Crypto

- **Cryptology** — The art and science of making and breaking "secret codes"
- **Cryptography**— making "secret codes"
- **Cryptanalysis**— breaking "secret codes"
- **Crypto**— all of the above (and more)

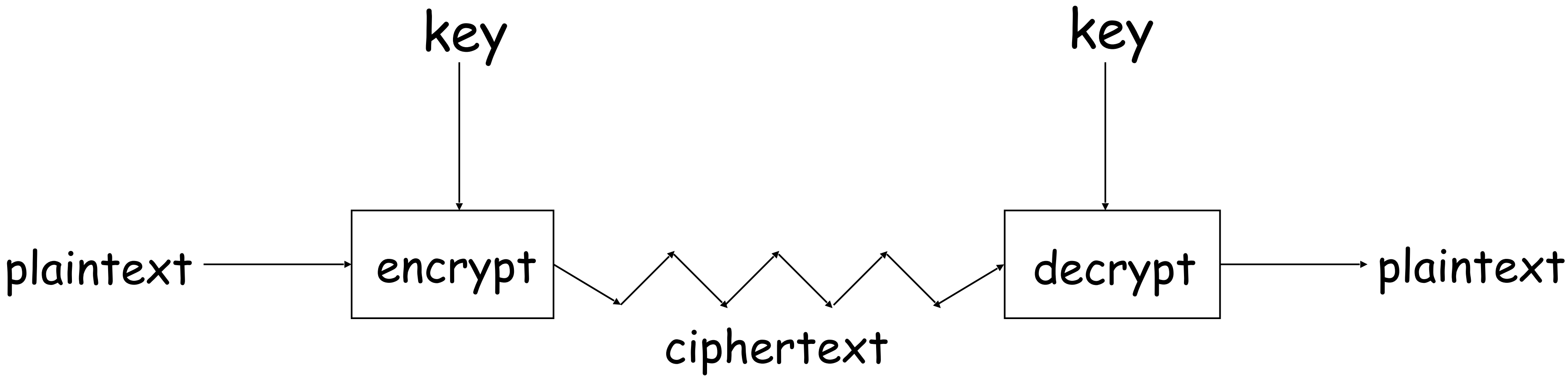
How to Speak Crypto

- A cipher or cryptosystem is used to encrypt the plaintext
- The result of encryption is ciphertext
- We decrypt ciphertext to recover plaintext
- A keyis used to configure a cryptosystem
- A symmetric key cryptosystem uses the same key to encrypt as to decrypt
- A public key cryptosystem uses a public key to encrypt and a private key to decrypt

Kerckhoffs' Principle

- Basic assumptions
  - The system is completely known to the attacker
  - Only the key is secret
  - That is, crypto algorithms are not secret
- This is known as **Kerckhoffs' Principle**
- Why do we make such an assumption?
  - Experience has shown that secret algorithms tend to be weak when exposed [Security by Obscurity]
    - Content Scrambling System (CSS) for DVD, reversed engineered… and dead.

Crypto as Black Box



A generic view of symmetric key crypto

Simple Substitution

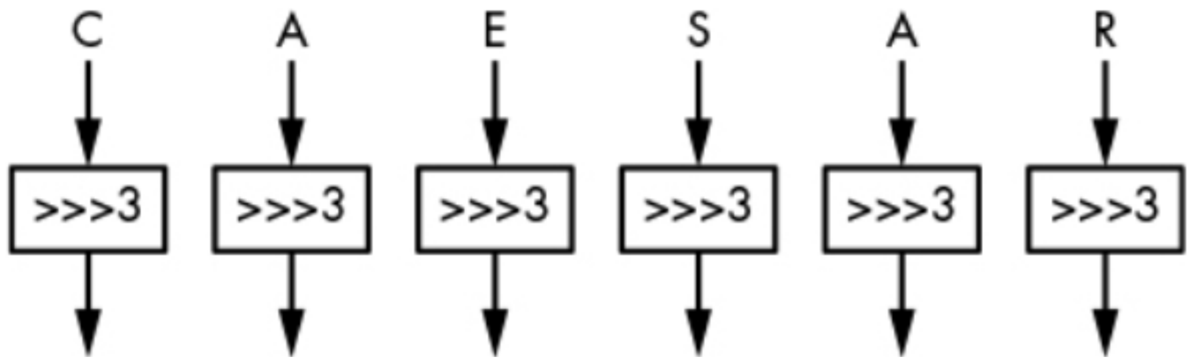
[Shift Cipher]

- Plaintext: fourscoreandsevenyearsago
- Key:

Plaintext	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
Ciphertext	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

- Ciphertext:

  IRXUVFRUHDQGVHYHQBHDUVDJR
- Shift by 3 is "Caesar's cipher"


C
A
E
S
A
R
>>>3
>>>3
>>>3
>>>3
>>>3
>>>3

Ceasar's Cipher Decryption

- Suppose we know a Caesar's cipher is being used:

Plaintext	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
Ciphertext	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

- Given ciphertext:
VSRQJHEREVTXDUHSDQWV
- Plaintext: spongebobsquarepants

Simple Substitution: General Case

- In general, simple substitution key can be any **permutation** of letters
  - Not necessarily a shift of the alphabet
- For example

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

- Then key space $26! > 2^{88}$ possible keys

Cryptanalysis I: Try Them All

- A simple substitution (shift by n) is used
  - But the key is unknown
- Given ciphertext: CSYEVIXIVQMREXIH
- How to find the key?
- **Key space**: only 26 possible keys
- **Exhaustive key search**
- Solution: key is $n = 4$

Exhaustive key search

- If we have 26 possible keys…
  - Trying all of them -> gotcha!
    - This means you have been unlucky though…
  - But… if you are lucky
    - 1 attempt -> gotcha!

- You need to consider the average
  - That is, you need, **on average**, to try only <u>half of the whole key space</u>
    - If you have 26 possible keys… 13 attempts! (on average!)
    - If you have 2^{64} possible keys… 2^{63} (on average!)

