# Simple Substitution

- 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"

# 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

# Not-so-Simple Substitution

- Shift by $n$ for some $n \in \{0,1,2,\ldots,25\}$
- Then key is $n$
- Example: key $n = 7$

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 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 |

# 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?
- Only 26 possible keys — try them all!
- Exhaustive key search
- Solution: key is $n = 4$

# 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 $26! > 2^{88}$ possible keys

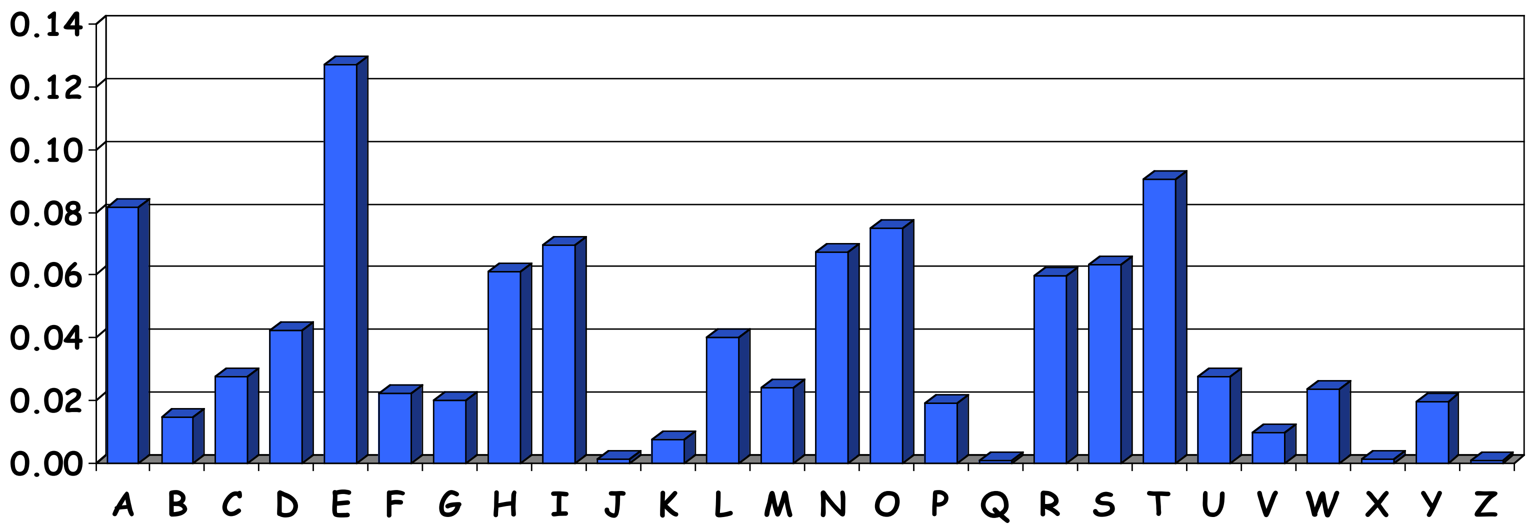
# Cryptanalysis II: Be Clever

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

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOX
BTFXQWAXBVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQ
WAEBIPBFXFQVXGTVJVWLBTPQWAEBFPBFHCVLXBQUFEVWLXGD
PEQVPQGVPPBFTIXPFHXZHVFAGFOTHFEFBQUFTDHZBQPOTHXTY
FTODXQHFTDPTOGHFQPBQWAQJJTODXQHFOQPWTBDHHIXQV
APBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHPBQPQJTQOTOGHF
QAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUVWF
LQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFH
XAFQHEFZQWGFLVWPTOFFA

# Cryptanalysis II

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



# Cryptanalysis II

- Ciphertext:

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQ
WAXBVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQ
VXGTVJVWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFH
XZHVFAGFOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQW
AQJJTODXQHFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYY
DZBOTHPBQPQJTQOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFF
ACFCCFHQWAUVWFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFH
FQAITIXPFHXAFQHEFZQWGFLVWPTOFFA

- 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 |

# Cryptanalysis: Terminology

- Cryptosystem is **secure** if best know attack is to try all keys
  - Exhaustive key search, that is
- Cryptosystem is **insecure** if *any* shortcut attack is known
- But then insecure cipher might be harder to break than a secure cipher!
  - What the … ?

