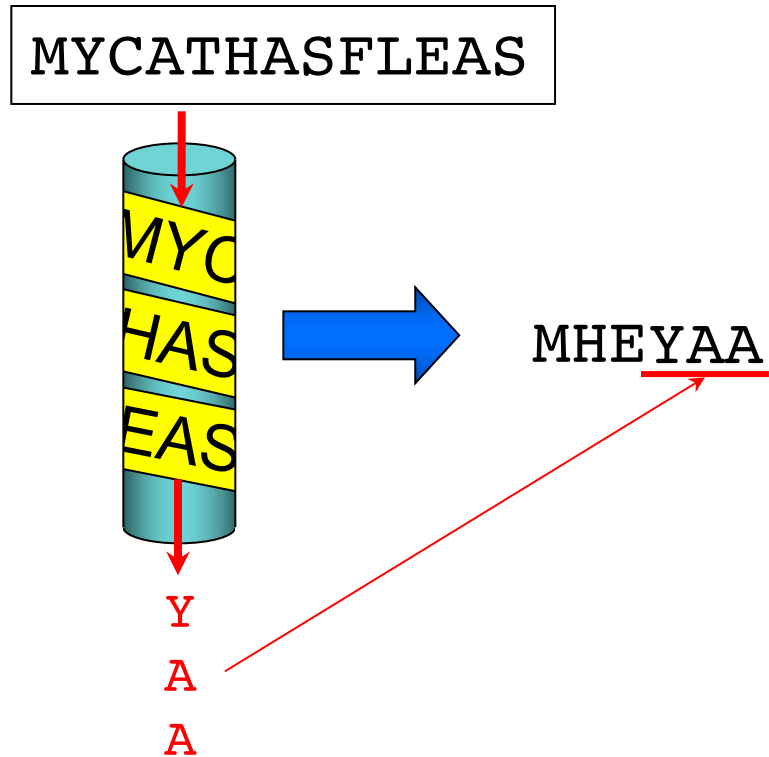


Transposition ciphers: **scytale**

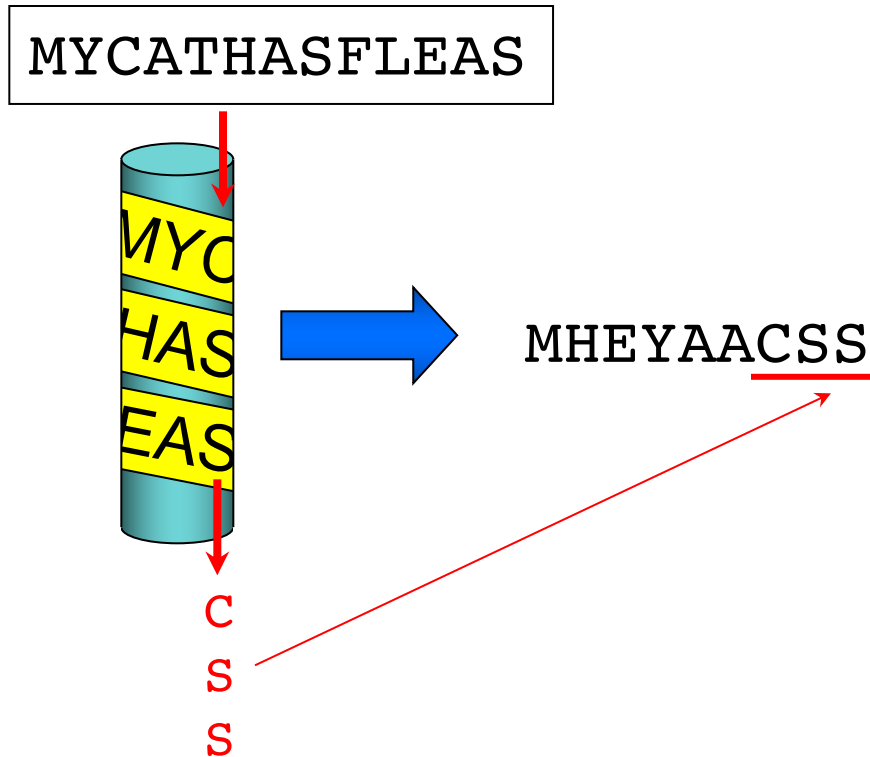
Secret = diameter of scytale



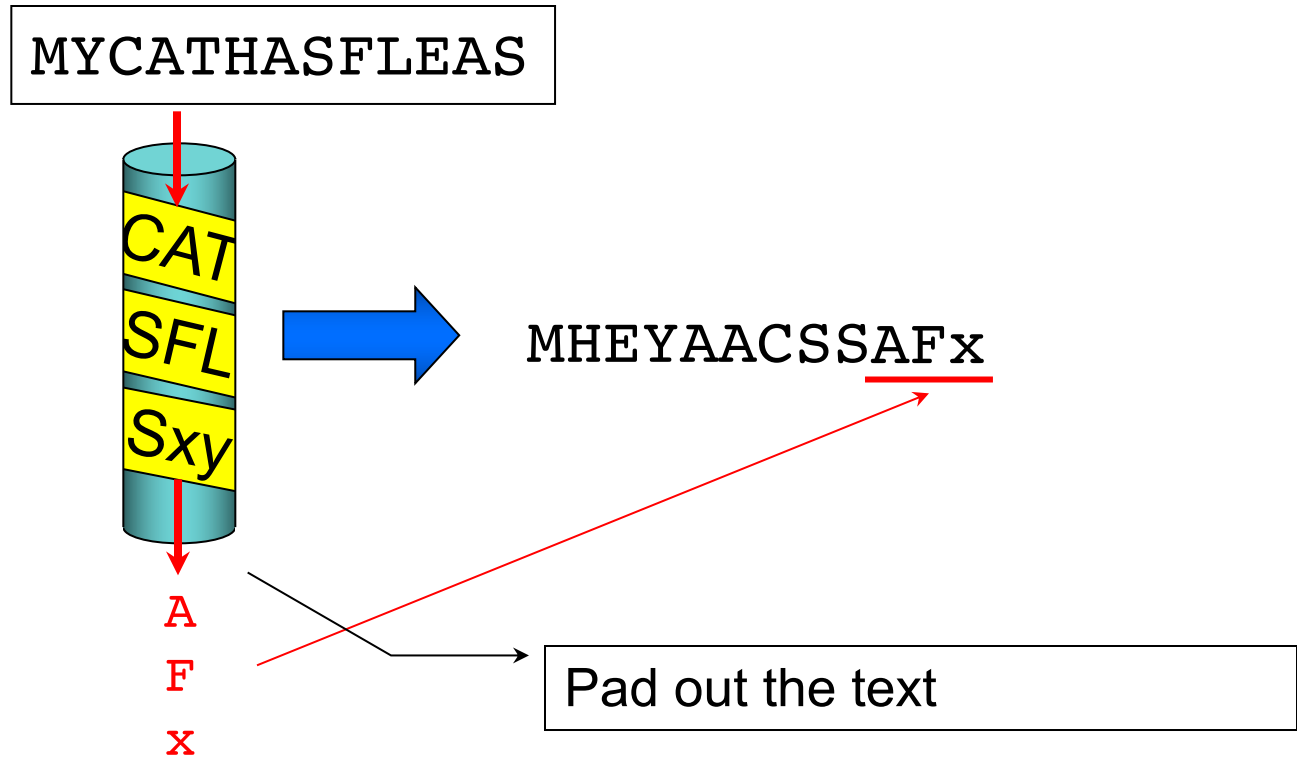
Transposition ciphers: scytale



Transposition ciphers: scytale



Transposition ciphers: scytale



Transposition ciphers: scytale

MYCATHASFLEAS



MHEYAACSSAFxTLy

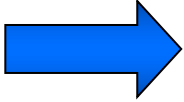
T
L
Y

scytale as a set of columns

Table version of scytale

- enter data horizontally, read it vertically
- secrecy is the width of the table

MYCATHASFLEAS

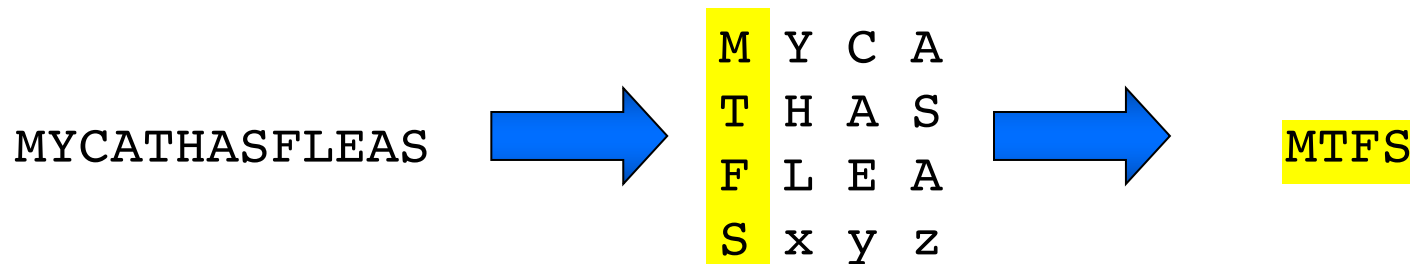


M	Y	C	A
T	H	A	S
F	L	E	A
S	x	y	z

scytale as a set of columns

Table version of scytale

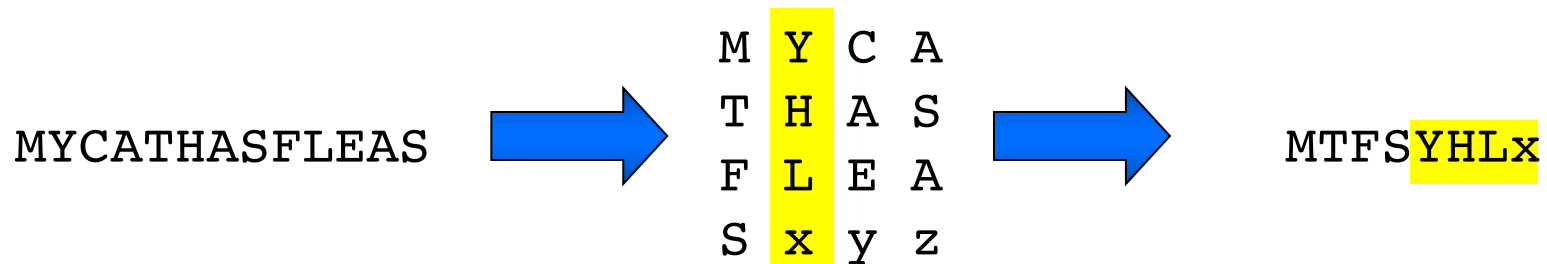
- enter data horizontally, read it vertically
- secrecy is the width of the table