Exhaustive key search

- A brute-force attack can produce false positives
  - A subset of keys that are able to decrypt the ciphertext… but are not the target key!
  - The likelihood to find a “wrong key” is related to the size of the key space and the length of the plaintext

Cryptanalysis II: Be Clever

- We know that a simple substitution is used
- But not necessarily a shift by n
- Find the key given the ciphertext:

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFX
QWAXBVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPB
FXFQVXGTVJVWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPB
FTIXPFHXZHVFAGFOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTO
GHFQPBQWAQJJTODXQHFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIP
BQWKFABVYYDZBOTHPBQPQJTQOTOGHFQAPBFEQJHDXXQVAVXEB
QPEFZBVFOJIWFFACFCCFHQWAUVWFLQHGFXVAFXQHFUFHILTTAV
WAFFAWTEVOITDHFHFQAITIXPFHXAFQHEFZQWGFLVWPTOFFA

Cryptanalysis II

- Cannot try all 2^{88} simple substitution keys
- Can we be more clever?
- English letter frequency counts…

Cryptanalysis II

- Ciphertext:

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQW
AXBVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXG
TVJVWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHV
FAGFOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJT
ODXQHFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHP
BQPQJTQOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQ
WAUVWFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPF
HXAFQHEFZQWGFLVWPTOFFA

- Analyze this message using statistics below

Ciphertext frequency counts:

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
21	26	6	10	12	51	10	25	10	9	3	10	0	1	15	28	42	0	0	27	4	24	22	28	6	8

It works because Simple Substitution doesn't hide the statistical properties of the Plaintext

Cryptanalysis: Terminology

- Cryptosystem is **secure** if best known attack is to try all keys
  - **Computationally secure** if this takes too much time, i.e., decades
- Cryptosystem is **insecure** if **any** shortcut attack is known

NOTE:

- Insecure cipher might be harder to break than a secure cipher!!

One-Time Pad: Encryption

e=000 h=001 i=010 k=011 l=100 r=101 s=110 t=111

Encryption: Plaintext ⊕ Key = Ciphertext

	h	e	i	l	h	i	t	l	e	r
Plaintext:	001	000	010	100	001	010	111	100	000	101
Key:	111	101	110	101	111	100	000	101	110	000
Ciphertext:	110	101	100	001	110	110	111	001	110	101
	s	r	l	h	s	s	t	h	s	r

One-Time Pad: Decryption

e=000	h=001	i=010	k=011	l=100	r=101	s=110	t=111

Decryption: Ciphertext $\oplus$ Key = Plaintext

	s	r	l	h	s	s	t	h	s	r
Ciphertext:	110	101	100	001	110	110	111	001	110	101
Key:	111	101	110	101	111	100	000	101	110	000
Plaintext:	001	000	010	100	001	010	111	100	000	101
	h	e	i	l	h	i	t	l	e	r

One-Time Pad

Double agent claims following "**key**" was used:

	s	r	l	h	s	s	t	h	s	r
Ciphertext:	110	101	100	001	110	110	111	001	110	101
"**key**":	101	111	000	101	111	100	000	101	110	000
"Plaintext":	011	010	100	100	001	010	111	100	000	101
	k	i	l	l	h	i	t	l	e	r

e=000 h=001 i=010 k=011 l=100 r=101 s=110 t=111

One-Time Pad

Or claims the key is...

	s	r	l	h	s	s	t	h	s	r
Ciphertext:	110	101	100	001	110	110	111	001	110	101
"key":	111	101	000	011	101	110	001	011	101	101
"Plaintext":	001	000	100	010	011	000	110	010	011	000
	h	e	l	i	k	e	s	i	k	e

e=000 h=001 i=010 k=011 l=100 r=101 s=110 t=111

Likely this is a false positive...

One-Time Pad Summary

- OTP is **Unconditionally secure**
  - Ciphertext provides **no** info about plaintext
  - All plaintexts are equally likely
- BUT, only when used correctly
  - Pad must be **random**, **used only once**
  - Pad is **known only to sender and receiver**
- Note: pad (key) is same size as message
- So, why not distribute msg instead of pad?

One-Time Pad Summary

- **Not so common in real life though…**
  - E-mail encryption, mobile phones, web browsers, etc, don't use OTP…

- In fact, it needs:
  - TRNG (True Random Number Generator)
  - Alice has to send the PAD to Bob securely before hand.
  - Pad cannot be re-used

Codebook Cipher

- Literally, a book filled with “codewords”
- Zimmerman Telegram encrypted via codebook

Februar	13605
fest	13732
finanzielle	13850
folgender	13918
Frieden	17142
Friedenschluss	17149

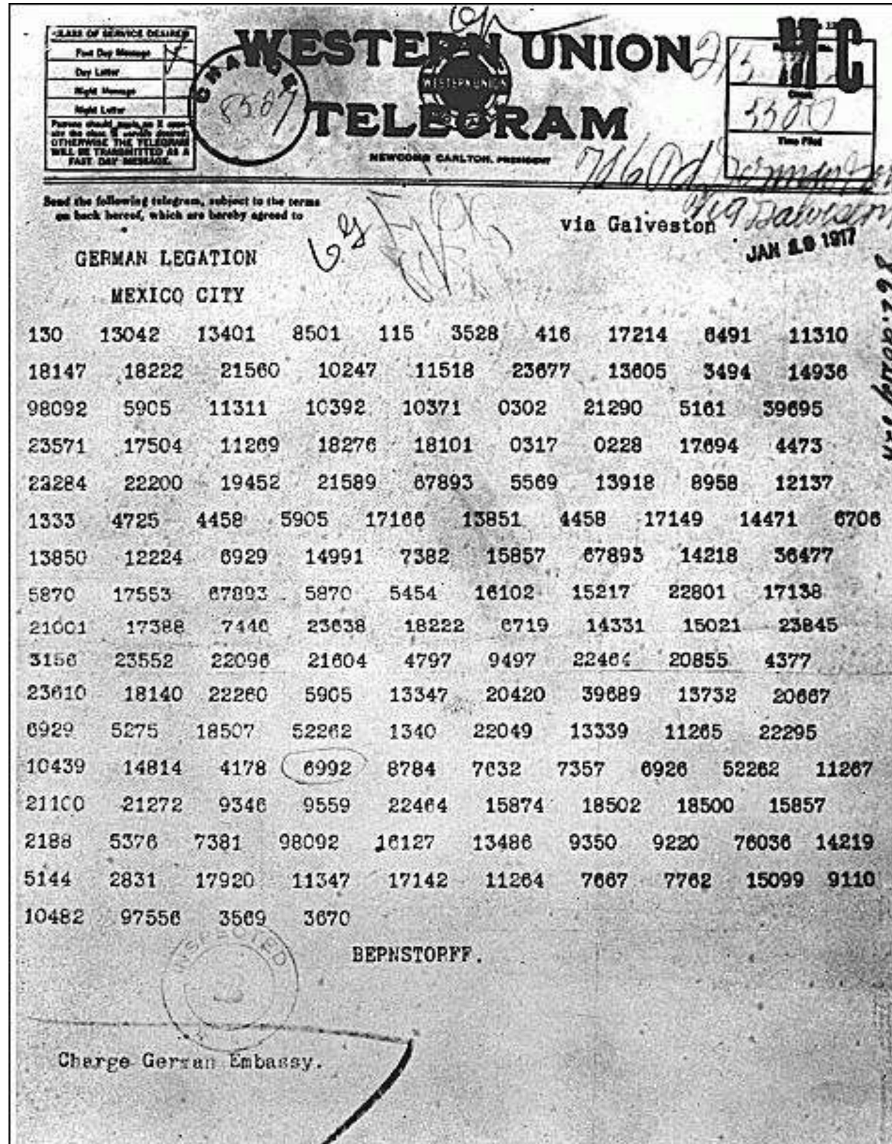
- Modern block ciphers are codebooks!
- More about this later…

Codebook Cipher: Additive

- Codebooks also (usually) use **additive**
- Additive — book of "random" numbers
  - Encrypt message with codebook
  - Then choose position in additive book
  - Add in additive to get ciphertext
  - Send ciphertext and additive position (MI)
  - Recipient subtracts additives before decrypting
- Why use an additive sequence?