# Double Transposition

- Plaintext: attackxatxdawn

| | col 1 | col 2 | col 3 |
|---|---|---|---|
| row 1 | a | t | t |
| row 2 | a | c | k |
| row 3 | x | a | t |
| row 4 | x | d | a |
| row 5 | w | n | x |

Permute rows and columns

$\Rightarrow$

| | col 1 | col 3 | col 2 |
|---|---|---|---|
| row 3 | x | t | a |
| row 5 | w | x | n |
| row 1 | a | t | t |
| row 4 | x | a | d |
| row 2 | a | k | c |

- Ciphertext: xtawxnattxadakc
- Key is matrix size and permutations: (3,5,1,4,2) and (1,3,2)

# One-Time Pad: Encryption

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

**Encryption:** Plaintext $\oplus$ 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 ⊕ 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

# One-Time Pad Summary

- Provably secure
  - Ciphertext gives no useful info about plaintext
  - All plaintexts are *equally likely*
- BUT, only when be 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?

# 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 additives to get ciphertext
  - Send ciphertext and additive position (MI)
  - Recipient subtracts additives before decrypting
- Why use an additive sequence?

# Chapter 3: Symmetric Key Crypto

The chief forms of beauty are order and symmetry…
— Aristotle

# 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
- Block cipher — generalized codebook
  - Block cipher key determines a codebook
  - Each key yields a different codebook
  - Employs both “confusion” and “diffusion”

# Stream Ciphers

# 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

# One-Time Pad: Encryption

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

**Encryption:** Plaintext $\oplus$ 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 ⊕ 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 |