scytale as a set of columns

Table version of scytale

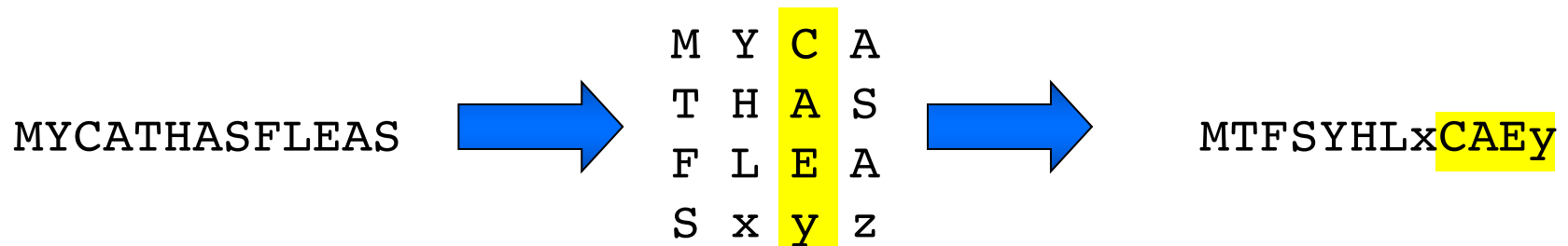
- enter data horizontally, read it vertically
- secrecy is the width of the table



scytale as a set of columns

Table version of scytale

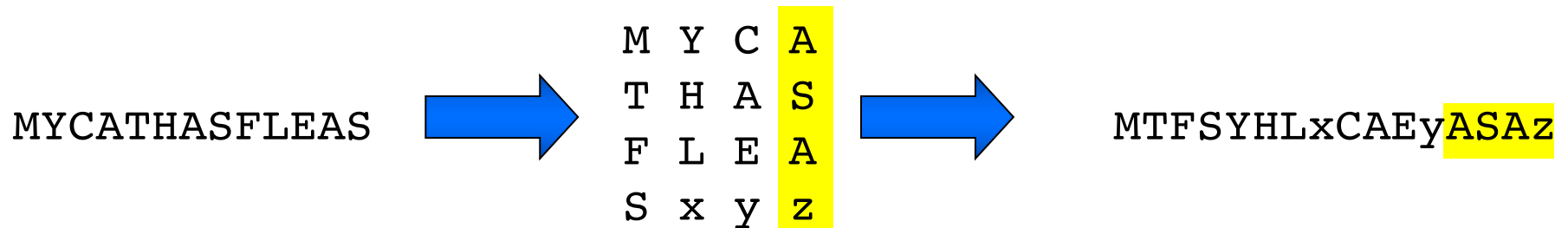
- enter data horizontally, read it vertically
- secrecy is the width of the table



scytale as a set of columns

Table version of scytale

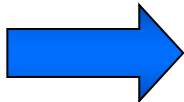
- enter data horizontally, read it vertically
- secrecy is the width of the table



