# Introduction to Cryptography

**Mark Tehranipoor**

**Introduction to Hardware Security & Trust**
**University of Florida**

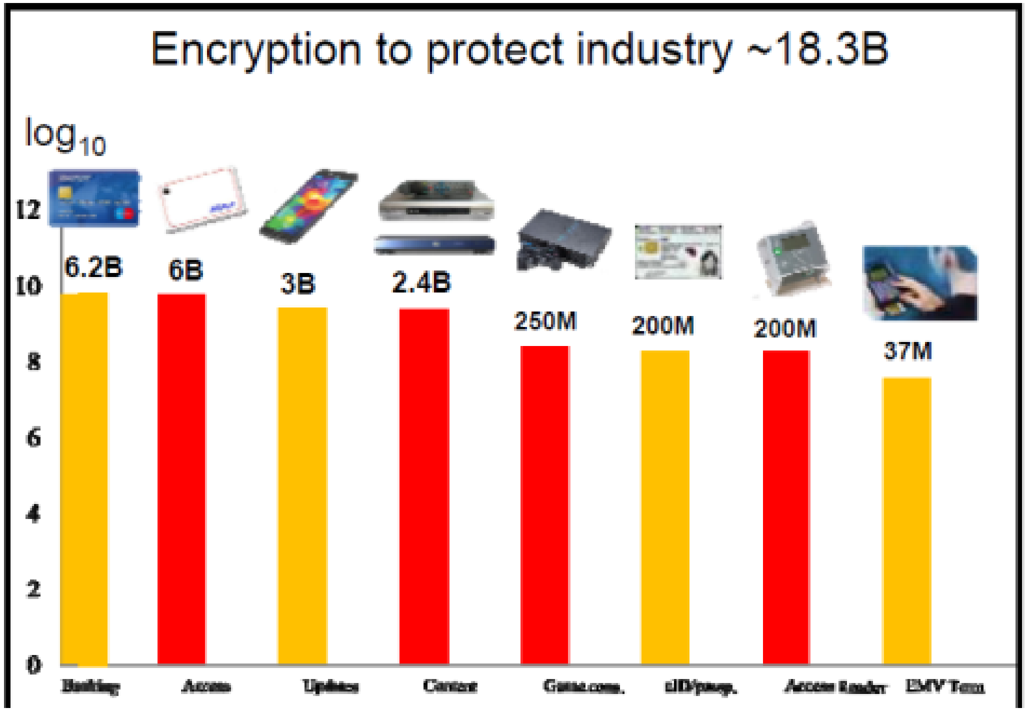
# Summary

- Substitution ciphers
- Permutations
- Making good ciphers
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

*Slides are courtesy of Leszek T. Lilien from WMich*
*http://www.cs.wmich.edu/~llilien/*

# Cryptography will play an increasingly Important Role …

- Crypto principles see growing usage in information protection
- A locking approach



**Cryptographic algorithms protects critical infrastructure and assets!**

# Threats to Messages

- Interception
- Interruption
  - Blocking msgs
- Modification
- Fabrication

**"A threat is blocked by control of a vulnerability"**

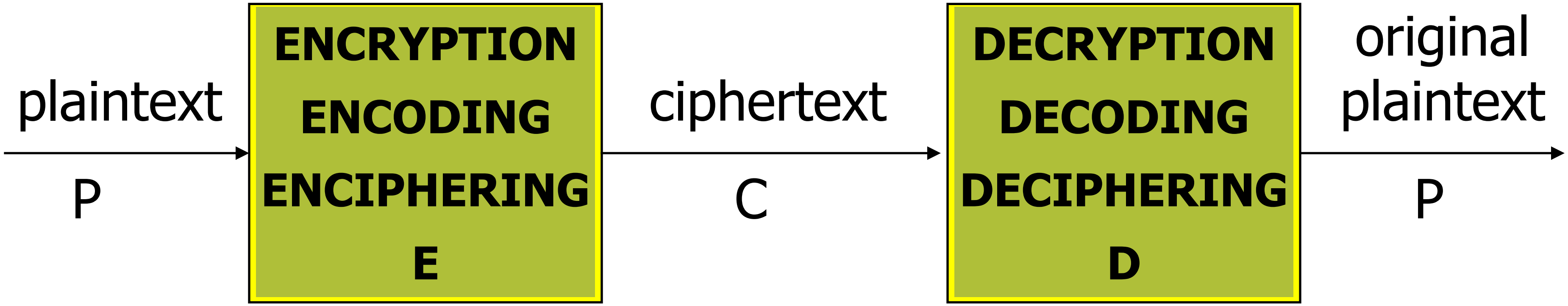
**[Pfleeger & Pfleeger]**

[cf. B. Endicott-Popovsky, U. Washington]

# Basic Terminology & Notation

- **Cryptology:**
  - cryptography + cryptanalysis
- **Cryptography:**
  - art/science of keeping message secure
- **Cryptanalysis:**
  - art/science of breaking ciphertext
    - *Enigma* in world war II
      - Read the real story – not fabrications!

# Basic Cryptographic Scheme



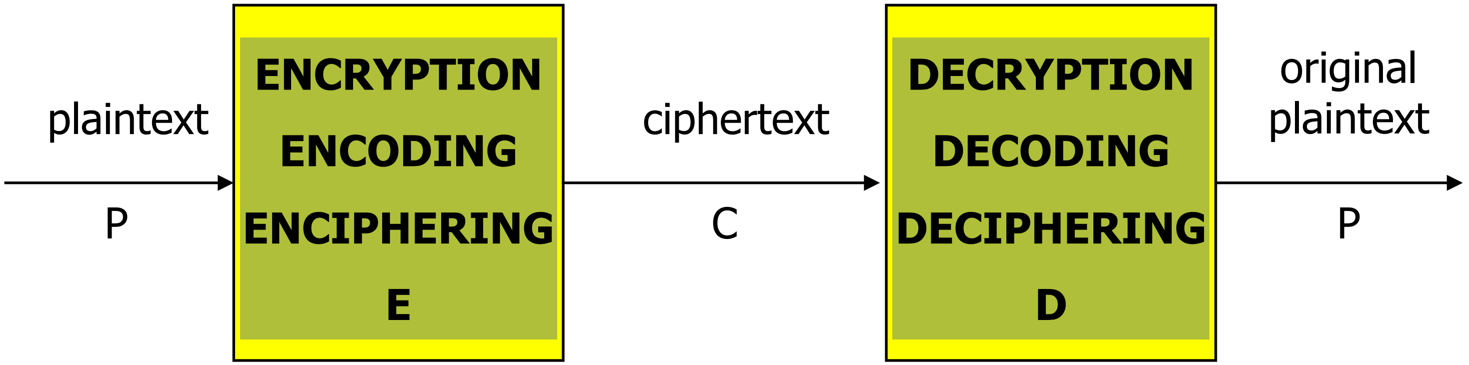
- $P = <p_1, p_2, ..., p_n>$ $p_i$ = i-th char of P
  - P = "DO NOT TELL ANYBODY" $p_1$ ="D", $p_2$ = "O", etc.
  - By convention, cleartext in uppercase
- $C = <c_1, c_2, ..., c_n>$ $c_i$ = i-th char of C
  - C = "ep opu ufmm bozcpez" $c_1$ ="e", $c_2$ ="p", etc.
  - By convention, ciphertext in lowercase