Zimmerman Telegram

- Perhaps most famous codebook ciphertext ever
- A major factor in U.S. entry into World War I



WESTERN UNION TELEGRAM

NEWCOMB CARLTON, PRESIDENT

Send the following telegram, subject to the terms on back hereof, which are hereby agreed to

via Galveston

JAN 19 1917

GERMAN LEGATION

MEXICO CITY

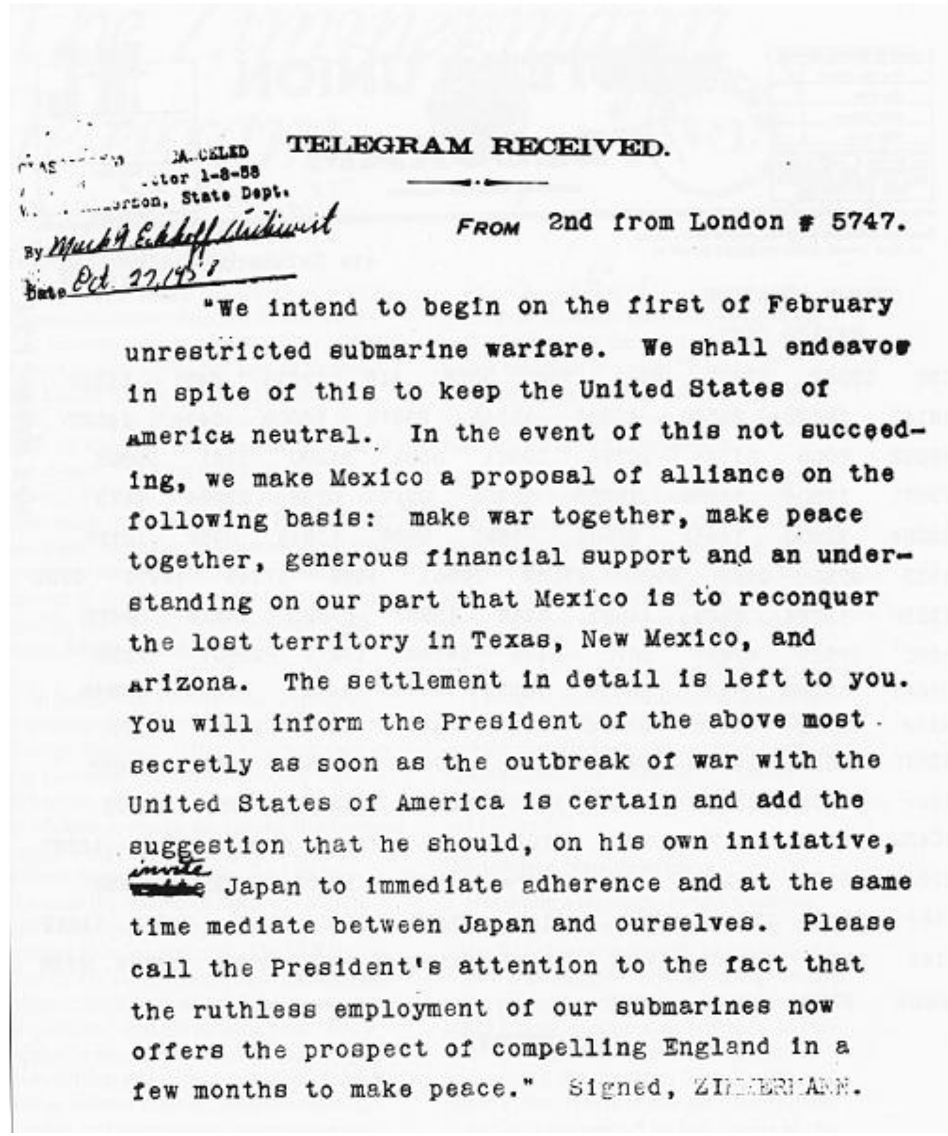
130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0317 0228 17694 4473
23284 22200 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13850 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
21001 17388 7446 23638 18222 6719 14331 15021 23845
3156 23552 22096 21604 4797 9497 22464 20855 4377
23610 18140 22260 5905 13347 20420 39689 13732 20667
6929 5275 18507 52262 1340 22049 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11267
21100 21272 9346 9559 22464 15874 18502 18500 15857
2188 5376 7381 98092 16127 13486 9350 9220 76036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 3670

BERNSTORFF.

Charge German Embassy.

Zimmerman Telegram Decrypted

- British had recovered partial codebook
- Then able to fill in missing parts



TELEGRAM RECEIVED.

FROM 2nd from London # 5747.

"We intend to begin on the first of February unrestricted submarine warfare. We shall endeavor in spite of this to keep the United States of America neutral. In the event of this not succeeding, we make Mexico a proposal of alliance on the following basis: make war together, make peace together, generous financial support and an understanding on our part that Mexico is to reconquer the lost territory in Texas, New Mexico, and Arizona. The settlement in detail is left to you. You will inform the President of the above most secretly as soon as the outbreak of war with the United States of America is certain and add the suggestion that he should, on his own initiative, invite Japan to immediate adherence and at the same time mediate between Japan and ourselves. Please call the President's attention to the fact that the ruthless employment of our submarines now offers the prospect of compelling England in a few months to make peace." Signed, ZIMMERMANN.