Columnar transposition cipher

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

MYCATHASFLEAS

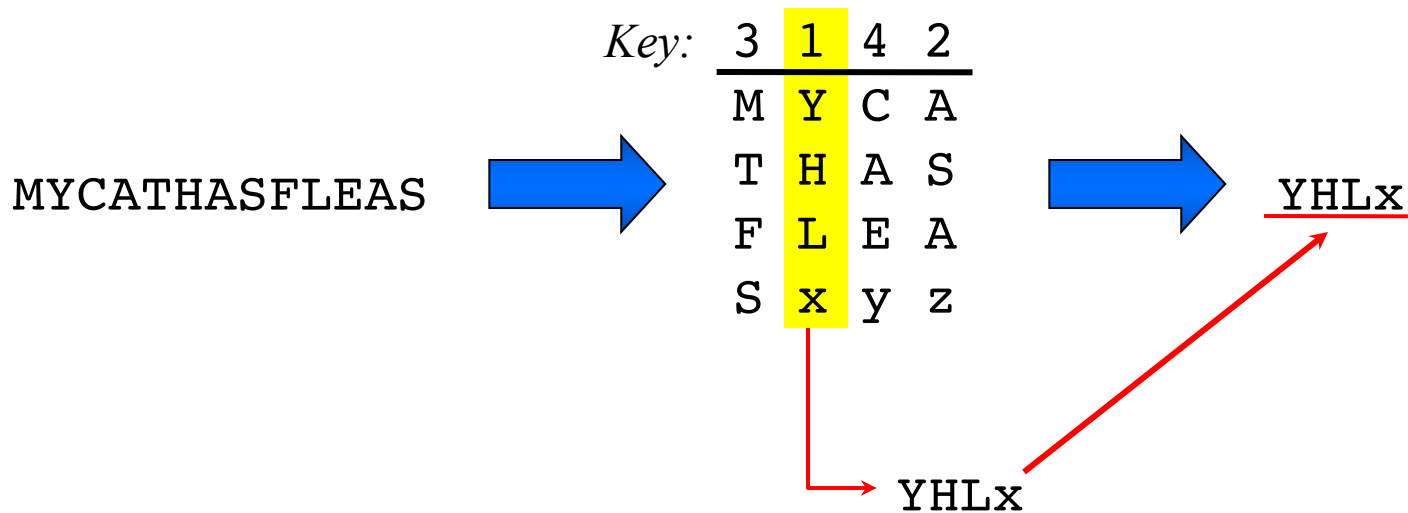


Key: 3 1 4 2

M	Y	C	A
T	H	A	S
F	L	E	A
S	x	y	z

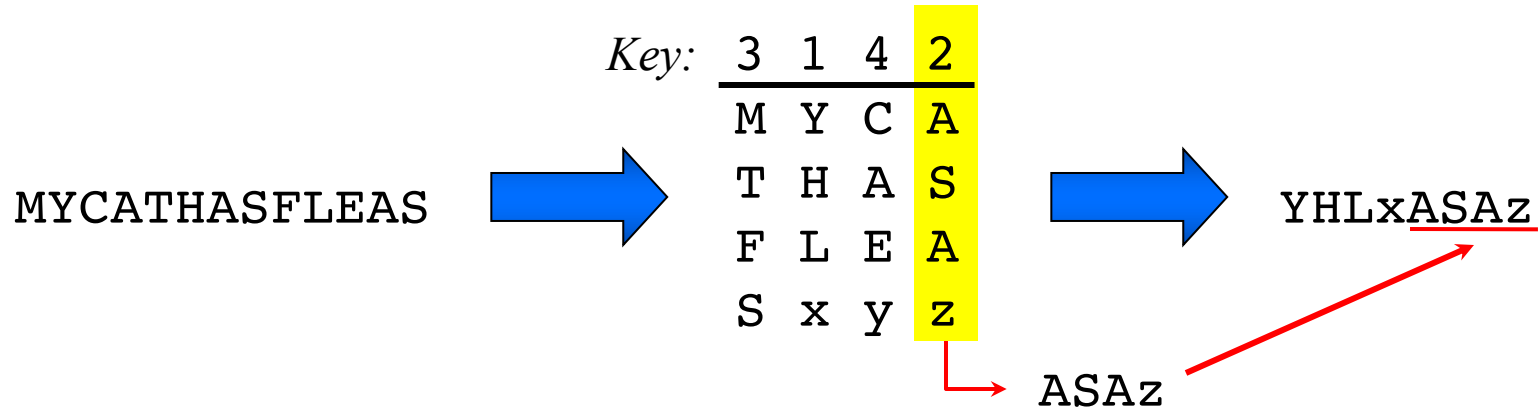
Columnar transposition cipher

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



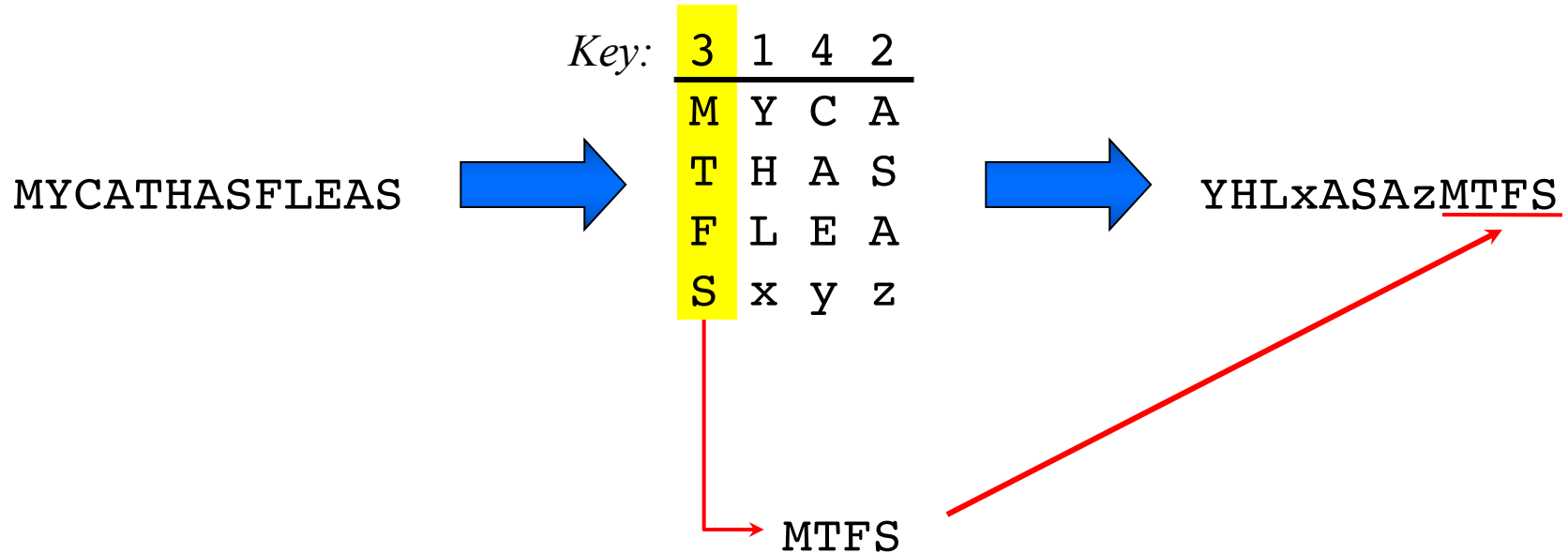
Columnar transposition cipher

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



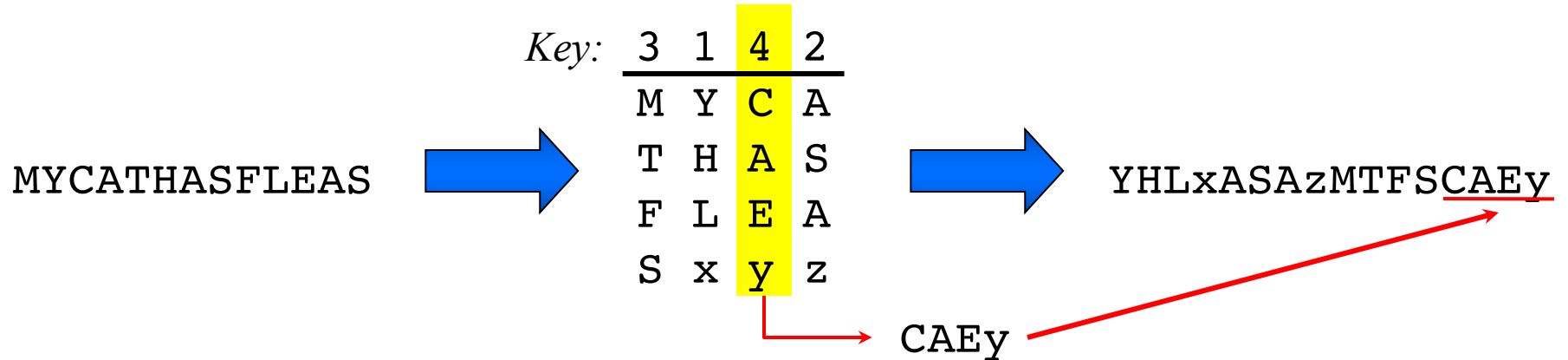
Columnar transposition cipher

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



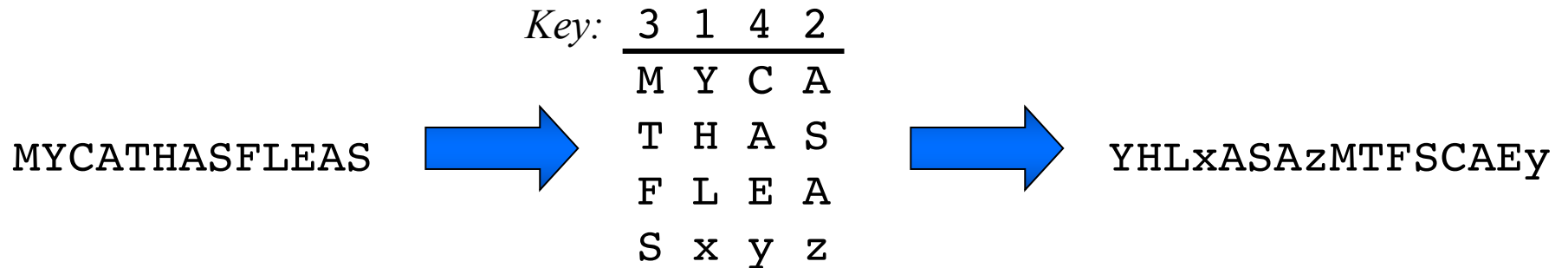
Columnar transposition cipher

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



Columnar transposition cipher

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



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 manual cryptography

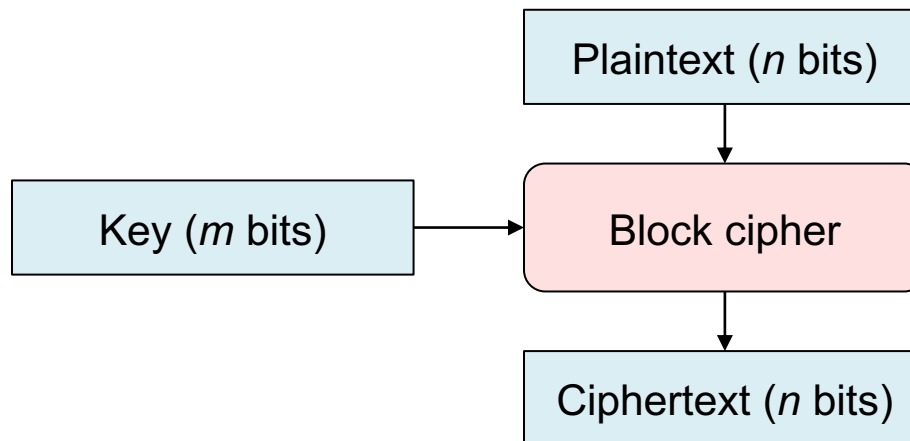
Computer Cryptography

Block ciphers

Block ciphers dominate computer cryptography

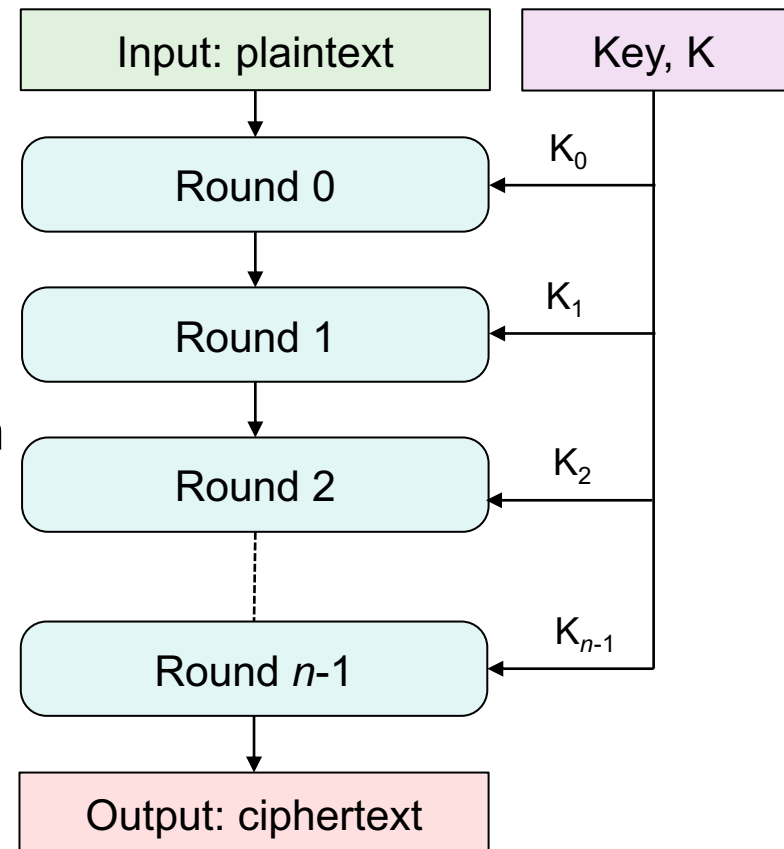
Encrypt a fixed number of bits at a time

Output blocksize (usually) = input blocksize



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



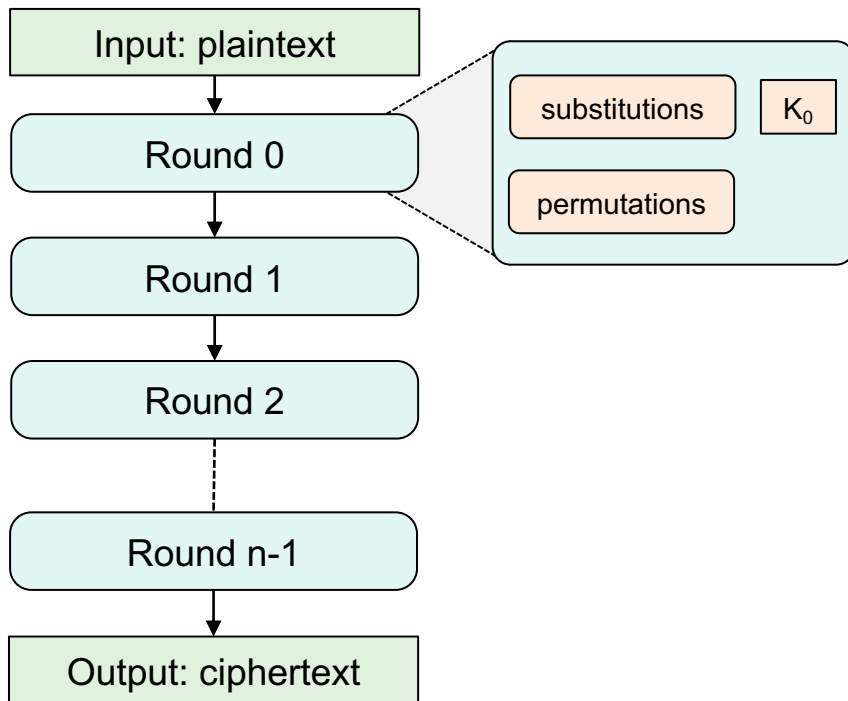
Structure of block ciphers

- Multiple **rounds** of combining the plaintext with the key
- Optional:
 - Convert key to internal form (possibly different per 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**