# Benefits of Cryptography

- **Improvement not a Solution!**
  - Minimizes problems
  - Doesn’t solve them
    - Remember: There is *no* solution!
  - Adds an envelope (encoding) to an open postcard (plaintext or cleartext)

[cf. D. Frincke, U. of Idaho]

# Formal Notation



plaintext

P

ENCRYPTION

ENCODING

ENCIPHERING

E

ciphertext

C

DECRYPTION

DECODING

DECIPHERING

D

original plaintext

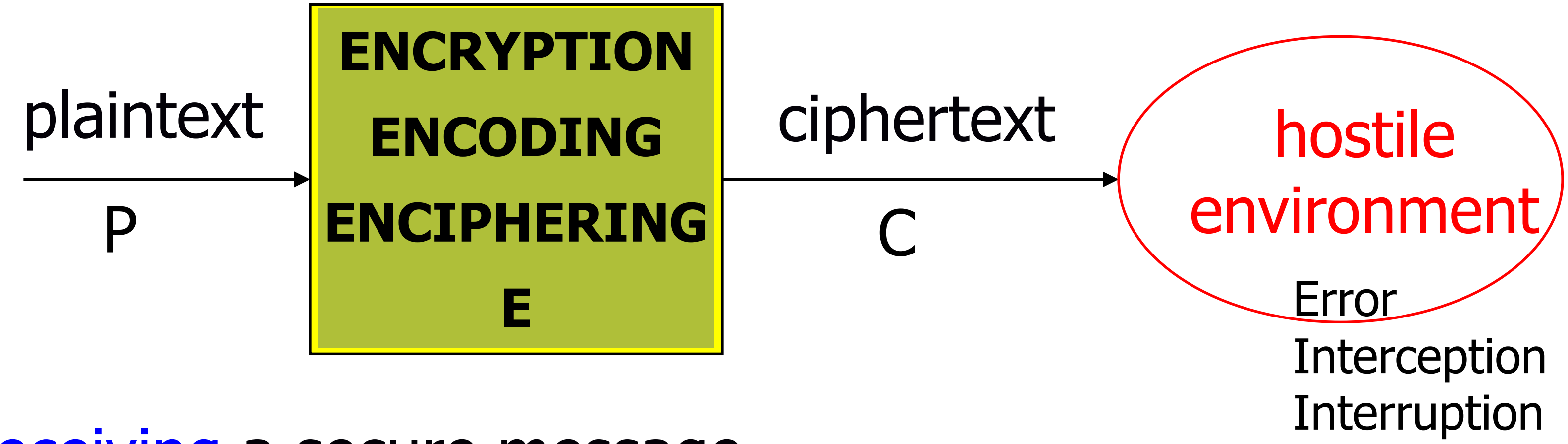
P

- C = E(P) E – encryption rule/algorithm
- P = D(C) D – decryption rule/algorithm

- We need a cryptosystem, where:
  - P = D(C)= D(E(P))
    - i.e., able to get the original message back

# Cryptography in Practice

- Sending a secure message



plaintext

P

ENCRYPTION

ENCODING

ENCIPHERING

E

ciphertext

C

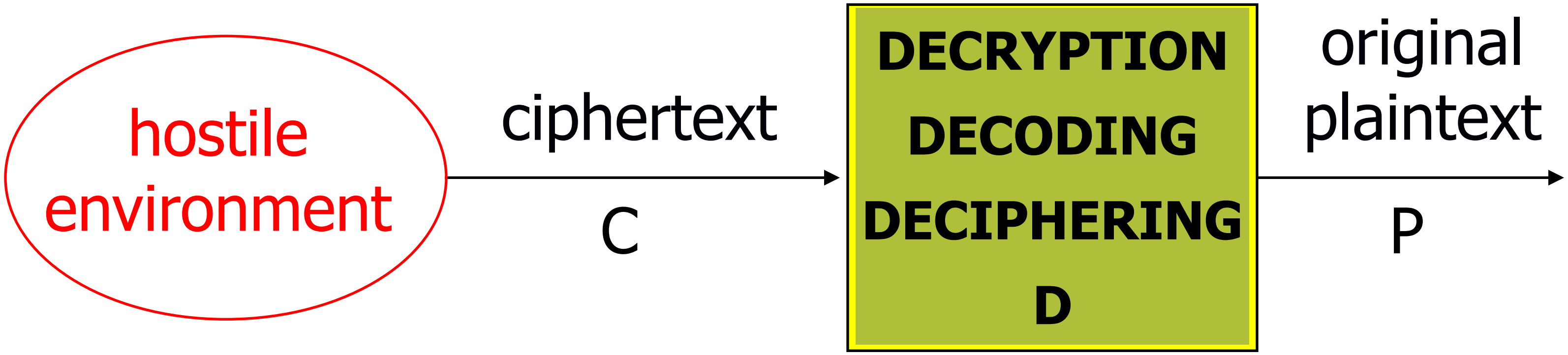
hostile environment

Error

Interception

Interruption

- Receiving a secure message



hostile environment

ciphertext

C

DECRYPTION

DECODING

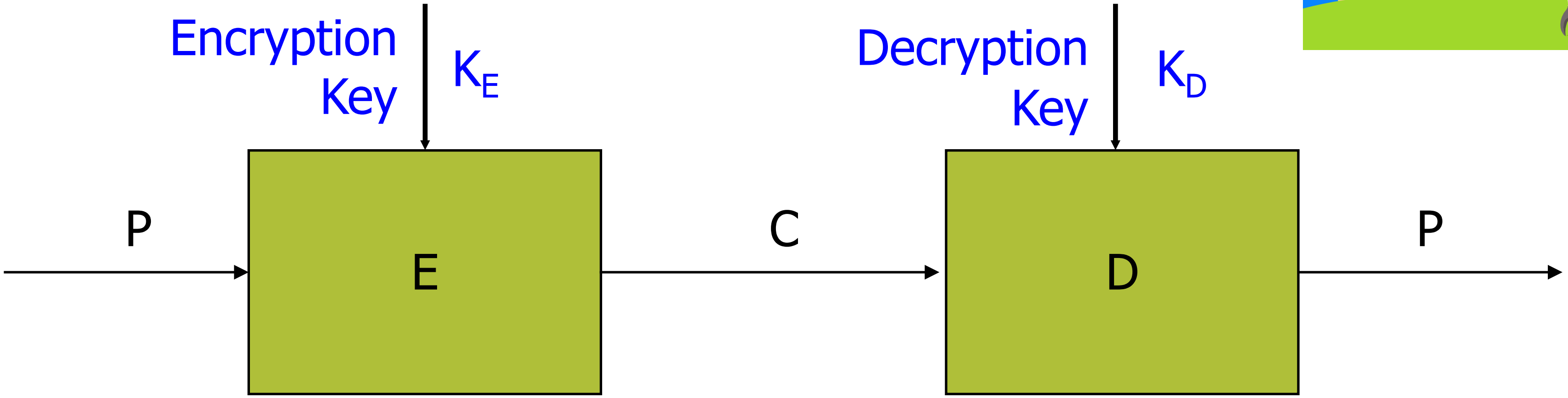
DECIPHERING

D

original plaintext

P

# Crypto System with Keys



- $C = E(K_E, P)$
  - $E$ = *set* of encryption algorithms / $K_E$ selects $E_i \in E$
- $P = D(K_D, C)$
  - $D$ = *set* of decryption algorithms / $K_D$ selects $D_j \in D$
- Crypto algorithms and keys are like door locks and keys
- We need: $P = D(K_D, E(K_E, P))$