Claude Shannon

- The father of Information Theory
- 1949 paper: *Comm. Thy. of Secrecy Systems*
- Fundamental concepts:
    - **Confusion** — obscure relationship between plaintext/key and ciphertext [**substitution**]
    - **Diffusion** — spread plaintext statistics through the ciphertext [**bit permutation**]
- Proved one-time pad is secure

Claude Shannon

- “Only Diffusion” or “Only Confusion” are insecure
    - Ex: Shift Ciphers (Confusion only)
    - Ex: Transposition Ciphers (Diffusion only)
- Concatenation of Diffusion and Confusion (any order) is Secure
- Called **Product Ciphers**
    - Block Ciphers are Product Ciphers
        - 1 changed bit in input -> on average half bits change in output [Excellent Diffusion property]

Possible attacks

- Exhaustive Key Search (or Brute Force)
- Letter Frequency Analysis
- Implementation Attacks [Side-channel Attacks]
  - What if we measured the **electrical power consumption** of the CPU while operating on the key?
  - What about the **electromagnetic radiation**?
    - $2 magnetic probe near an **iPhone 4** when the phone was performing cryptographic operations
  - What about measuring the **run time**?
- Social Engineering Attacks
  - Attacking the weakest link… possibly you

Taxonomy of Cryptography

- **Symmetric Key**
  - Same key for encryption and decryption
  - Modern types: Stream ciphers, Block ciphers
- **Public Key** (or "asymmetric" crypto)
  - Two keys, one for encryption (public), and one for decryption (private)
  - And digital signatures
- **Hash algorithms**
  - Can be viewed as "one way" crypto
- **Hybrid Schemes**
  - Combinations of the above techniques

Taxonomy of Cryptanalysis

- From perspective of info available to Trudy…
  - Ciphertext only — Trudy's worst case scenario
  - Known plaintext
  - Chosen plaintext
    - "Lunchtime attack"
  - Adaptively chosen plaintext
  - Related key
  - Forward search (public key crypto)
  - And others…

Symmetric Key Crypto

- Stream cipher — generalize one-time pad
  - Except that key is relatively short
  - Key is stretched into a long **keystream**
  - Keystream is used just like a one-time pad
  - AKA Vernam Ciphers
- Block cipher — generalized codebook
  - Block cipher key determines a codebook
  - Each key yields a different codebook
  - Employs both "confusion" and "diffusion"

Stream Ciphers

- Very simplified example:
- Key: 011011

Key →  → Keystream

- Keystream: 1101001011001011....

- Encryption: $y_i = x_i + s_i \ mod\ 2$
  - y_i = ciphertext bit
  - x_i = plaintext bit
  - s_i = keystream bit

$$y_i \equiv x_i \oplus s_i$$

XOR

0	0	=	0
0	1	=	1
1	0	=	1
1	1	=	0

Stream Ciphers

- Encrypting bits individually
  - Bit from the Keystream -> bit in the Plaintext

- Synchronous Stream Ciphers
  - Keystream depends on the key only

- Asynchronous Stream Ciphers
  - Keystream depends on the key and the Ciphertext
  - We'll not see these ones

- Security relies completely on the Keystream
  - It tries to look random… but still deterministic

Stream Ciphers

- Why XOR though?

XOR

0	0	=	0
0	1	=	1
1	0	=	1
1	1	=	0

50% -> 0
50% -> 1

To guess a 4-bit key you need to toss a coin 4 times:

$$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2}$$

If the key is 56-bit long:

$$\frac{1}{2^{56}}$$

Remember: brute-force in 2^{55} attempts <u>in average</u>

Stream Ciphers

- Why XOR though?

Besides, it's invertible!

XOR

0	0	=	0
0	1	=	1
1	0	=	1
1	1	=	0

0	0	=	0
1	1	=	0
0	1	=	1
1	0	=	1

Start from 1, XOR 1 with 1 and you get 0

To get back, start with the result 0, XOR 0 with 1 and you get back to 1

Stream Ciphers

Once upon a time, not so very long ago… stream ciphers were the king of crypto

- Today, not as popular as block ciphers

We'll discuss two stream ciphers:

- A5/1
  - Based on shift registers
  - Used in GSM mobile phone system
- RC4
  - Based on a changing lookup table
  - Used many places

A5/1: Shift Registers

- A5/1 uses 3 shift registers:
  - **X**: 19 bits $(x_0, x_1, x_2, \ldots, x_{18})$
  - **Y**: 22 bits $(y_0, y_1, y_2, \ldots, y_{21})$
  - **Z**: 23 bits $(z_0, z_1, z_2, \ldots, z_{22})$