Substitution = **S-box**

- Table lookup
- Converts a small block of input to a block of output

Permutation

- Scrambles the bits in a prescribed order

Key application per round

- **Subkey** per round derived from the key
- Can drive behavior of s-boxes
- May be XORed with the output of each round

Create Confusion & Diffusion

- **Confusion**: no direct correlation between a bit of the key and resulting ciphertext
- **Diffusion**: Changing one bit of input should change, on average, $\frac{1}{2}$ of output bits

DES

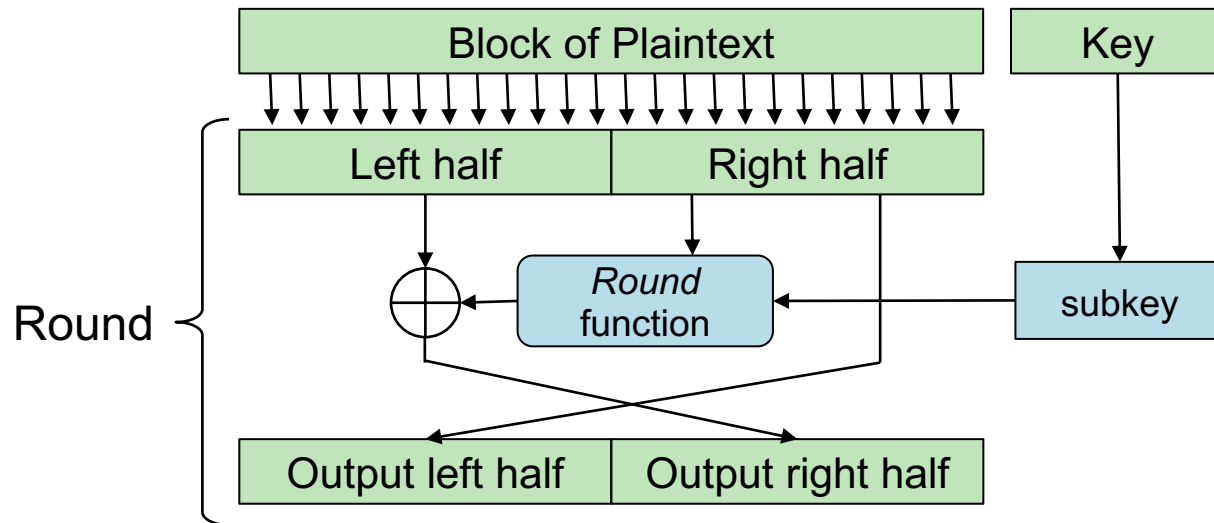
Data Encryption Standard

- 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

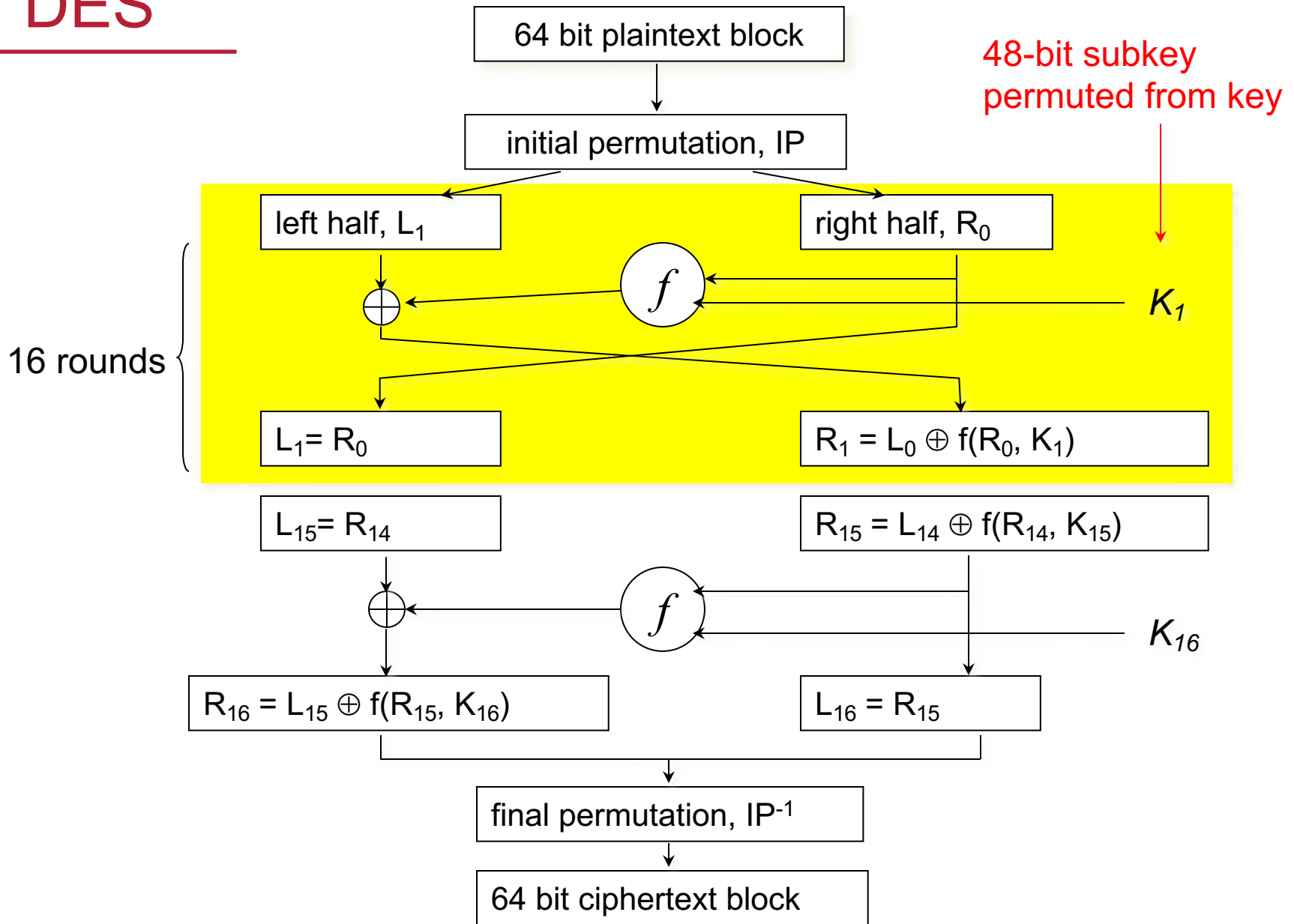
Feistel cipher

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

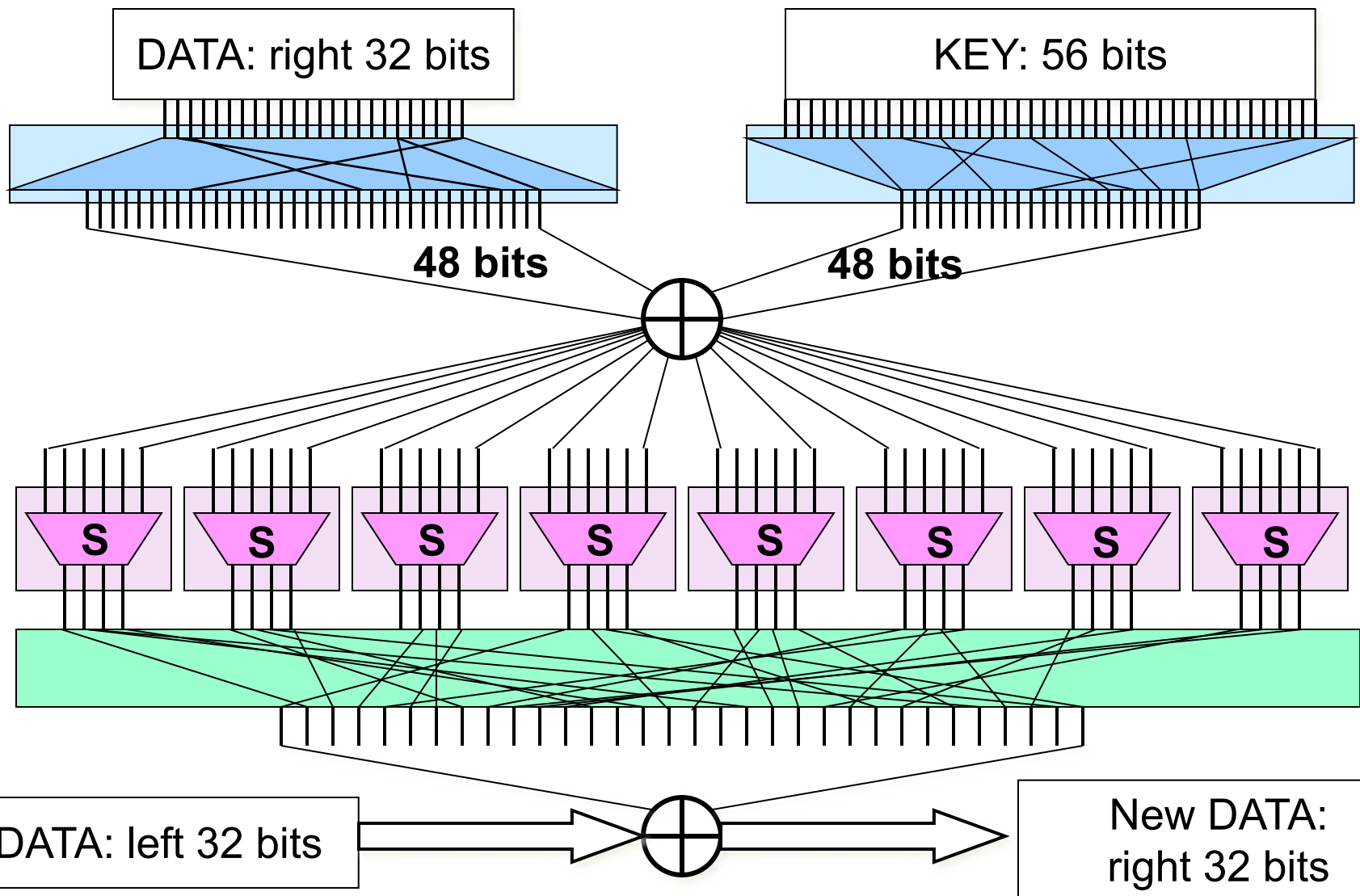
- **Plaintext block is split in two**
 - *Round* function applied to one half of the block
 - Output of the round function is XORed with other half of the block
 - Halves are swapped
- AES is not a Feistel cipher – it does not split the block



DES



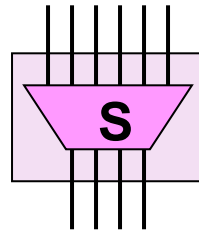
DES: f per round



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



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

- 48 bits divided into eight 6-bit sub-blocks
- Each block is operated by a separate S-box
- **S-boxes are key components of DES's security**
- Net result: 48-bit input generates 32-bit output

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
- Now you can build a DES cracker for < \$10,000

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 search between 1 and >100 billion 64-bit (e.g., RC5) keys per second

Increasing The Key

Can double encryption work for DES?