# Classification of Cryptosystems w.r.t. Keys

- Keyless cryptosystems exist (e.g., Caesar's cipher)
  - Less secure

- Symmetric cryptosystems: $K_E = K_D$
  - Classic
  - Encipher and decipher using the same key
    - Or one key is easily derived from other

- Asymmetric cryptosystems: $K_E \neq K_D$
  - Public key system
  - Encipher and decipher using different keys
    - Computationally infeasible to derive one from other

[cf. B. Endicott-Popovsky, U. Washington]

# Cryptanalysis (1)

- **Cryptanalysts goals:**
  - Break a single msg
  - Recognize patterns in encrypted msgs, to be able to break the subsequent ones
  - Infer meaning w/o breaking encryption
    - Unusual volume of msgs between enemy troops may indicate a coming attack
    - Busiest node may be enemy headquarters
  - Deduce the key, to facilitate breaking subsequent msgs
  - Find vulnerabilities in implementation or environment of an encryption algorithm
  - Find a general weakness in an encryption algorithm

# Cryptanalysis (2)

- **Information for cryptanalysts:**
  - Intercepted encrypted msgs
  - Known encryption algorithms
  - Intercepted plaintext
  - Data known or suspected to be ciphertext
  - Math or statistical tools and techniques
  - Properties of natural languages
    - Esp. adversary’s natural language
      - To confuse the enemy, Americans used Navajo language in WW2
  - Propertiers of computer systems
- Role of ingenuity / luck
- There are *no* rules!!!

# Breakable Encryption (1)

- **Breakable encryption**
  - *Theoretically*, it is possible to devise unbreakable cryptosystems
  - *Practical* cryptosystems almost always are breakable, given adequate time and computing power
  - The trick is to make breaking a cryptosystem hard enough for the intruder

[cf. J. Leiwo, VU, NL]

# Breakable Encryption (2)

- Example: Breakability of an encryption algorithm

  Msg with just 25 characters
  - $26^{25}$ possible decryptions ~ $10^{35}$ decryptions
  - Only one is the right one
  - Brute force approach to find the right one:
    - At $10^{10}$ (10 bln) decryption/sec => $10^{35}$ / $10^{10}$ = $10^{16}$ sec = 10 bln yrs !
    - Infeasible with current technology
- Be smarter – use ingenuity
  - ❑ Could reduce $26^{25}$ to, say, $10^{15}$ decryptions to check

    At $10^{10}$ decr./sec => $10^{15}$ / $10^{10}$ = $10^{5}$ sec = ~ 1 day

# Requirements for Crypto Protocols

- Messages should get to destination
- Only the recipient should get it
- Only the recipient should see it
- Proof of the sender's identity
- Message shouldn't be corrupted in transit
- Message should be sent/received once

[cf. D. Frincke, U. of Idaho]

- Proofs that message was sent/received (non-repudiation)

# Representing Characters

- Letters (uppercase only) represented by numbers 0-25 (modulo 26).

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **...** | **X** | **Y** | **Z** |
| **0** | **1** | **2** | **3** | **...** | **23** | **24** | **25** |

- Operations on letters:

**A + 2 = C**

**X + 4 = B** (circular!)

**...**

# Basic Types of Ciphers

- Substitution ciphers
  - Letters of P replaced with other letters by E
- Transposition (permutation) ciphers
  - *Order* of letters in P rearranged by E
- Product ciphers
  - E "=" $E_1$ "+" $E_2$ "+" ... "+" $E_n$
    - Combine two or more ciphers to enhance the security of the cryptosystem

# Substitution Ciphers

- **Substitution Ciphers:**
  - **Letters of P replaced with other letters by E**

# The Caesar Cipher (1)

- $c_i = E(p_i) = p_i + 3 \bmod 26$ (*26* letters in the English alphabet)

  Change each letter to the third letter following it (circularly)

  A → D, B → E, ... X → A, Y → B, Z → C

- Can represent as a permutation $\pi$: $\pi(i) = i+3 \bmod 26$

  $\pi(0)=3$, $\pi(1)=4$, ...,

  $\pi(23)=26 \bmod 26=0$, $\pi(24)=1$, $\pi(25)=2$

- Key = 3, or key = ‘D’ (because D represents 3)

# The Caesar Cipher (2)

- Example [cf. B. Endicott-Popovsky]
  - P (plaintext): HELLO WORLD
  - C (ciphertext): khoor zruog

- Caesar Cipher is a monoalphabetic substitution cipher (= simple substitution cipher)
  One key is used
  One letter substitutes the letter in P

# Attacking a Substitution Cipher

- Exhaustive search
  - If the key space is small enough, try all possible keys until you find the right one
  - Cæsar cipher has 26 possible keys
    from A to Z  OR: from 0 to 25
- Statistical analysis (attack)
  - Compare to so called 1-gram (unigram) model of English
  - It shows frequency of (single) characters in English
  - The longer the C, the more effective statistical analysis would be

[cf. Barbara Endicott-Popovsky, U. Washington]

# 1-grams (Unigrams) for English

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | 0.080 | h | 0.060 | n | 0.070 | t | 0.090 |
| b | 0.015 | i | 0.065 | o | 0.080 | u | 0.030 |
| c | 0.030 | j | 0.005 | p | 0.020 | v | 0.010 |
| d | 0.040 | k | 0.005 | q | 0.002 | w | 0.015 |
| e | 0.130 | l | 0.035 | r | 0.065 | x | 0.005 |
| f | 0.020 | m | 0.030 | s | 0.060 | y | 0.020 |
| g | 0.015 | | | | | z | 0.002 |

[cf. Barbara Endicott-Popovsky, U.  Washington]

# Statistical Attack – Step 1

- Compute frequency f(c) of each letter c in ciphertext
- Example: c = ‘khoor zruog’
  - 10 characters: 3 * ‘o’, 2 * ‘r’, 1 * {k, h, z, u, g}
  - f(c):

    f(g)=0.1 f(h)=0.1 f(k)=0.1 f(o)=0.3 f(r)= 0.2

    f(u)=0.1 f(z)=0.1 $f(c_i)$ = 0 for any other $c_i$
- Apply 1-gram model of English
  - Frequency of (single) characters in English
  - 1-grams on previous slide

[cf. Barbara Endicott-Popovsky, U. Washington]

# Statistical Analysis – Step 2

- phi $\varphi(i)$ - correlation of frequency of letters *in ciphertext* with frequency of corresponding letters *in English* —for key i

- For key i: $\varphi(i) = \Sigma_{0 \leq c \leq 25}\ f(c) * p(c - i)$
  - *c* representation of character (a-0, …, z-25)
  - f(c) is frequency of letter c in ciphertext C
  - *p*(*x*) is frequency of character *x* in English
  - Intuition: sum of probabilities for words in P, if i were the key

c is a letter in ciphertext thus c-i is the letter in plaintext.