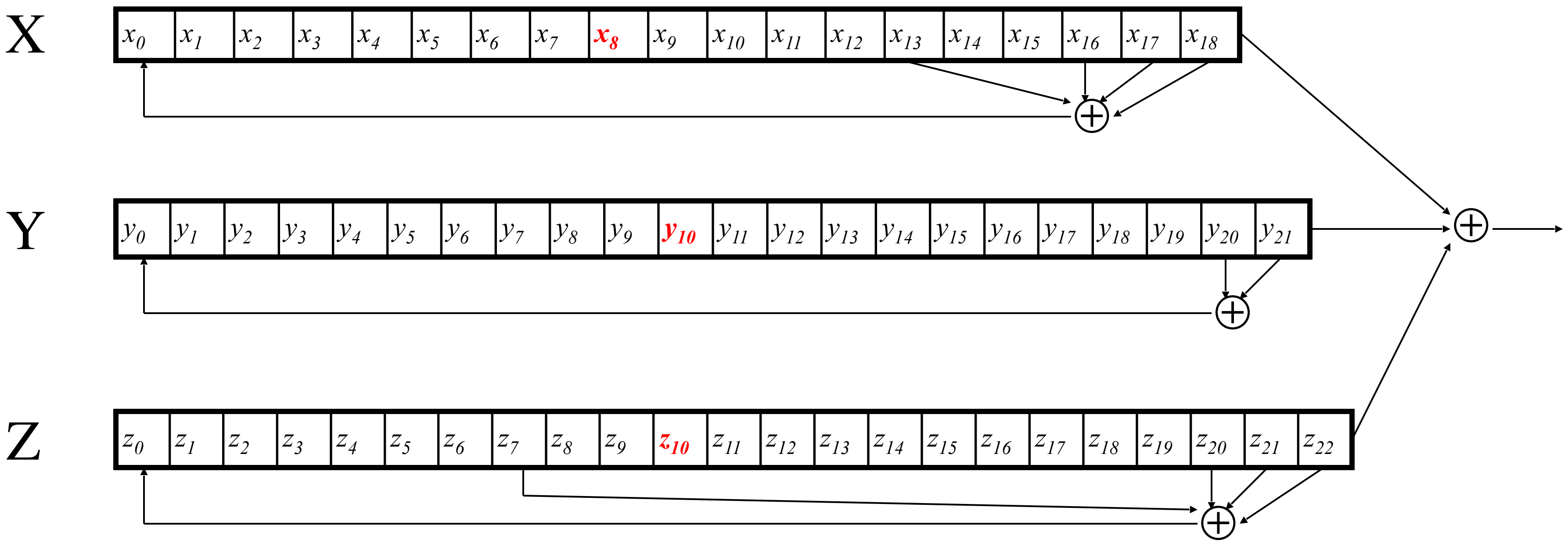
A5/1: Keystream

- At each iteration: $m = \text{maj}(x_8, y_{10}, z_{10})$
  - Examples: $\text{maj}(0,1,0) = 0$ and $\text{maj}(1,1,0) = 1$
- If $x_8 = m$ then X steps
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - $x_i = x_{i-1}$ for $i = 18,17,\ldots,1$ and $x_0 = t$
- If $y_{10} = m$ then Y steps
  - $t = y_{20} \oplus y_{21}$
  - $y_i = y_{i-1}$ for $i = 21,20,\ldots,1$ and $y_0 = t$
- If $z_{10} = m$ then Z steps
  - $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
  - $z_i = z_{i-1}$ for $i = 22,21,\ldots,1$ and $z_0 = t$
- Keystream **bit** is $x_{18} \oplus y_{21} \oplus z_{22}$

When register steps:

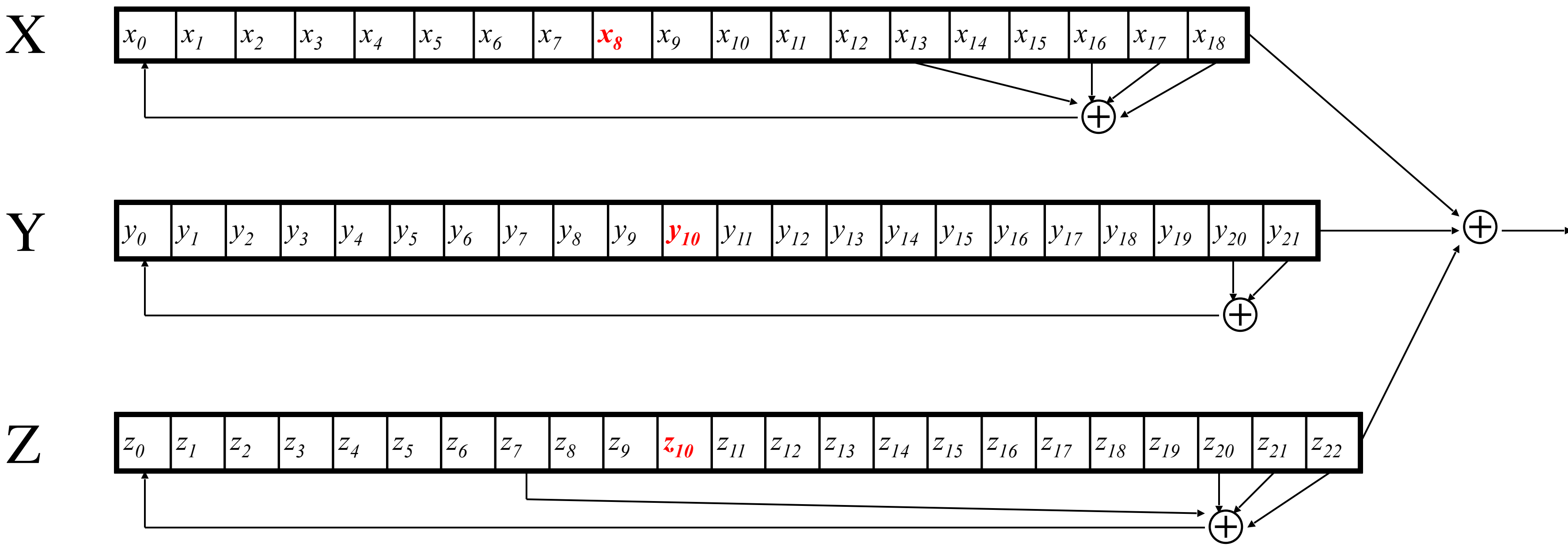
1. Computes new first bit
2. THEN, shifts

A5/1



- Each variable here is a single bit
- Key is used as **initial fill** of registers
- Each register steps (or not) based on $\text{maj}(x_8, y_{10}, z_{10})$
- Keystream bit is XOR of rightmost bits of registers

A5/1

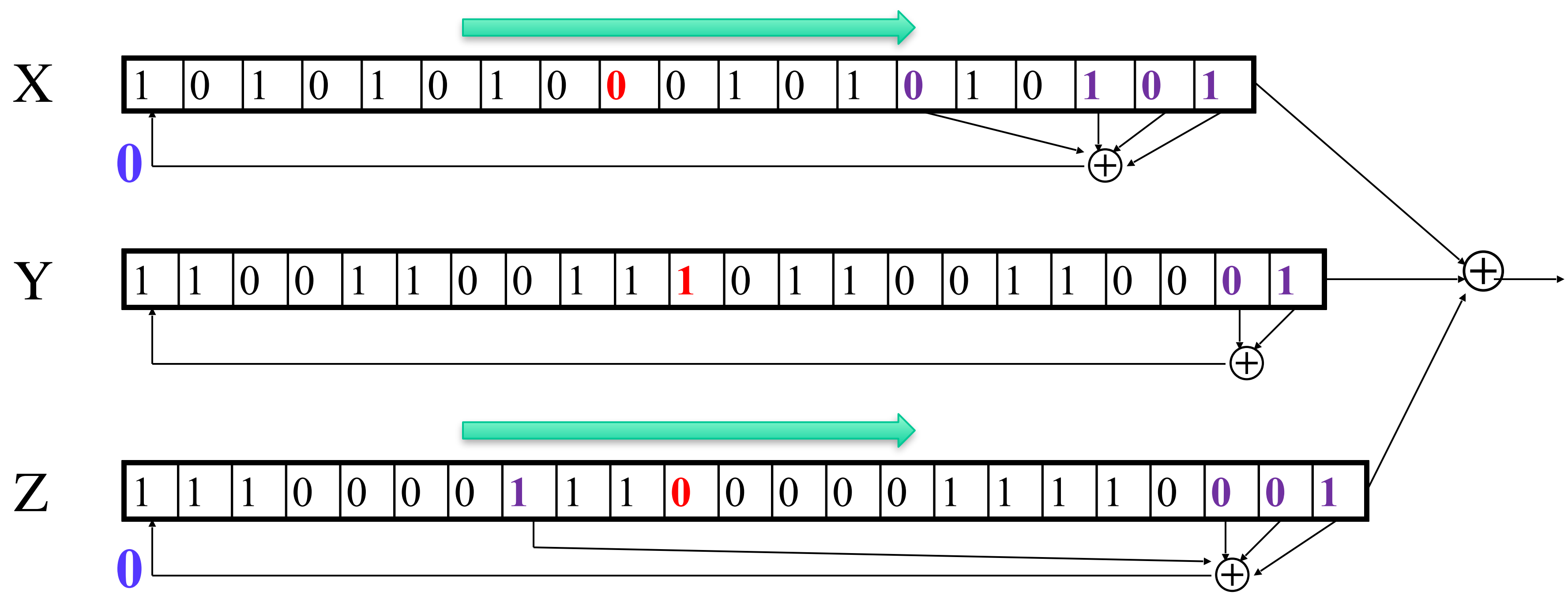


- Example - Key (64bits):