– Useless if we could find a key K such that:

$$E_K(P) = E_{K_2}(E_{K_1}(P))$$

– 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

Triple DES key lengths

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

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

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

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$

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

$$C = E_K(D_K(E_K(P))) \equiv C = E_{K_1}(P)$$

AES ... successor to DES

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.

<http://csrc.nist.gov/encryption/aes/>

AES (Advanced Encryption Standard)

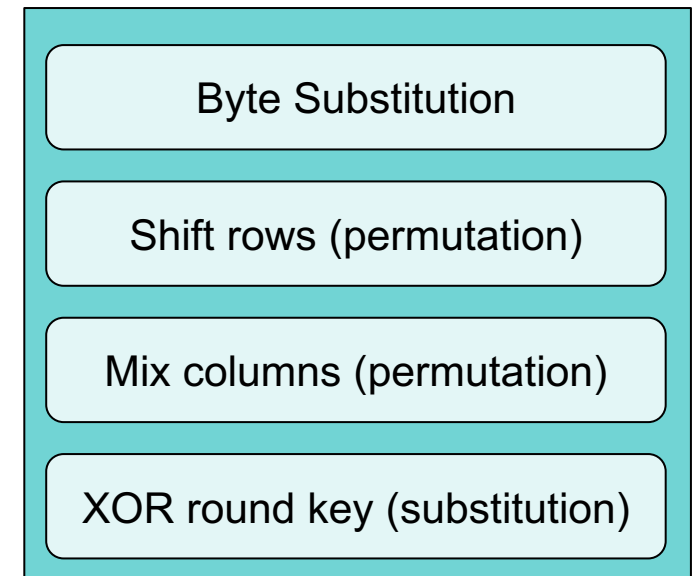
- Block cipher: 128-bit blocks
 - DES used 64-bit blocks
- Successor to DES as a standard encryption algorithm
 - DES: 56-bit key
 - AES: 128, 192, or 256 bit keys

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 did 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



AES Decryption

Same rounds
... but in reverse order

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.
- DES is relatively slow
 - It was designed with hardware encryption in mind: 3DES is 3x slower than DES
 - *Still much faster than RSA public key cryptosystems!*

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 against AES

- 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 $2^{189.7}$ (~189 bits)
 - AES-256 has computational complexity of $2^{254.9}$ (~254 bits)
- 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, 3DES
 - FIPS standard since 1976; 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
- Twofish
 - Successor to Blowfish; key length from 128, 192, 256 bits; 128-bit blocks
- IDEA
 - 128-bit keys; operates on 64-bit blocks
 - More secure than DES but faster algorithms are available

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

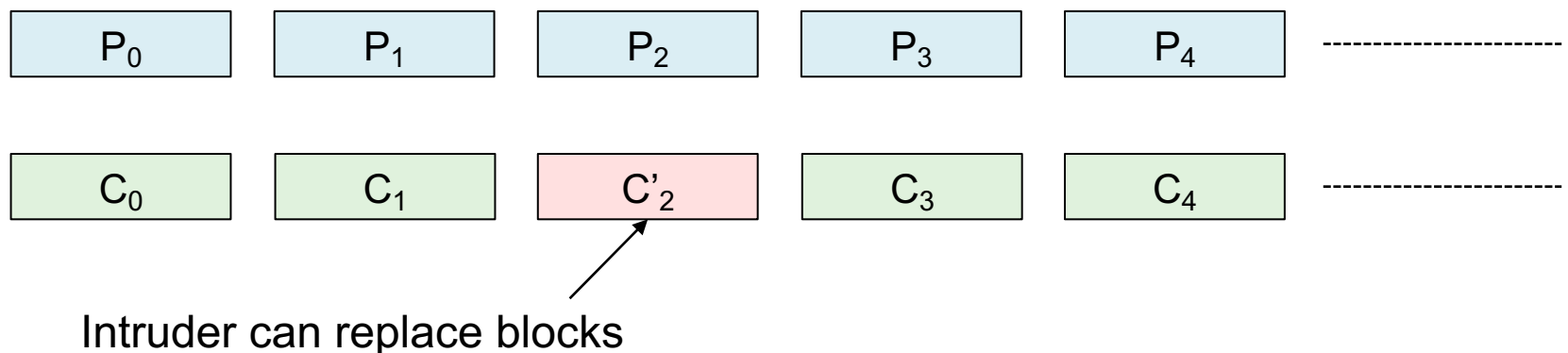
Create a predictive approximation of inputs to outputs

- 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
- 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.

Block chaining

Not a good idea to use block ciphers directly

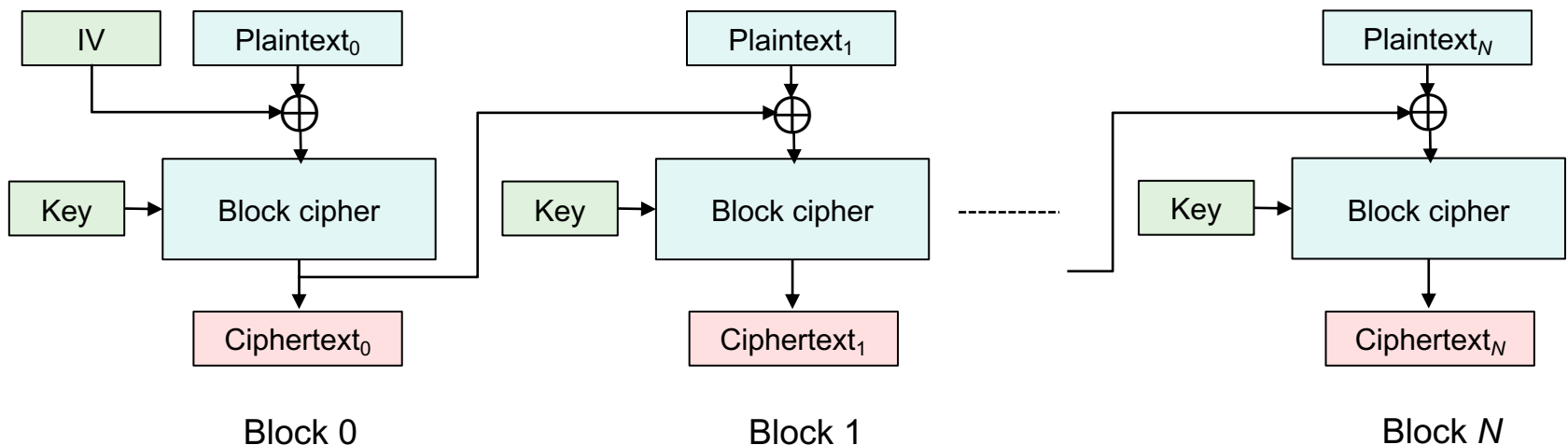
- Streams of data are broken into k -byte blocks
 - Each block encrypted separately
 - This is called **Electronic Codebook (ECB)**
- Problems
 1. Same plaintext results in identical encrypted blocks
Enemy can build up a code book of plaintext/ciphertext matches
 2. Attacker can add/delete/replace blocks



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

$$c_i = E_K(m) \oplus c_{i-1}$$

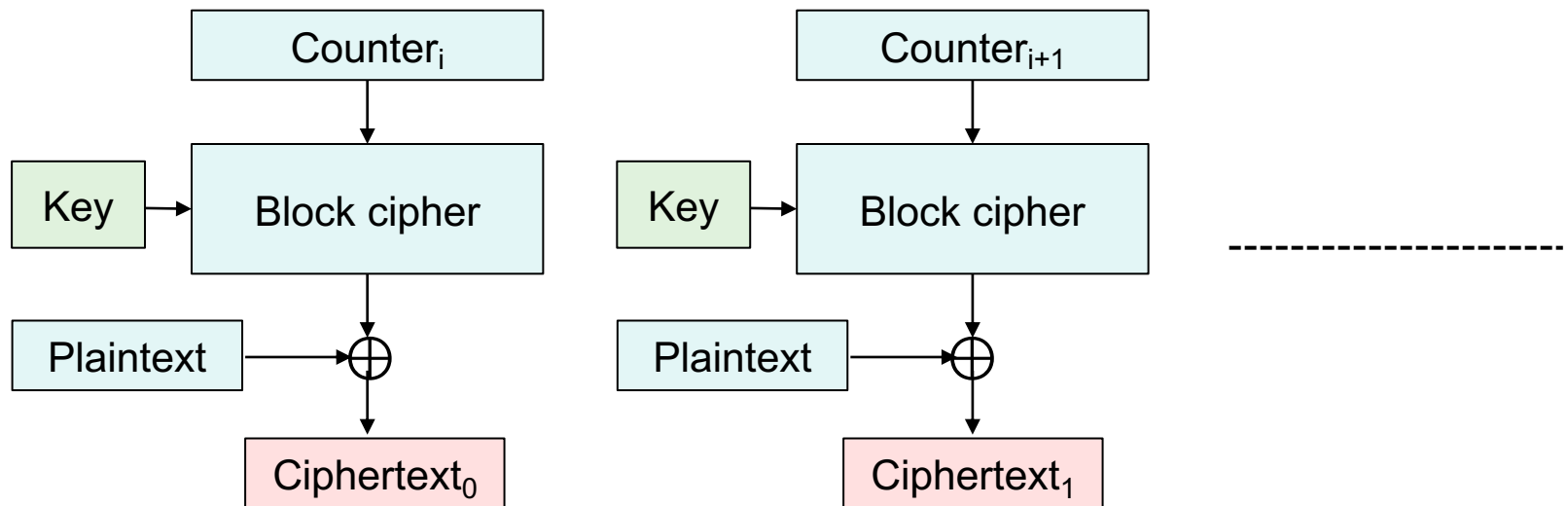


CBC Observations

- Identical plaintext does not produce the same ciphertext
- Each block is a function of all previous blocks
- An attacker can still cause data corruption