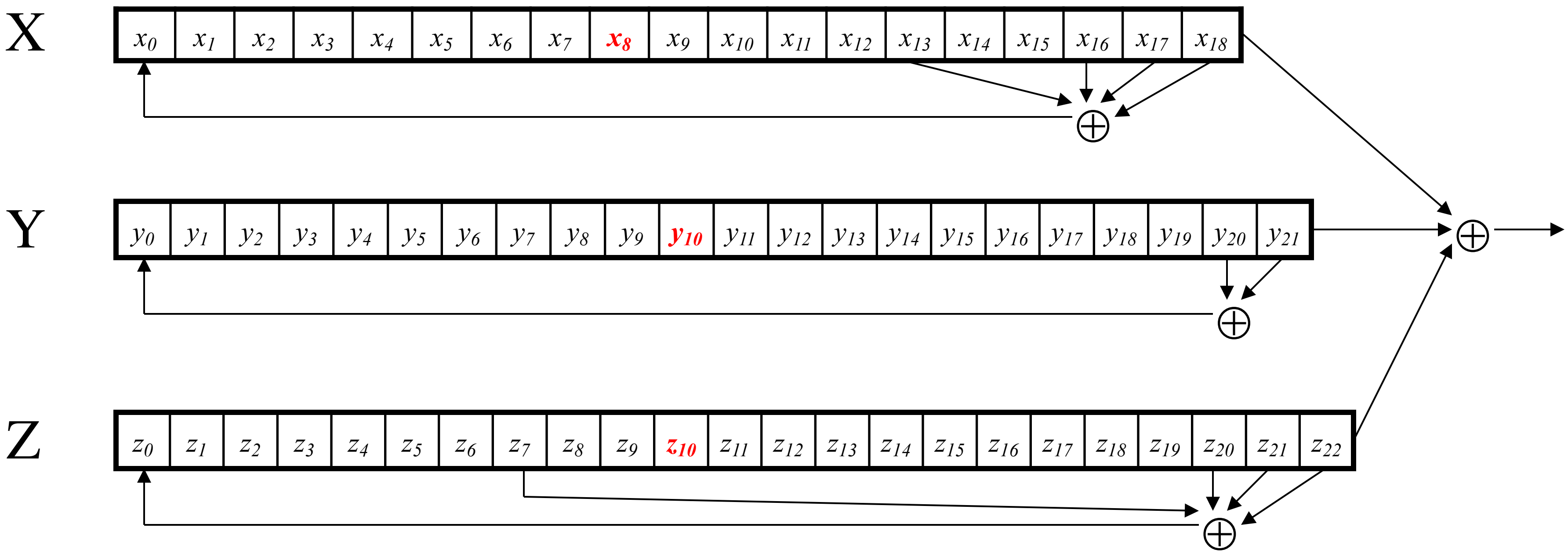
# A5/1: Shift Registers

- 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

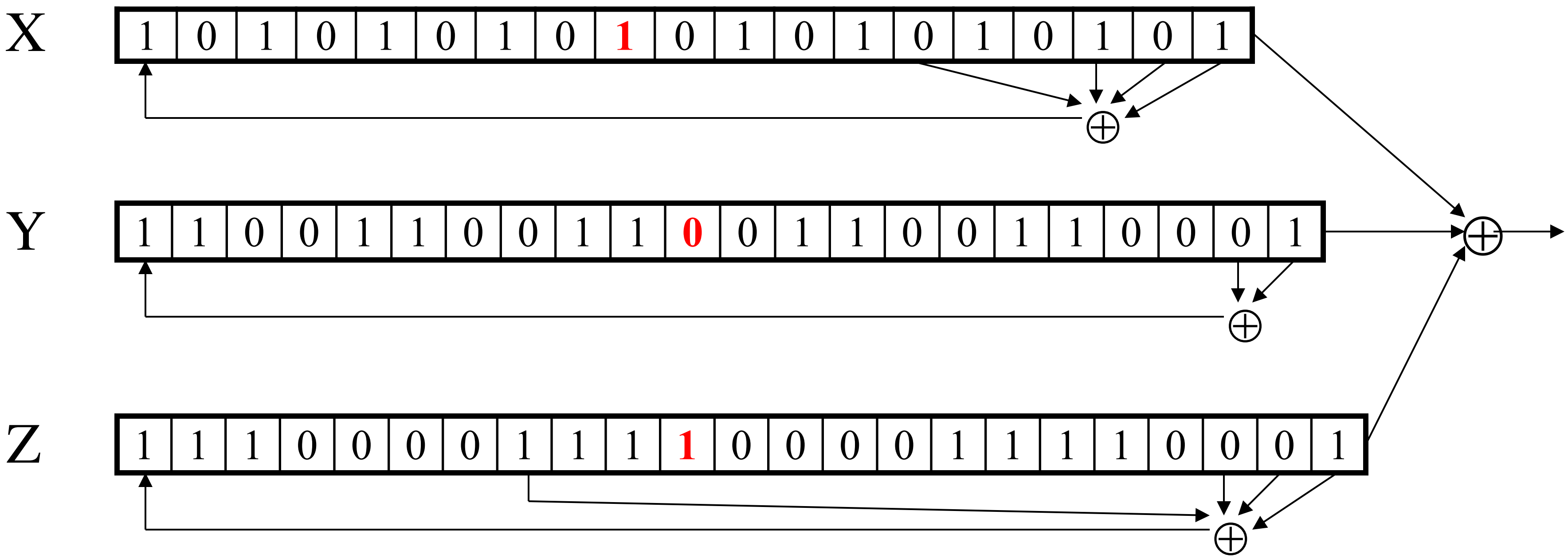
- At each iteration: $m = \text{maj}(x_8, y_{10}, z_{10})$
  - Examples: $\text{maj}(0,1,0) = 0$ and $\text{maj}(1,1,0) = 1$
- If $x_8 = m$ then X *steps*
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - $x_i = x_{i-1}$ for $i = 18,17,\ldots,1$ and $x_0 = t$
- If $y_{10} = m$ then Y *steps*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