1010101000101010101110011001110110011000111100001110000011110001

  - 1010101000101010101 (first 19 bits) -> X
  - 1100110011101100110001 (middle 22 bits) -> Y
  - 11100001110000011110001 (last 23 bits) -> Z

A5/1

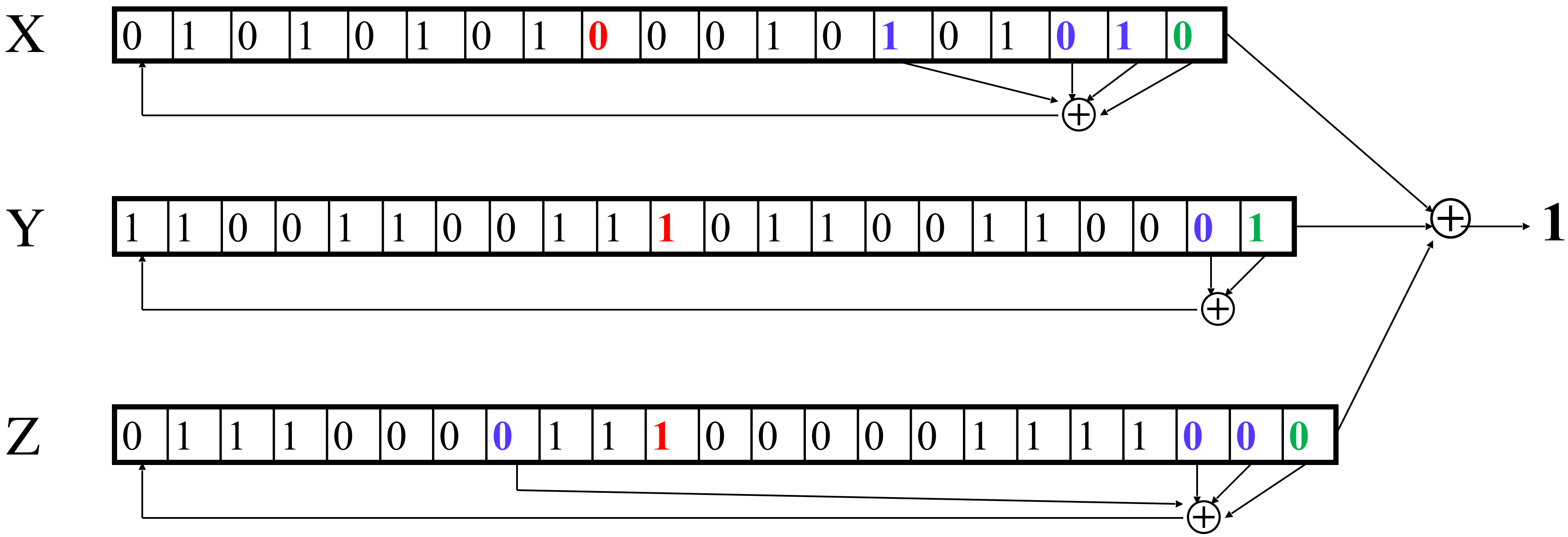


1. Majority vote: $m = \text{maj}(x_8, y_{10}, z_{10}) = \text{maj}(0,1,0) = 0$
2. Compute new first bits:

   X: $0 \oplus 1 \oplus 0 \oplus 1 = 0$

   Z: $1 \oplus 0 \oplus 0 \oplus 1 = 0$
3. Shift!

A5/1



1. Majority vote: $m = \text{maj}(x_8, y_{10}, z_{10}) = \text{maj}(0,1,0) = 0$
2. Compute new first bits:

   X: $0 \oplus 1 \oplus 0 \oplus 1 = 0$

   Z: $1 \oplus 0 \oplus 0 \oplus 1 = 0$
3. Shift!

Shift Register Crypto

- Shift register crypto efficient in hardware
- Often, slow if implemented in software
- In the past, very, very popular
- Today, more is done in software due to fast processors
- Shift register crypto still used some
  - Especially in resource-constrained devices

Stream Ciphers

- Stream ciphers were popular in the past
  - Efficient in hardware
  - Speed was needed to keep up with voice, etc.
  - Today, processors are fast, so software-based crypto is usually more than fast enough
- Future of stream ciphers?
  - Shamir declared "the death of stream ciphers"
  - May be greatly exaggerated…

Block Ciphers

- Used more often than Stream Ciphers
- Stream Ciphers are small and fast, good with little computational resources
  - Ex, old cell phones
- Stream Ciphers were considered more efficient
  - Fewer gates (hardware efficient)
  - Fewer clock cycles (software efficient)
- But modern Block Ciphers are similarly efficient

(Iterated) Block Cipher

- Plaintext and ciphertext consist of fixed-sized blocks
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of **key** and **output** of previous round
- Usually implemented in software

Feistel Cipher: Encryption

- **Feistel cipher** is a type of block cipher
  - **Not** a specific block cipher
- Split plaintext block into left and right halves: $P = (L_0, R_0)$
- For each round $i = 1, 2, \ldots, n$, compute:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

where F is **round function** and K_i is **subkey**

- Ciphertext: $C = (L_n, R_n)$

Feistel Cipher: Decryption

- Start with ciphertext $C = (L_n, R_n)$
- For each round $i = n, n-1, \ldots, 1$, compute

  $R_{i-1} = L_i$

  $L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$

where F is round functionand K_i is subkey

- Plaintext: $P = (L_0, R_0)$
- Decryption works for any function F
  - But only secure for certain functions F