Block encryption: Counter (CTR) mode

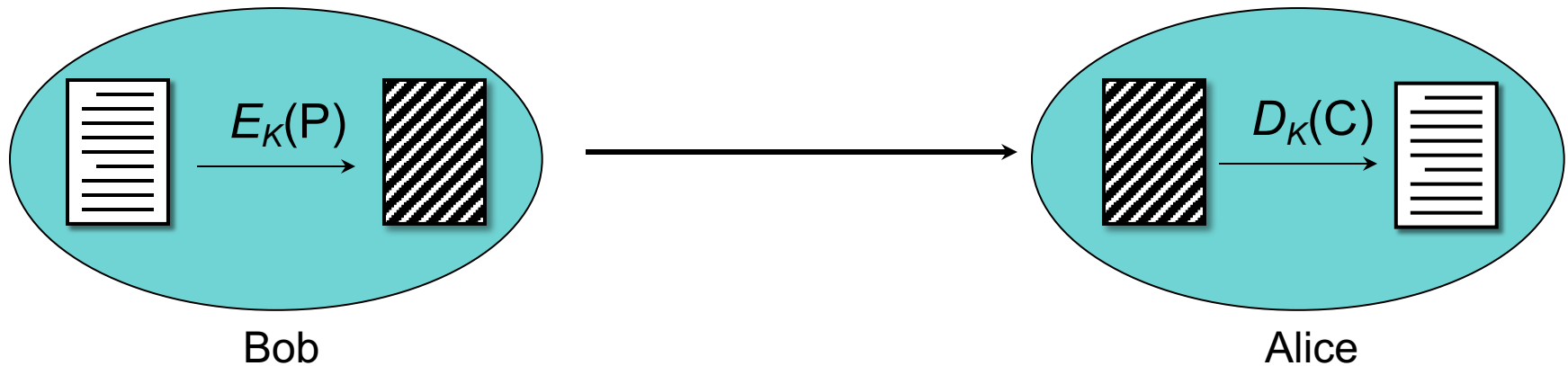
- Random starting **counter** = bunch of k random bits, just like IV
 - Any function producing a non-repeating sequence (an incrementing number is a common function)
- **Encrypt the counter with the key**
- Exclusive-or result with plaintext block



Key Distribution

Communicating with symmetric cryptography

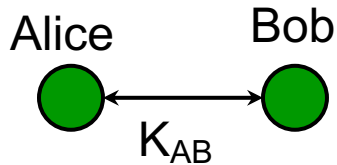
- Both parties must agree on a secret key, K
- Message is encrypted, sent, decrypted at other side



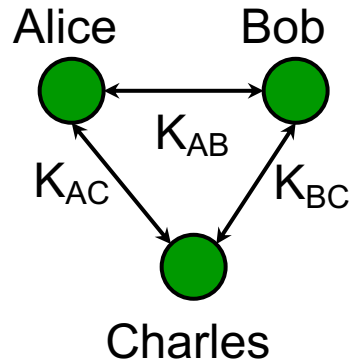
- **Key distribution must be secret**
 - otherwise messages can be decrypted
 - users can be impersonated

Key explosion

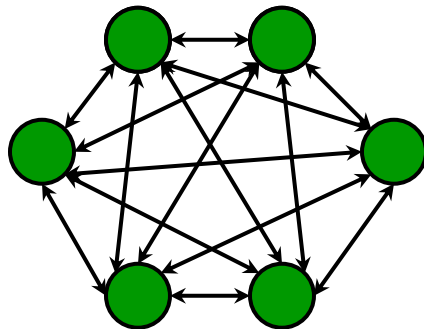
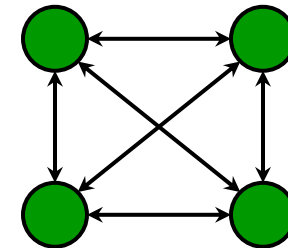
Each pair of users needs a separate key for secure communication



2 users: 1 key



3 users: 3 keys



6 users: 15 keys

100 users: 4,950 keys

1000 users: 399,500 keys

$$n \text{ users: } \frac{n(n-1)}{2} \text{ keys}$$

Key distribution

Secure key distribution is the biggest problem with symmetric cryptography

Public Key Cryptography

Public-key algorithm

- Two related keys.

$$C = E_{K_1}(P) \quad P = D_{K_2}(C)$$

$$C' = E_{K_2}(P) \quad P = D_{K_1}(C')$$

K_1 is a **public** key

K_2 is a **private** key

- Examples:

- RSA, Elliptic curve algorithms
DSS (digital signature standard),

- Key length

- Unlike symmetric cryptography, not every number is a valid key
- 3072-bit RSA = 256-bit elliptic curve = 128-bit symmetric cipher
- 15360-bit RSA = 521-bit elliptic curve = 256-bit symmetric cipher

Trapdoor functions

- Public key cryptography relies on trapdoor functions
- Trapdoor function
 - Easy to compute in one direction
 - Inverse is difficult to compute without extra information

Example:

96171919154952919 is the product of two prime #s. What are they?

If you're told that one of them is 100225441
then it's easy to compute the other: 959555959

RSA Public Key Cryptography

- Ron Rivest, Adi Shamir, Leonard Adleman created a true public key encryption algorithm in 1977
- Each user generates two keys:
 - **Private key** (kept secret)
 - **Public key** (can be shared with anyone)
- Difficulty of algorithm based on the difficulty of factoring large numbers
 - keys are functions of a pair of large (~300 digits) prime numbers

RSA algorithm

How to generate keys

- choose two random large prime numbers p, q
- Compute the product $n = pq$
- randomly choose the encryption key, e , such that:
 e and $(p - 1)(q - 1)$ are relatively prime
- Compute a decryption key, d such that:
 $ed = 1 \pmod{(p - 1)(q - 1)}$
 $d = e^{-1} \pmod{(p - 1)(q - 1)}$
- discard p, q

The security of the algorithm rests on our understanding that factoring n is extremely difficult

RSA Encryption

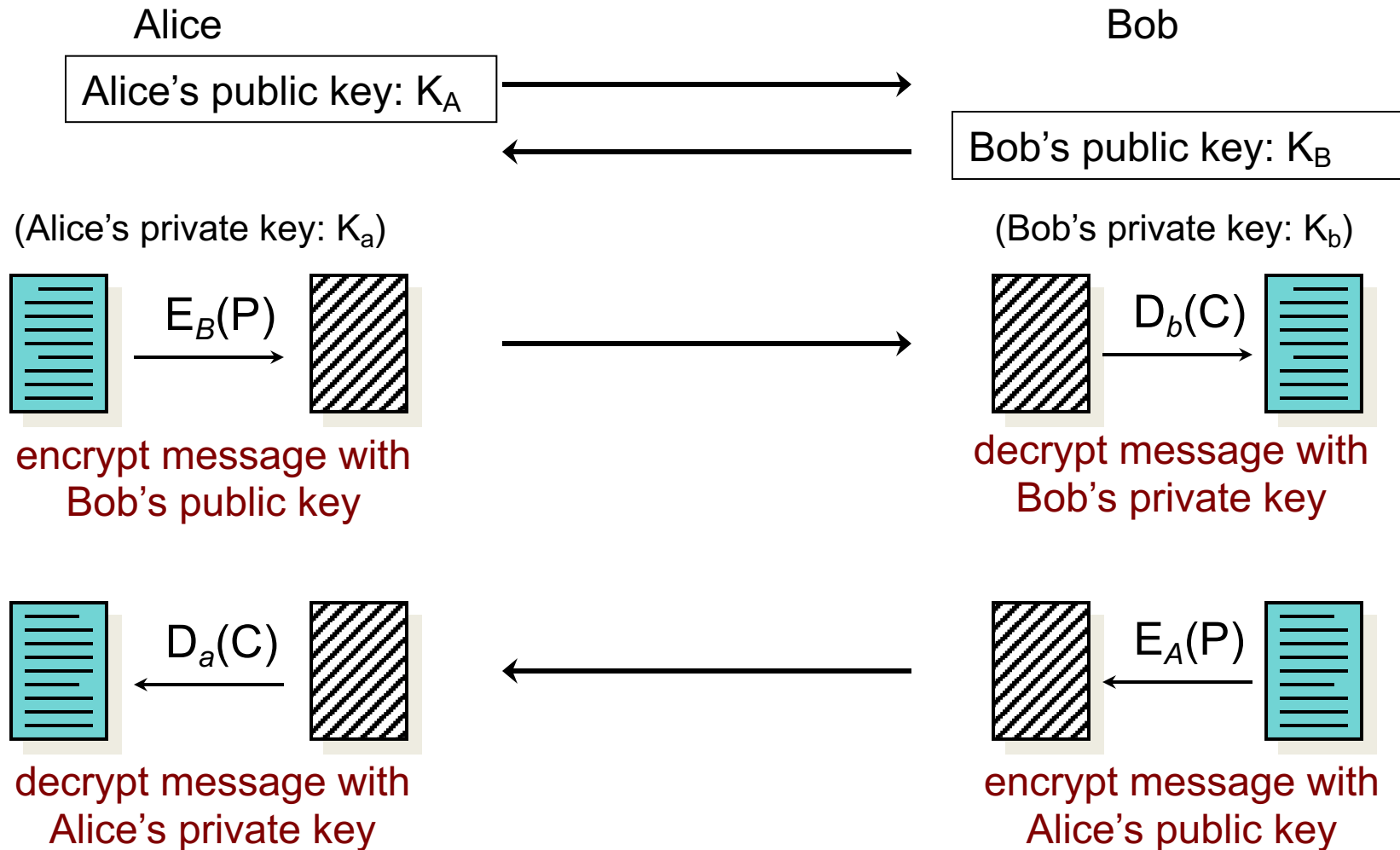
- Key pair: e, d
- Agreed-upon modulus: n
- Encrypt:
 - divide data into numerical blocks $< n$
 - encrypt each block:
$$c = m^e \bmod n$$
- Decrypt:
$$m = c^d \bmod n$$