- Example: C = ‘khoor zruog’ (P = ‘HELLO WORLD’)

f(c): f(g)=0.1, f(h)=0.1, f(k)=0.1, f(o)=0.3, f(r)=0.2, f(u)=0.1, f(z)=0.1

c: g - 6, h - 7, k - 10, o - 14, r - 17, u - 20, z - 25

$$\varphi(i) = 0.1p(6 - i) + 0.1p(7 - i) + 0.1p(10 - i) +$$
$$+ 0.3p(14 - i) + 0.2p(17 - i) + 0.1p(20 - i) +$$
$$+ 0.1p(25 - i)$$

[cf. Barbara Endicott-Popovsky, U. Washington]

# Statistical Attack – Step 2a (Calculations)

- Correlation φ(i) for 0≤ i ≤25

| *i* | φ(*i*) | *i* | φ(*i*) | *i* | φ(*i*) | *i* | φ(*i*) |
|---|---|---|---|---|---|---|---|
| 0 | 0.0482 | 7 | 0.0442 | 13 | 0.0520 | 19 | 0.0315 |
| 1 | 0.0364 | 8 | 0.0202 | 14 | 0.0535 | 20 | 0.0302 |
| 2 | 0.0410 | 9 | 0.0267 | 15 | 0.0226 | 21 | 0.0517 |
| 3 | 0.0575 | 10 | 0.0635 | 16 | 0.0322 | 22 | 0.0380 |
| 4 | 0.0252 | 11 | 0.0262 | 17 | 0.0392 | 23 | 0.0370 |
| 5 | 0.0190 | 12 | 0.0325 | 18 | 0.0299 | 24 | 0.0316 |
| 6 | 0.0660 | | | | | 25 | 0.0430 |

[cf. Barbara Endicott-Popovsky, U. Washington]

# Statistical Attack – Step 3 (The Result)

- Most probable keys (largest φ(i) values):
  - i = 6, φ(i) = 0.0660
    - plaintext EBIIL TLOLA
  - i = 10, φ(i) = 0.0635
    - plaintext AXEEH PHKEW
  - i = 3, φ(i) = 0.0575
    - plaintext HELLO WORLD
  - i = 14, φ(i) = 0.0535
    - plaintext WTAAD LDGAS
- Only English phrase is for *i* = 3
  - That's the key (3 or 'D') – code broken

[cf. Barbara Endicott-Popovsky, U. Washington]

# Caesar’s Problem

- Conclusion: Key is too short
  - 1-char key – monoalphabetic substitution
    - Can be found by exhaustive search
    - Statistical frequencies not concealed well by short key
      - They look too much like ‘regular’ English letters
- Solution: Make the key longer
  - n-char key ($n \geq 2$) – polyalphabetic substitution
    - Makes exhaustive search much more difficult
    - Statistical frequencies concealed much better
      - Makes cryptanalysis harder

[cf. Barbara Endicott-Popovsky, U. Washington]

# Other Substitution Ciphers

**n-char key:**

- Polyalphabetic substitution ciphers
- Vigenere Tableaux cipher

# Polyalphabetic Substitution - Examples

- Flatten (difuse) *somewhat* the frequency distribution of letters by combining high and low distributions

- Example – 2-key substitution:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key1: | a | d | g | j | m | p | s | v | y | b | e | h | k | n | q | t | w | z | c | f | i | l | o | r | u | x |
| Key2: | n | s | x | c | h | m | r | w | b | g | l | q | v | a | f | k | p | u | z | e | j | o | t | y | d | i |

- **Question**:

  How Key1 and Key2 were defined?

[cf. J. Leiwo, VU, NL]

# Polyalphabetic Substitution - Examples

- ....

- Example:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key1: | a | d | g | j | m | p | s | v | y | b | e | h | k | n | q | t | w | z | c | f | i | l | o | r | u | x |
| Key2: | n | s | x | c | h | m | r | w | b | g | l | q | v | a | f | k | p | u | z | e | j | o | t | y | d | i |

- Answer:

  Key1 – start with ‘a’, skip 2, take next,
  skip 2, take next letter, ... (circular)

  Key2 - start with ‘n’ (2nd half of alphabet), skip 4,
  take next, skip 4, take next, ... (circular)

[cf. J. Leiwo, VU, NL]

# Polyalphabetic Substitution - Examples

– Example:

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key1: | a | d | g | j | m | p | s | v | y | b | e | h | k |
| Key2: | n | s | x | c | h | m | r | w | b | g | l | q | v |

| | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key1: | n | q | t | w | z | c | f | i | l | o | r | u | x |
| Key2: | a | f | k | p | u | z | e | j | o | t | y | d | i |