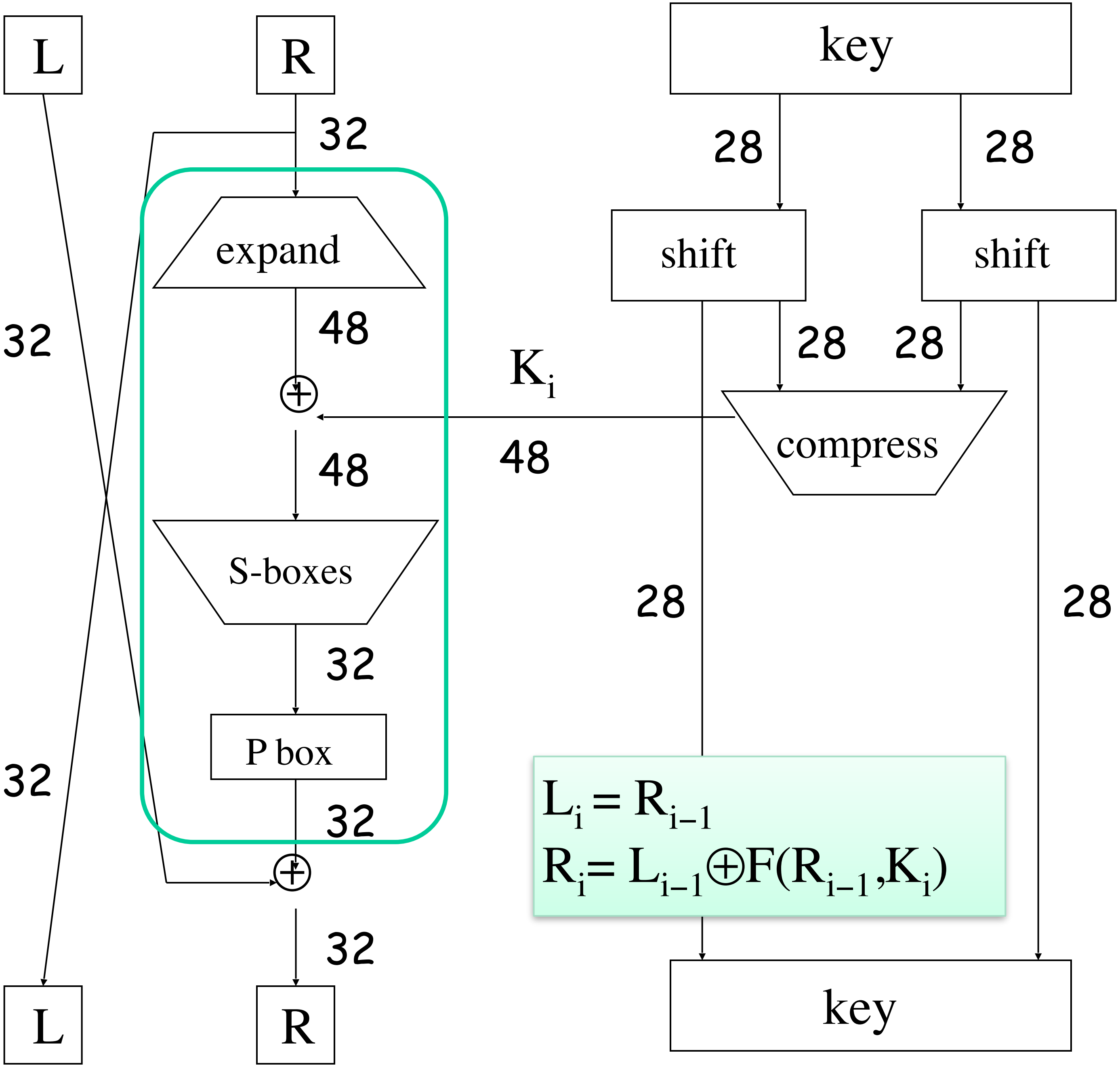
Data Encryption Standard

- **DES** developed in 1970's
- Based on IBM's Lucifer cipher
- DES was U.S. government standard
- DES was controversial
  - NSA secretly involved
  - Design process was secret
  - Key length reduced from 128 to 56 bits
  - Subtle changes to Lucifer algorithm

DES Numerology

- DES is a Feistel cipher with…
  - 64 bit block length
  - 56 bit key length
  - 16 rounds
  - 48 bits of key used each round
    - Different subkey per each round
- Round function is simple (for block cipher)
- Security depends heavily on “S-boxes”
  - Each S-box maps 6 bits to 4 bits

One Round of DES



$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$

DES Expansion Permutation

- Input 32 bits

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

- Output 48 bits

31	0	1	2	3	4	3	4	5	6	7	8
7	8	9	10	11	12	11	12	13	14	15	16
15	16	17	18	19	20	19	20	21	22	23	24
23	24	25	26	27	28	27	28	29	30	31	0

DES Expansion Permutation

- Input 32 bits

0	1	2	3	4	5	6	7	8	9	10	**11**	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

- Output 48 bits

31	0	1	2	3	4	3	4	5	6	7	8
7	8	9	10	**11**	12	**11**	12	13	14	15	16
15	16	17	18	19	20	19	20	21	22	23	24
23	24	25	26	27	28	27	28	29	30	31	0

DES S-box

- 8 “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- Here is S-box number 1

input bits (0,5)

↓

input bits (1,2,3,4)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

DES S-box

- Input: 001100

input bits (0,5)

↓ input bits (1,2,3,4)

	0000	0001	0010	0011	0100	0101	**0110**	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	**1011**	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

DES P-box

- Input 32 bits

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

- Output 32 bits

15	6	19	20	28	11	27	16	0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8	18	12	29	5	21	10	3	24

DES Last Word (Almost)

- An initial permutation before round 1
- Halves are swapped after last round
- A final permutation (inverse of initial perm) applied to (R_{16}, L_{16})
- None of this serves any security purpose
  - Possibly, arranging Ciphertext, Plaintext and bits for 8-bit data busses

DES Last notes

- Key schedule is fast to implement in hardware
- Decryption function is the same as the Encryption function ,except the key schedule that is reversed
- Larger S-boxes would have been cryptographically better
  - Eight 4-by-6 tables were the max size which could be fit in an integrated circuit (1974)

Security of DES

- Security depends heavily on S-boxes
  - Everything else in DES is linear
- 35+ years of intense analysis has revealed no back door
- Attacks, essentially exhaustive key search
- **Inescapable conclusions**
  - Designers of DES knew what they were doing
  - Designers of DES were way ahead of their time (designed to withstand differential cryptoanalysis, an attack not known until 1990)

Security of DES

- To succeed, a Differential Cryptanalysis attack needs:
  - 2^{47} chosen plaintext-ciphertext pairs

- Actually, in 1993 another attack was found: Linear Cryptanalysis
  - 2^{43} chosen plaintext-ciphertext pairs needed