Communication with public key algorithms

Different keys for encrypting and decrypting

- No need to worry about key distribution

Communication with public key algorithms



RSA isn't good for communication

Calculations are very expensive

Common speeds:

Algorithm	Bytes/sec
AES-128-ECB	148,000,000
AES-128-CBC	153,000,000
AES-256-ECB	114,240,000
RSA-2048 encrypt	3,800,000
RSA-2048 decrypt	96,000

- AES ~1500x faster to decrypt; 40x faster to encrypt than RSA
- RSA is also subject to mathematical attacks
 - Certain numbers may expose weaknesses
- **If anyone learns your private key, they can read all your messages**

Key Exchange

Diffie-Hellman Key Exchange

Key distribution algorithm

- Allows two parties to exchange keys securely
- Not public key encryption
- Based on difficulty of computing discrete logarithms in a finite field compared with ease of calculating exponentiation

Allows us to negotiate a secret **common key** without fear of eavesdroppers

Diffie-Hellman Key Exchange

- All arithmetic performed in a field of integers modulo some large number
- Both parties agree on
 - a **large prime number p**
 - and a **number $\alpha < p$**
- Each party generates a public/private key pair

Private key for user i : X_i

Public key for user i : $Y_i = \alpha^{X_i} \bmod p$

Diffie-Hellman exponential key exchange

- Alice has secret key X_A
- Alice sends Bob public key Y_A
- Alice computes
- Bob has secret key X_B
- Bob sends Alice public key Y_B

$$K = Y_B^{X_A} \bmod p$$

$K = (\text{Bob's public key}) (\text{Alice's private key}) \bmod p$

Diffie-Hellman exponential key exchange

- Alice has secret key X_A
- Alice sends Bob public key Y_A
- Alice computes
- Bob has secret key X_B
- Bob sends Alice public key Y_B
- Bob computes

$$K = Y_B^{X_A} \bmod p$$

$$K = Y_A^{X_B} \bmod p$$

$$***K' = (Alice's public key) (Bob's private key) \bmod p***$$

Diffie-Hellman exponential key exchange

- Alice has secret key X_A
- Alice sends Bob public key Y_A
- Alice computes

$$K = Y_B^{X_A} \bmod p$$

- expanding:

$$\begin{aligned} K &= Y_B^{X_A} \bmod p \\ &= (\alpha^{X_B} \bmod p)^{X_A} \bmod p \\ &= \alpha^{X_B X_A} \bmod p \end{aligned}$$

- Bob has secret key X_B
- Bob sends Alice public key Y_B
- Bob computes

$$K = Y_A^{X_B} \bmod p$$

- expanding:

$$\begin{aligned} K &= Y_A^{X_B} \bmod p \\ &= (\alpha^{X_A} \bmod p)^{X_B} \bmod p \\ &= \alpha^{X_A X_B} \bmod p \end{aligned}$$

$$\mathbf{K = K'}$$

K is a common key, known *only* to Bob and Alice

Hybrid Cryptosystems

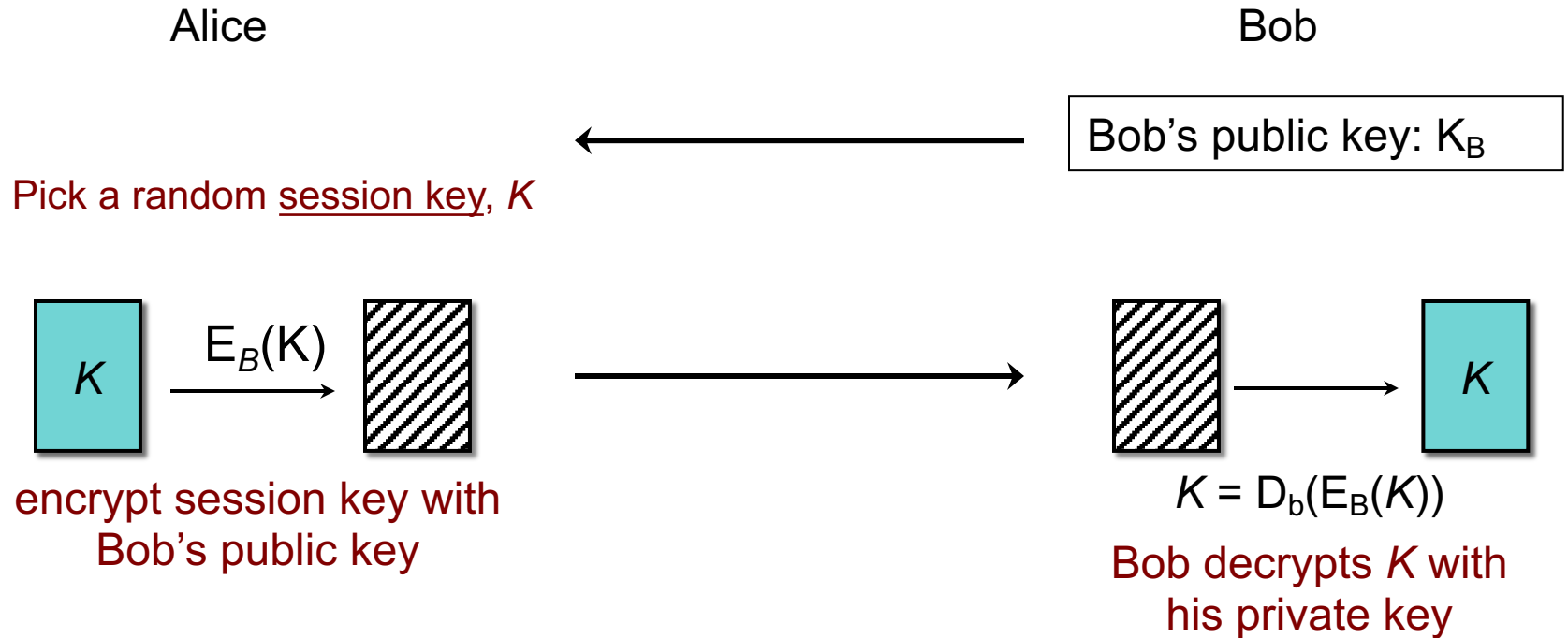
Hybrid Cryptosystems

- **Session key**: randomly-generated key for one communication session
- Use a **public key algorithm** to send the session key
- Use a **symmetric algorithm** to encrypt data with the session key

Public key algorithms are almost never used to encrypt messages

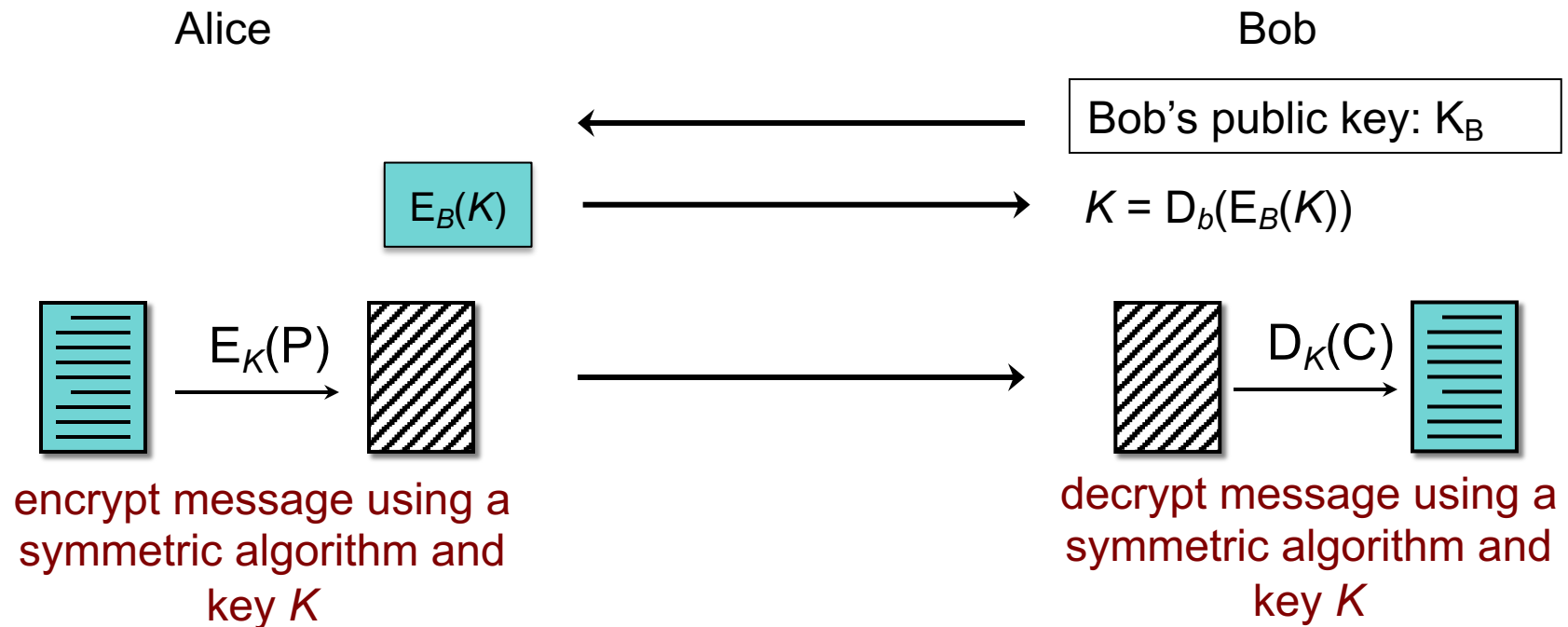
- MUCH slower; vulnerable to *chosen-plaintext attacks*
- RSA-2048 approximately 55x slower to encrypt and 2,000x slower to decrypt than AES-256