# A5/1




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

# A5/1




- In this example, $m = \text{maj}(x_8, y_{10}, z_{10}) = \text{maj}(\mathbf{1},\mathbf{0},\mathbf{1}) = \mathbf{1}$
- Register X steps, Y does not step, and Z steps
- Keystream bit is XOR of right bits of registers
- Here, keystream bit will be $0 \oplus 1 \oplus 0 = 1$

# 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

# RC4

- A self-modifying lookup table
- Table always contains a permutation of the byte values $0,1,\ldots,255$
- Initialize the permutation using key
- At each step, RC4 does the following
  - Swaps elements in current lookup table
  - Selects a keystream byte from table
- Each step of RC4 produces a **byte**
  - Efficient in software
- Each step of A5/1 produces only a bit
  - Efficient in hardware

# RC4 Initialization

- S[] is permutation of 0,1,...,255
- key[] contains N bytes of key

```
for i = 0 to 255
        S[i] = i
        K[i] = key[i (mod N)]
next i
j = 0
for i = 0 to 255
        j = (j + S[i] + K[i]) mod 256
        swap(S[i], S[j])
next i
i = j = 0
```

# RC4 Keystream

- At each step, swap elements in table and select keystream byte

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap(S[i], S[j])
t = (S[i] + S[j]) mod 256
keystreamByte = S[t]
```

- Use keystream bytes like a one-time pad
- **Note:** first few hundreds bytes should be discarded
  - Otherwise, related key attack exists

# 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

# (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 $i = 1, 2, ..., n$, 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 function and $K_i$ is subkey
- Plaintext: $P = (L_0, R_0)$
- Decryption works for any function $F$