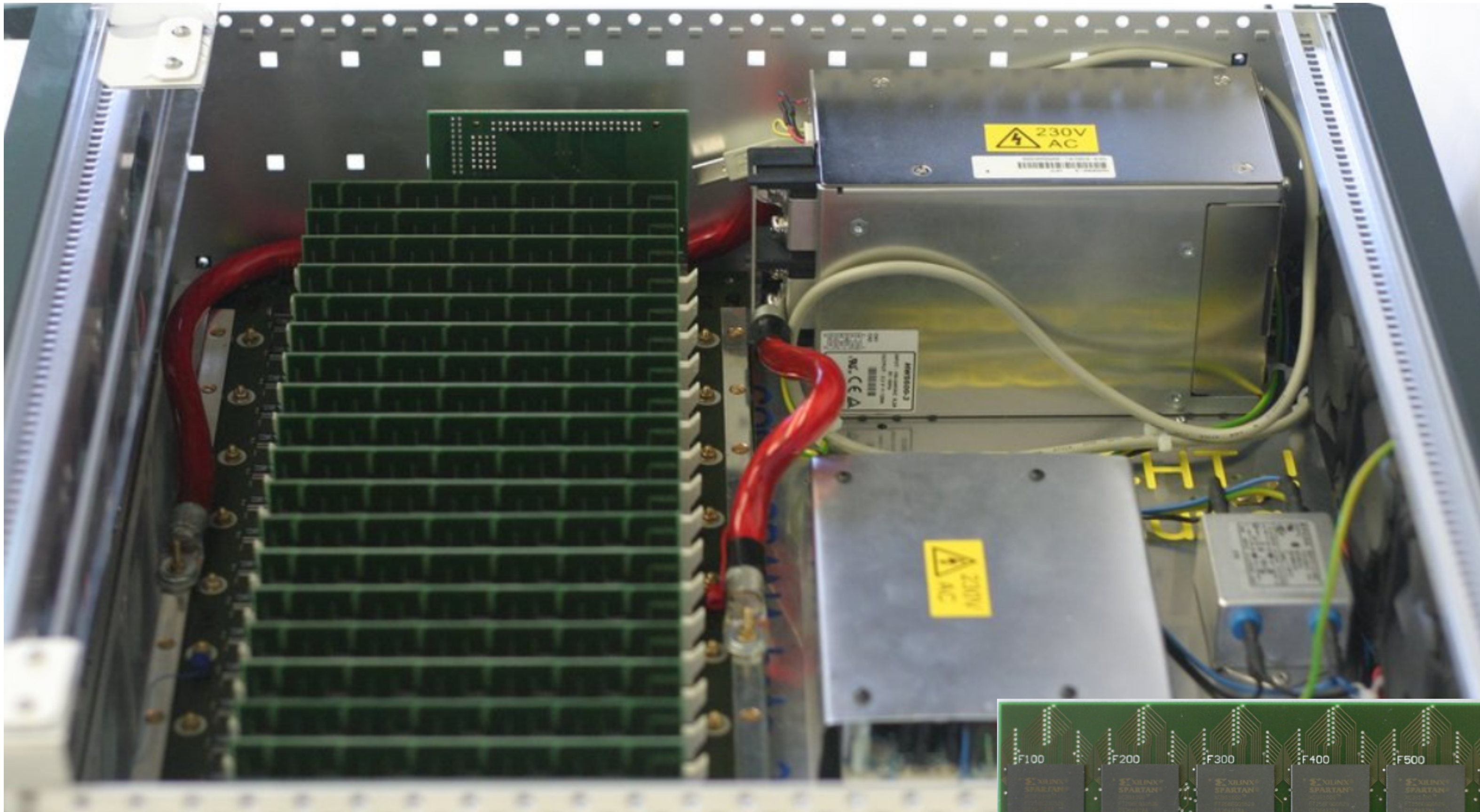
- Both unlikely in real-world systems

Security of DES

- Brute-force attack is still a problem!
- 2^{55} keys to test. A lot in 1974……or not?
  - A brute-force cracker would have broken the cipher in a matter of days
  - Cost: maybe around $20,000,000
- Bottom line: enough funds -> sure cracking

- COPACOBANA (Cost-Optimized Parallel Code-Breaker) (2006)
  - Cracks DES in 7 days for $10,000

COPACOBANA



Security of DES

- Distributed Brute-force attack
  - 1999, several nodes helped crack DES in 22hours [DES Challenge III]
- Hmmm… a malware could gently asks your PC to help in cracking ciphers somewhere…

How to strengthen a cipher

- Double and Triple Encryption
- Key Whitening

Block Cipher Notation

- P = plaintext block
- C =ciphertext block
- Encrypt P with key K to get ciphertext C
  - $C = E(P, K)$
- Decrypt C with key K to get plaintext P
  - $P = D(C, K)$
- Note: $P = D(E(P, K), K)$ and $C = E(D(C, K), K)$
  - But $P \neq D(E(P, K_1), K_2)$ and $C \neq E(D(C, K_1), K_2)$ when $K_1 \neq K_2$

Double Encryption

- Let's assume we "double-encrypted" a plaintext using DES twice with two different keys: K_1 and K_2
- $C = E(E(P, K_1), K_2)$
  - We obtain a key space: $2^{k_1} * 2^{k_2} = 2^{2k}$
- Except that an attack called Meet-in-the-Middle is possible…
  - Even though Double DES encrypts the data with two different 56-bit keys, it can be broken with 2^{57} encryption and decryption operations

Meet-in-the-Middle

- Why not $C = E(E(P,K),K)$ instead?
  - Trick question — still just 56 bit key
- Why not $C = E(E(P,K_1),K_2)$ anyway?
- A (semi-practical) **known plaintext** attack (MITM):
  - Pre-compute table of $E(P,K_1)$ for every possible key K_1 (resulting table has 2^{56} entries)
  - Then for each possible K_2 compute $D(C,K_2)$ until a match in table is found
  - When match is found, have $E(P,K_1) = D(C,K_2)$
  - Result gives us keys: $C = E(E(P,K_1),K_2)$

Triple DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- **Triple DES** or **3DES** (168-bit key)
  - $C = E(D(E(P,K_1),K_2),K_3)$
  - $P = D(E(D(C,K_1),K_2),K_3)$

Why Encrypt-Decrypt-Encrypt?

- If $K_1 = K_2 = K_3 = K$, then "classic" DES
  - Backward compatible: $E(D(E(P,K),K),K) = E(P,K)$
  - Legacy reasons

- 2 keys version (112 bit key)
  - $C = E(D(E(P,\mathbf{K_1}),K_2),\mathbf{K_1})$
  - $P = D(E(D(C,\mathbf{K_1}),K_2),\mathbf{K_1})$
- **Key Whitening** (184-bit key)
  - Two additional 64-bit keys (K_1 and K_2) are XORed prior and after DES
  - $$C = (E\left(\left(P \oplus K_1\right), K\right) \oplus \mathbf{K_2})$$