Communication with a hybrid cryptosystem

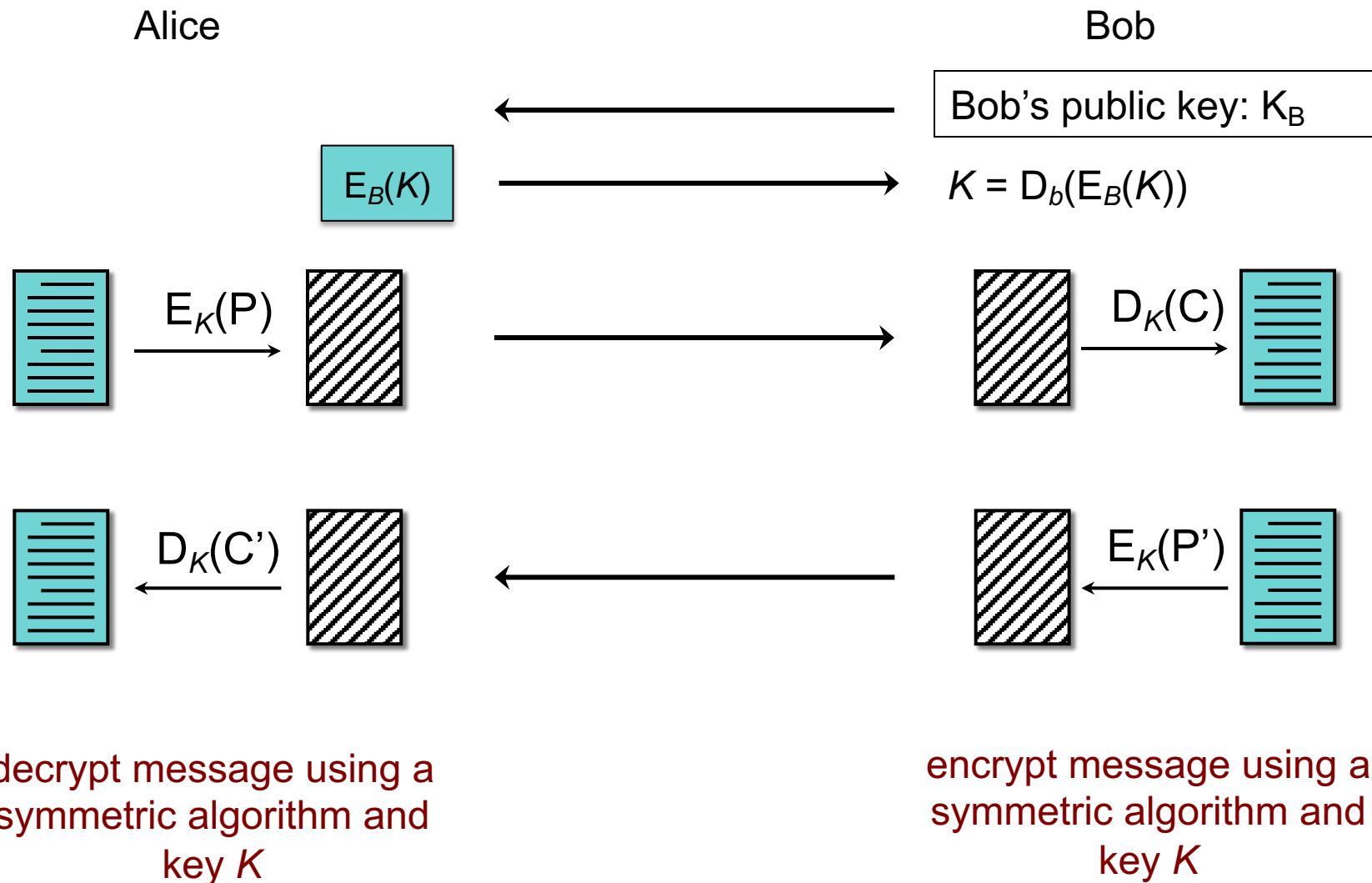


Now Bob knows the secret session key, K

Communication with a hybrid cryptosystem



Communication with a hybrid cryptosystem



Forward Secrecy

Forward Secrecy

Suppose an attacker steals Bob's private key

- Future messages can be compromised
- He can go through past messages & decrypt old session keys

Pick a session key
Encrypt it with the Bob's public key

Bob decrypts the session key

Security rests entirely on the secrecy of Bob's private key

If Bob's private key is compromised, all recorded past traffic can be decrypted

Forward Secrecy

- **Forward secrecy**
 - Compromise of long term keys does not compromise past session keys
 - There is no one secret to steal that will compromise multiple messages
- **Diffie-Hellman key exchange**
 - Generate set of keys per session
 - Use derived common key as the encryption/decryption key
 - Or as a key to encrypt a session key
 - Not recoverable as long as long as private keys are thrown away
 - Unlike RSA keys, Diffie-Hellman makes key generation simple
- **Keys must be ephemeral**
 - Client & server will generate new Diffie-Hellman parameters for each session – all will be thrown away

Diffie-Hellman is preferred over RSA for key exchange to achieve forward secrecy – generating Diffie-Hellman keys is a rapid, low-overhead process

Cryptographic systems: summary

- **Symmetric ciphers**
 - Based on “SP networks” = substitution & permutation sequences
- **Asymmetric ciphers** – public key cryptosystems
 - Based on **trapdoor** functions
Easy to compute in one direction; difficult to compute in the other direction without special information (the trapdoor)
- **Hybrid cryptosystem**
 - Pick a random session key
 - Use a public key algorithm to send
 - Use a symmetric key algorithm to encrypt traffic back & forth
- **Key exchange algorithms (more to come later)**
 - Diffie Hellman
 - Public key

Enables secure communication without knowledge of a shared secret

Looking ahead

RSA cryptography in the future

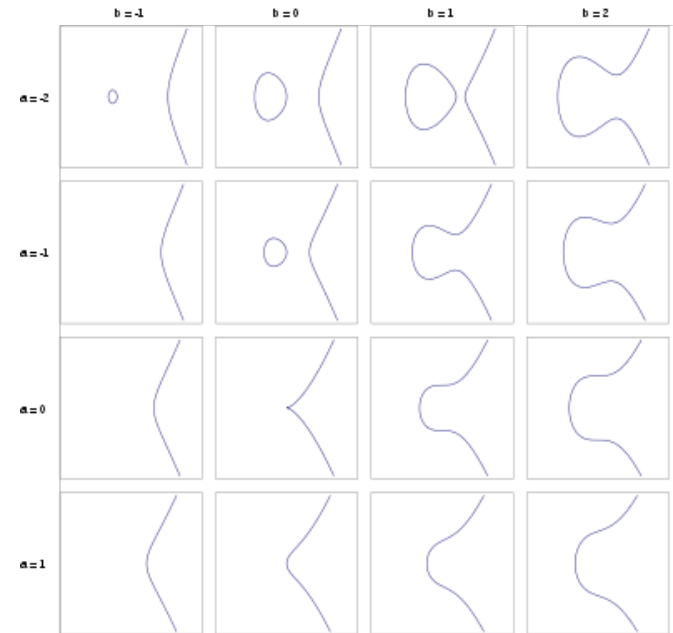
- Based on the difficulty of factoring products of two large primes
- Factoring algorithms get more efficient as numbers get larger
 - As the ability to decrypt numbers increases, the key size must therefore grow even faster
 - This is not sustainable (especially for embedded devices)

Elliptic Curve Cryptography

- Alternate approach: elliptic curves

$$y^2 = x^3 + ax + b$$

- Using discrete numbers, pick
 - A prime number as a maximum (modulus)
 - A curve equation
 - A public base point on the curve
 - A random private key
 - Public key is derived from the private key, the base point, and the curve
- To compute the private key from the public,
 - We need an elliptic curve discrete logarithm function
 - This is difficult and is the basis for ECC's security



Catalog of elliptic curves
https://en.wikipedia.org/wiki/Elliptic_curve

ECC vs. RSA

- **RSA is still the most widely used public key cryptosystem**
 - Mostly due to inertia & widespread implementations
 - Simpler implementation & faster decryption
- **ECC offers higher security with fewer bits than RSA**
 - ECC is also faster (for key generation & encryption) and uses less memory
 - NIST defines 15 standard curves for ECC
 - Many implementations support only a couple (P-256, P-384)
- **For long-term security**

The European Union Agency for Network and Information Security (ENISA) and the National Institute for Science & Technology (NIST) recommend:

 - AES: 256 bit keys
 - RSA: 15,360 bit keys
 - ECC: 512 bit keys

<https://www.keylength.com/en/4/>

<http://https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014>

Quantum Computers

Once (if) real quantum computers can be built, they can

– **Factor efficiently**

- Shor's algorithm factors numbers exponentially faster
- RSA will not be secure anymore

– **Find discrete logarithms & elliptic curve discrete logarithms efficiently**

- Diffie-Hellman key exchange & ECC will not be secure

Symmetric cryptography is largely immune to attacks

- Crack a symmetric cipher in time proportional to the square root of the key space size: $2^{n/2}$ vs. 2^n
 - Use 256-bit AES to be safe

2016: NSA called for a migration to “post-quantum cryptographic algorithms” – but no agreement yet on what those will be

2019: Narrowed submissions down to 26 leading candidates

Quantum Computers

- Quantum computing is not faster at everything
 - There are only four types of algorithms currently known where quantum computing offers an advantage
- Researchers are developing algorithms that are based on problems quantum computers do not help with
- 2017: IBM presented CRYSTALS (Cryptographic Suite for Algebraic Lattices) – category of equations called *lattice problems*
 - Example: add 3 out of a set of 5 numbers
 - Give the sum to a friend and ask them to determine which numbers were added
 - Try this if someone picks 500 out of 1,000 numbers with 1,000 digits each
- 2019: IBM demonstrated the use of this algorithm in a tape drive

<https://www.scientificamerican.com/article/new-encryption-system-protects-data-from-quantum-computers/>

The End