  - But only secure for certain functions $F$

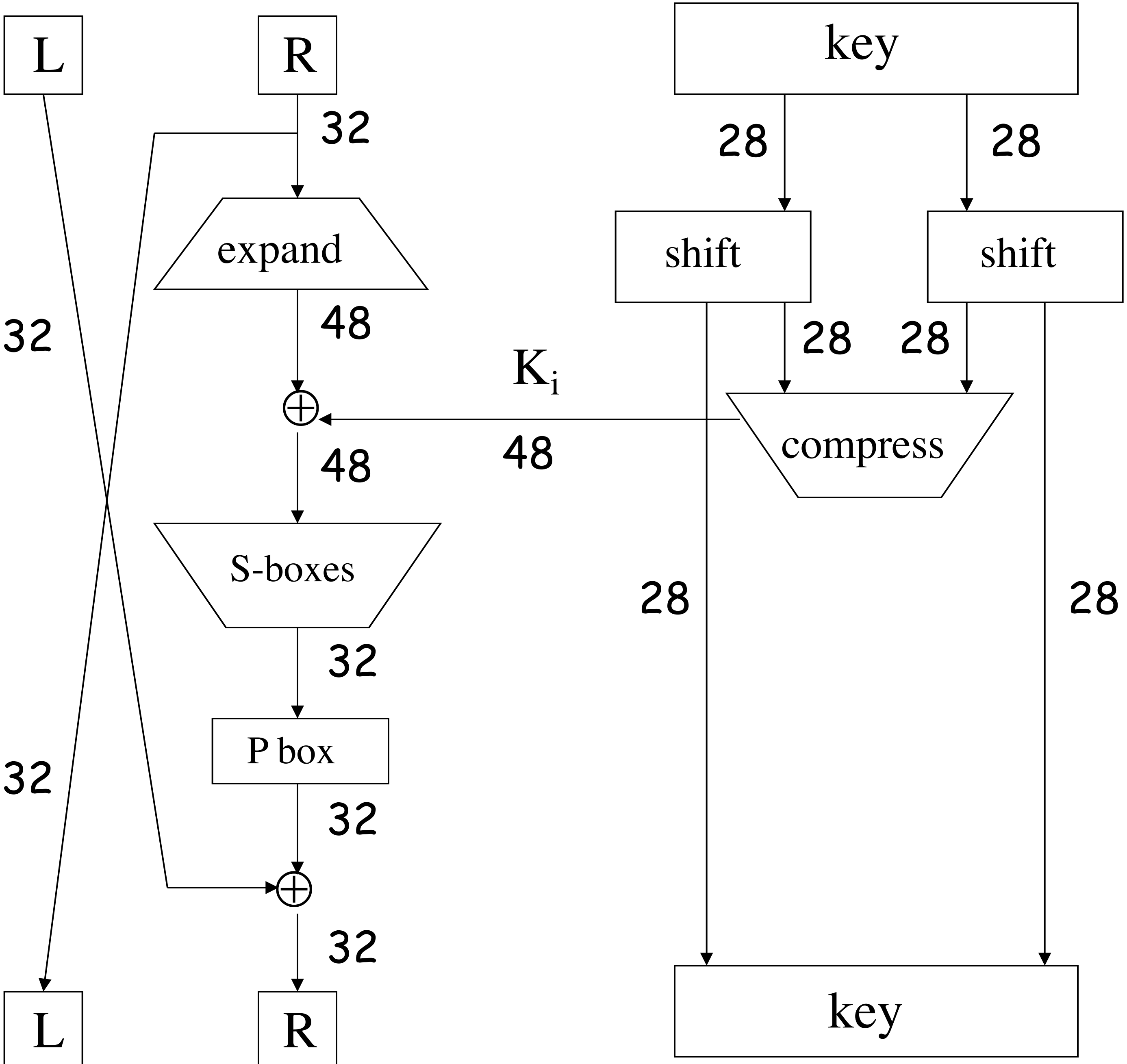
# Data Encryption Standard

- DES developed in 1970's
- Based on IBM's Lucifer cipher
- DES was U.S. government standard
- Development of 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 (subkey)
- 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



# 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 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 Subkey

- 56 bit DES key, numbered 0,1,2,…,55
- Left half key bits, LK

```
49 42 35 28 21 14  7
 0 50 43 36 29 22 15
 8  1 51 44 37 30 23
16  9  2 52 45 38 31
```

- Right half key bits, RK

```
55 48 41 34 27 20 13
 6 54 47 40 33 26 19
12  5 53 46 39 32 25
18 11  4 24 17 10  3
```

# DES Subkey

- For rounds `i=1,2,...,16`
  - Let LK = (LK circular shift left by $r_i$)
  - Let RK = (RK circular shift left by $r_i$)
  - Left half of subkey $K_i$ is of LK bits

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 16 | 10 | 23 | 0 | 4 | 2 | 27 | 14 | 5 | 20 | 9 |
| 22 | 18 | 11 | 3 | 25 | 7 | 15 | 6 | 26 | 19 | 12 | 1 |

  - Right half of subkey $K_i$ is RK bits

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 23 | 2 | 8 | 18 | 26 | 1 | 11 | 22 | 16 | 4 | 19 |
| 15 | 20 | 10 | 27 | 5 | 24 | 17 | 13 | 21 | 7 | 0 | 3 |

# DES Subkey

- For rounds 1, 2, 9 and 16 the shift $r_i$ is 1, and in all other rounds $r_i$ is 2
- Bits 8,17,21,24 of LK omitted each round
- Bits 6,9,14,25 of RK omitted each round
- **Compression permutation** yields 48 bit subkey $K_i$ from 56 bits of LK and RK
- **Key schedule** generates subkey

# 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 (at least wrt certain cryptanalytic techniques)