– **Plaintext: TOUGH STUFF**

– **Ciphertext: ffirv zfjpm**

use n (=2) keys in turn for consecutive P chars in P

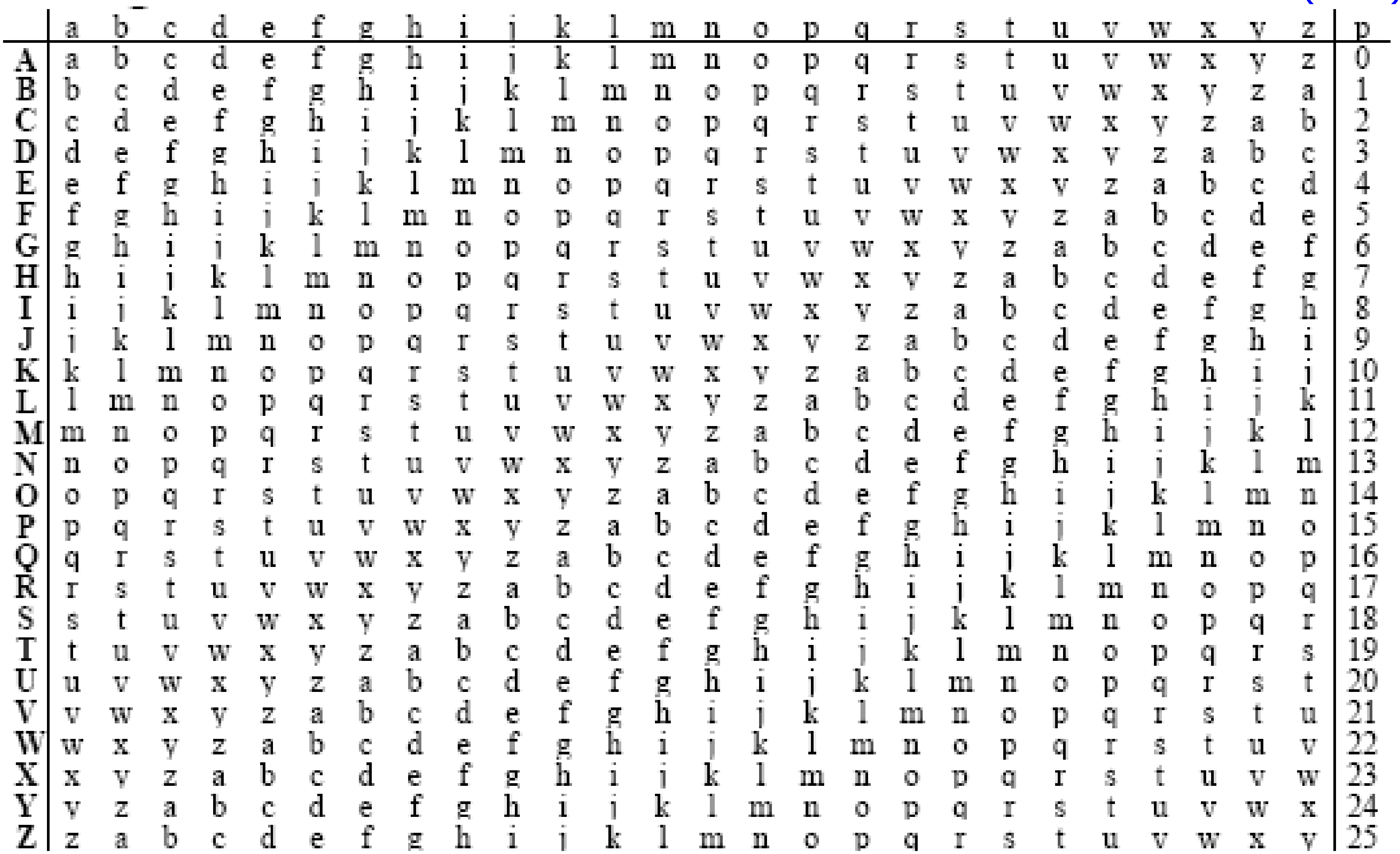
- Note:
  - Different chars mapped into the same one: **T, O → f**
  - Same char mapped into different ones: **F → p, m**
  - '**f**' most frequent in C (0.30); in English: f(**f**) = 0.02 << f(**e**) = 0.13

[cf. J. Leiwo, VU, NL]

# Vigenere Tableaux (1)

Note: Row A – shift 0 (a->a)
Row B – shift 1 (a->b)
Row C – shift 2 (a->c)
…
Row Z – shift 25 (a->z)

[cf. J. Leiwo, VU, NL]



| | 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 | p |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | 0 |
| B | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | 1 |
| C | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | 2 |
| D | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | 3 |
| E | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | 4 |
| F | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | 5 |
| G | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | 6 |
| H | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | 7 |
| I | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | 8 |
| J | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | 9 |
| K | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | 10 |
| L | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | 11 |
| M | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | 12 |
| N | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | 13 |
| O | o | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | 14 |
| P | p | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | 15 |
| Q | q | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | 16 |
| R | r | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | 17 |
| S | s | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | 18 |
| T | t | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | 19 |
| U | u | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | 20 |
| V | v | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | 21 |
| W | w | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | 22 |
| X | x | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | 23 |
| Y | y | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | 24 |
| Z | z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | 25 |

# Vigenère Tableaux (2)

- Example

  Key:

  **EXODUS**

  Plaintext P:

  **YELLOW SUBMARINE FROM YELLOW RIVER**

  Extended keyword (re-applied to mimic words in P):

  **YELLOW SUBMARINE FROM YELLOW RIVER**

  **EXODUS EXODUSEXO DUSE XODUSE XODUS**

  Ciphertext:

  **cbxoio wlppujmks ilgq vsofhb owyyj**

[cf. J. Leiwo, VU, NL]

# Vigenère Tableaux (3)

- Example

  …

  Extended keyword (re-applied to mimic words in P):

  ```
  YELLOW SUBMARINE FROM YELLOW RIVER
  EXODUS EXODUSEXO DUSE XODUSE XODUS
  ```

  Ciphertext:

  ```
  cbzoio wlppujmks ilgq vsofhb owyyj
  ```

- Answer:

  c from P indexes row

  c from extended key indexes column

  e.g.: row Y and column e → 'c'

  row E and column x → 'b'

  row L and column o → 'z'

  …

[cf. J. Leiwo, VU, NL]

# Transposition Ciphers (1)

- Rearrange letters in plaintext to produce ciphertext
- Example 1a and 1b: Columnar transposition
  - Plaintext: **HELLO WORLD**
  - Transposition onto: (a) 3 columns:

**HEL**
**LOW**
**ORL**
**DXX**

**XX** - padding

  - Ciphertext (read column-by column):

(a) **hlodeorxlwlx**

(b) onto 2 columns:

```
HE
LL
OW
OR
LD
```

(b) **hloolelwrd**

  - What is the key?
    - Number of columns: (a) key = 3 and (b) key = 2

# Transposition Ciphers (2)

- Example 2: Rail-Fence Cipher
  - Plaintext: **HELLO WORLD**
  - Transposition into 2 rows (rails) column-by-column:

    **HLOOL**

    **ELWRD**
  - Ciphertext: **hloolelwrd** (Does it look familiar?)

    [cf. Barbara Endicott-Popovsky, U. Washington]
  - What is the key?
    - Number of rails key = 2

# Product Ciphers

- A.k.a. combination ciphers
- Built of multiple blocks, each is:
  - Substitution

or:

  - Transposition
- Example: two-block product cipher
  - $E_2(E_1(P, K_{E1}), K_{E2})$
- Product cipher might *not* necessarily be stronger than its individual components used separately!
  - Might not be even as strong as individual components

# Criteria for “Good” Ciphers

- “Good” depends on intended application
  - Substitution
    - C hides chars of P
    - If > 1 key, C dissipates high frequency chars
  - Transposition
    - C scrambles text => hides n-grams for n > 1
  - Product ciphers
    - Can do all of the above
  - What is more important for your app?

    What facilities available to sender/receiver?
    - E.g., no supercomputer support on the battlefield

# Criteria for “Good” Ciphers

- **Commercial Principles of Sound Encryption Systems**
  1. Sound mathematics
     - Proven vs. not broken so far
  2. Verified by expert analysis
     - Including outside experts
  3. Stood the test of time
     - Long-term success is not a guarantee
       - Still. Flows in many E’s discovered soon after their release

- Examples of popular commercial encryption:
  - DES / RSA / AES

DES = Data Encryption Standard
RSA = Rivest-Shamir-Adelman
AES = Advanced Encryption Standard (rel. new)

[cf. A. Striegel]

# Stream and Block Ciphers (1)

a. Stream ciphers

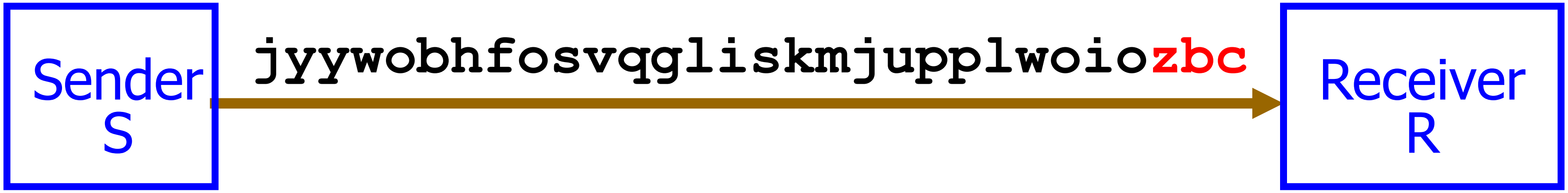
b. Problems with stream ciphers

c. Block ciphers

d. Pros / cons for stream and block ciphers

# Stream Ciphers (1)

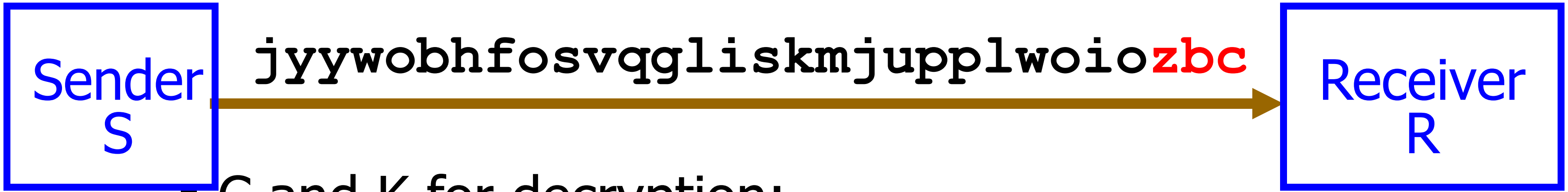
- Stream cipher: 1 char from P → 1 char for C
  - Example: polyalphabetic cipher
    - P and K (repeated 'EXODUS'):
      **YELLOWSUBMARINEFROMYELLOWRIVER**
      **EXODUSEXODUSEXODUSEXODUSEXODUS**
    - Encryption (char after char, using Vigenère Tableaux):
      (1) E(**Y**, **E**) → **c** (2) E(**E**, **X**) → **b** (3) E(**L**, **O**) → **z** . . .
    - C: **cbzoiowlppujmksilgqvsofhbowyyj**
    - C as sent (in the right-to-left order):



**jyywobhfosvqgliskmjupplwoiozbc**

# Stream Ciphers (2)

- Example: polyalphabetic cipher - cont.
  - C as received (in the right-to-left order):



Sender S

jyywobhfosvqgliskmjupplwoiozbc

Receiver R

  - C and K for decryption:

    cbzoiowlppujmksilgqvsofhbowyyj
    EXODUSEXODUSEXODUSEXODUSEXODUS

  - Decryption:

    (1) D(c, E) → Y (2) D(b, X) → E (3) D(z, O) → L . . .

  - Decrypted P:

    YEL. . .

Q: Do you know how D uses Vigenère Tableaux?

A: Finds c under column e → Y

# Problems with Stream Ciphers (1)

- Problems with stream ciphers
  - Dropping a char from key K results in wrong decryption
  - Example:
    - P and K (repeated '**EXODUS**') with a char in K missing:

      **YELLOWSUBMARINEFROMYELLOWRIVER**
      **EODUSEXODUSEXODUSEXODUSEXODUSE**

      missing X in K ! (no errors in repeated K later)

    - Encryption

      (using VT):

      1) E(**Y**,**E**) → **c**
      2) E(**E**,**O**) → **s**
      3) E(**L**,**D**) → **o**

      **. . .**

    - Ciphertext: **cso...**

      C in the order as sent (right-to-left):

      **...osc**

# Problems with Stream Ciphers (2)

- C as received (in the right-to-left order):

  `...osc` →

- C and correct K (‘**EXODUS**’) for decryption:

  `cso...`

  `EXO...`

- Decryption (using VT, applying correct key):

  1) D(**c**, **E**) → **Y**
  2) D(**s**, **X**) → **V**
  3) D(**o**, **O**) → **A**

  …

- Decrypted P:

  **YVA**... - Wrong!

  – We know it’s wrong, Receiver might not know it *yet*!

What if message is corrupted in a noisy area?

# Problems with Stream Ciphers (3)

- The problem might be recoverable
  - Example:

    If R had more characters decoded, R might be able to detect that S dropped a key char, and R could recover

    - E.g., suppose that R decoded:

      **`YELLOW SUBMAZGTR`**

      - R could guess, that the 2nd word should really be:

        **`SUBMARINE`**

      - => R would know that S dropped a char from K after sending "**`SUBMA`**"
      - => R could go back 4 chars, drop a char from K ("recalibrate K with C"), and get "resynchronized" with S

# Block Ciphers (1)

- We can do better than using recovery for stream ciphers
  - Solution: use block ciphers

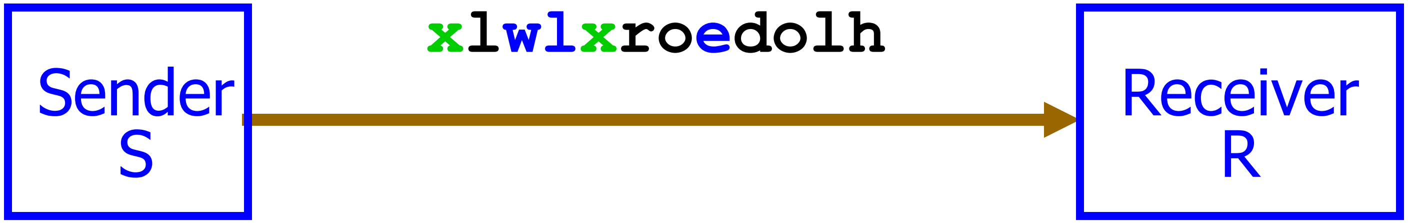
- Block cipher:
  1 *block* of chars from P → 1 *block* of chars for C
  - Example of block cipher: columnar transposition
  - Block size = "o(message length)" (informally)

# Block Ciphers (2)

- Why block size = "o(message length)" ?
  - Because R must wait for "almost" the entire C before R can decode some characters near beginning of P
  - E.g., for P = `HELLO WORLD`, block size is "o(10)"
  - Suppose that Key = 3 (3 columns):

```
HEL
LOW
ORL
DXX
```

  - C as sent (in the right-to-left order):



`xlwlxroedolh`

# Block Ciphers (3)

– C as received (in the right-to-left order): `xlwlxroedolh`

– R knows: K = 3, block size = 12 (=> 4 rows)

```
123
456
789
abc
```

a=10
b=11
c=12

=> R knows that characters wil be sent in the order:
1st-4th-7th-10th--2nd-5th-8th-11th--3rd-6th-9th-12th

– R must wait for at least:
- 1 char of C to decode 1st char of P (‘**h**’)
- 5 chars of C to decode 2nd char of P (‘**he**’)
- 9 chars of C to decode 3rd, 4th, and 5th chars of P (‘**hello**’)
- 10 chars of C to decode 6th, 7th, and 8th chars of P (‘**hello wor**’)
- etc.

# Block Ciphers (4)

- *Informally*, we might call ciphers like the above example columnar transposition cipher "weak-block" ciphers
  - R can get some (even most) but not all chars of P before entire C is received
    - R can get one char of P immediately
      - » the 1st-after 1 of C (delay of 1 - 1 = 0)
    - R can get some chars of P with "small" delay
      - » e.g., 2nd-after 5 of C (delay of 5 - 2 = 3)
    - R can get some chars of P with "large" delay
      - » e.g., 3rd-after 9 of C (delay of 9 – 3 = 6)

- There are block ciphers when R cannot even start decoding C before receiving the entire C
  - *Informally*, we might call them "strong-block" ciphers

# Pros / Cons for Stream and Block Ciphers (1)

- Pros / cons for stream ciphers
  - + Low delay for decoding individual symbols
    - Can decode as soon as received
  - + Low error propagation
    - Error in E($c_1$) does not affect E($c_2$)
  - - Low diffusion
    - Each char separately encoded => carries over its frequency info
  - - Susceptibility to malicious insertion / modification
    - Adversary can fabricate a new msg from pieces of broken msgs, even if he doesn’t know E (just broke a few msgs)

# Pros / Cons for Stream and Block Ciphers (2)

- Pros / cons for block ciphers
  - + High diffusion
    - Frequency of a *char* from P diffused over (a few chars of) a *block* of C
  - + Immune to insertion
    - Impossible to insert a char into a block without easy detection (block size would change)
    - Impossible to modify a char in a block without easy detection (if checksums are used)

# Pros / Cons for Stream and Block Ciphers (3)

- Pros / cons for block ciphers — Part 2
  - - High delay for decoding individual chars
    - See example for ‘`hello worldxx`’ above
      - For some E can’t decode even the 1st char before whole k chars of a block are received
  - - High error propagation
    - It affects the block, not just a single char

# Cryptanalysis (1)

- What cryptanalysts do when confronted with unknown?

  Four possible situations w.r.t. available info:

  1) C available
  2) Full P available
  3) Partial P available
  4) E available (or D available)

- (1) – (4) suggest 5 different approaches

# Cryptanalysis (2)

- Cryptanalyst approaches
  1) Ciphertext-only attack
     - We have shown examples for such attacks
       - E.g., for Caesar’s cipher, columnar transposition cipher
  2) Known plaintext attack
     - Analyst have C and P
       - Needs to deduce E such that C=E(P), then finds D
  3) Probable plaintext attack
     - Partial decryption provides partial match to C
       - This provides more clues

# Cryptanalysis (3)

- Cryptanalyst approaches – cont.

4) Chosen plaintext attack

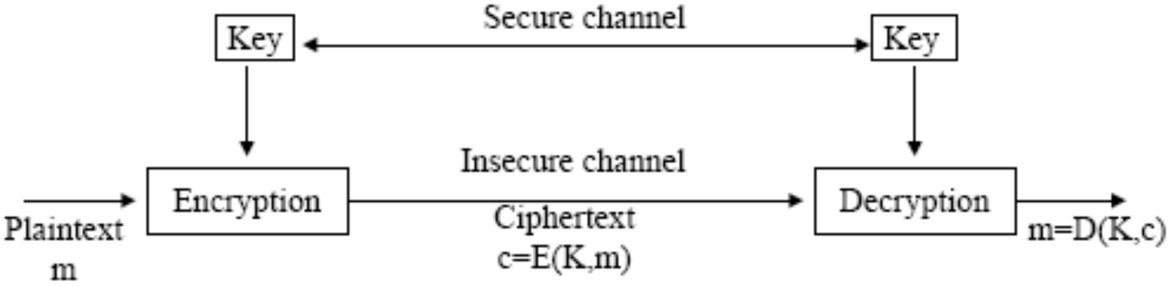
  - Analyst able to fabricate encrypted msgs
    - Then observe effects of msgs on adversary's actions
      - » This provides further hints

5) Chosen ciphertext attack

  - Analyst has both E and C
  - Run E for many candidate plaintexts to find P for which E(P) = C
    - Purpose: to find $K_E$

# Symmetric and Asymmetric Cryptosystems (1)

- Symmetric encryption = secret key encryption
  - $K_E = K_D$ — called a secret key or a private key
  - Only sender S and receiver R know the key



[cf. J. Leiwo]

  - As long as the key remains secret, it also provides authentication (= proof of sender's identity)

# Symmetric and Asymmetric Cryptosystems (3)

- Asymmetric encryption = public key encryption (PKE)
  - $K_E \neq K_D$ — public and private keys
- PKE systems eliminate symmetric encryption problems
  - Need no secure key distribution channel
    - => easy key distribution

# Symmetric and Asymmetric Cryptosystems (4)

- One PKE approach:
  - R keeps her private key $K_D$
  - R can distribute the correspoding public key $K_E$ to anybody who wants to send encrypted msgs to her
    - No need for secure channel to send $K_E$
    - Can even post the key on an open Web site — it is public!
  - Only private $K_D$ can decode msgs encoded with public $K_E$!
    - Anybody ($K_E$ is public) can encode
    - Only owner of $K_D$ can decode

# DES (Data Encryption Standard)

# Background and History of DES (1)

- Early 1970's - NBS (Nat'l Bureau of Standards) recognized general public's need for a secure crypto system

  NBS – part of US gov't / Now: NIST – Nat'l Inst. of Stand's & Technology

  - "Encryption for the masses" [A. Striegel]
  - Existing US gov't crypto systems were not meant to be made public
    - E.g. DoD, State Dept.
  - Problems with proliferation of commercial encryption devices
    - Incompatible
    - Not extensively tested by independent body

# Background and History of DES (2)

- 1972 - NBS calls for proposals for a *public* crypto system
  - Criteria:
    - Highly secure / easy to understand / publishable / available to all / adaptable to diverse app's / economical / efficient to use / able to be validated / exportable
    - ▪ In truth: Not *too* strong (for NSA, etc.)
- 1974 – IBM proposed its Lucifer
  - DES *based* on it
  - Tested by NSA (Nat'l Security Agency) and the general public
- Nov. 1976 – DES adopted as US standard for *sensitive but unclassified* data / communication
  - Later adopted by ISO (Int'l Standards Organization)
  - Official name: DEA - Data Encryption Algorithm / DEA-1 abroad

# Overview of DES

- DES - a block cipher
  - a product cipher
  - 16 rounds (iterations) on the input bits (of P)
    - substitutions (for confusion) and permutations (for diffusion)
  - Each round with a *round key*
    - Generated from the user-supplied key
- Easy to implement in S/W or H/W

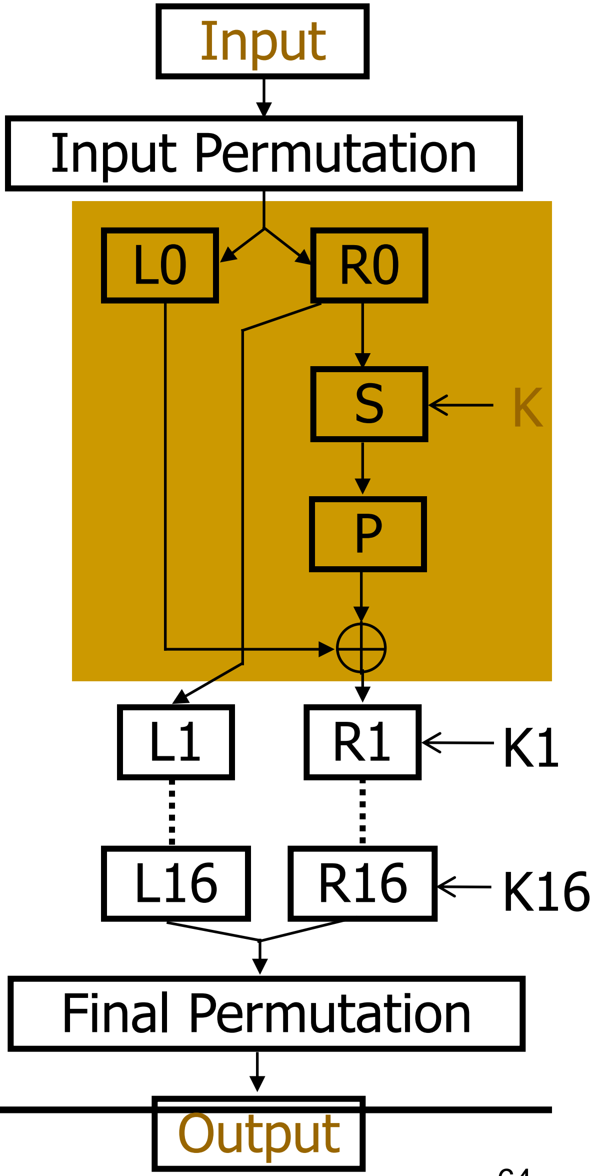
- There are 72,000,000,000,000,000 (72 quadrillion) or more possible encryption keys that can be used.
- For each given message, the key can be chosen at random from among this enormous number of keys.

# Basic Structure

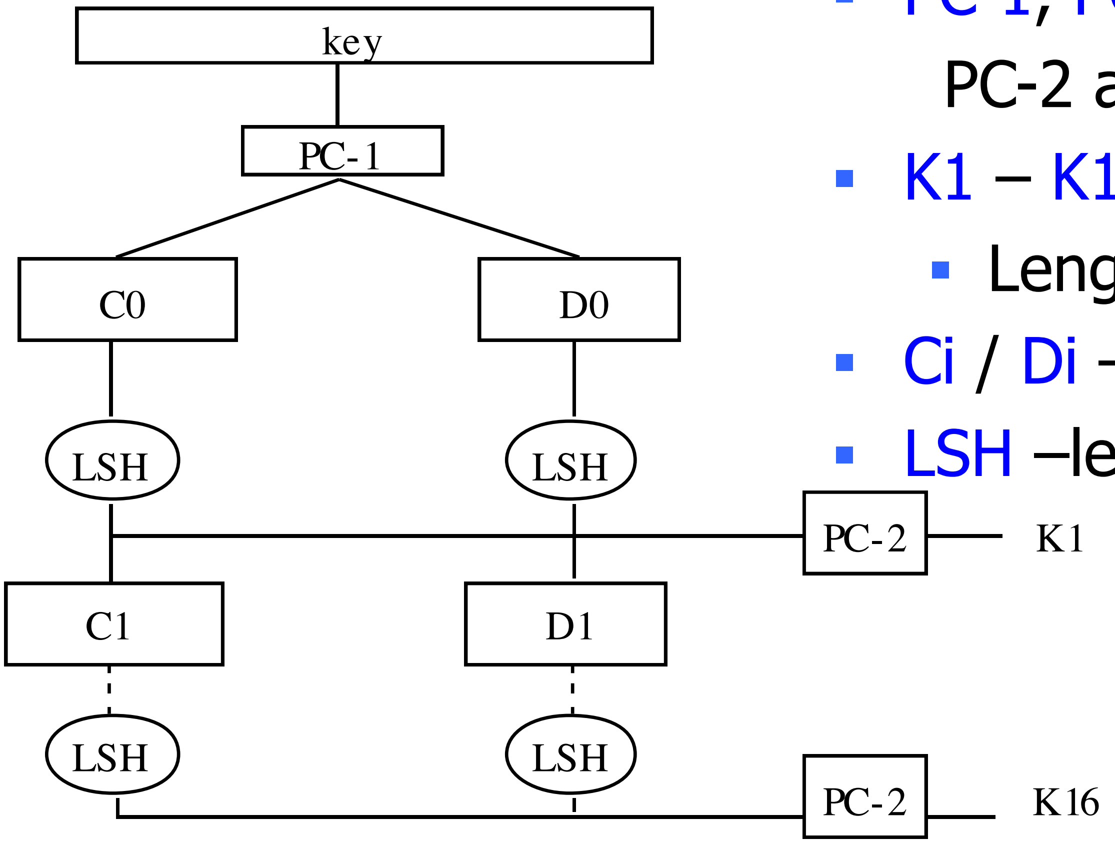
[Fig. – cf. J. Leiwo]

- Input: 64 bits (a block)
- Li/Ri– left/right half of the input block for iteration i (32 bits) – subject to substitution S and permutation P
- K - user-supplied key
- Ki - round key:
  - 56 bits used +8 unused

    (unused for E but often used for error checking)
- Output: 64 bits (a block)
- Note: Ri becomes L(i+1)
- All basic op's are simple logical ops
  - Left shift / XOR



# Generation of Round Keys



- key – user-supplied key (input)
- PC-1, PC-2 – permutation tables

  PC-2 also extracts 48 of 56 bits
- K1 – K16 – round keys (outputs)
  - Length(Ki) = 48
- Ci / Di – confusion / diffusion (?)
- LSH –left shift (rotation) tables

[Fig: cf. Barbara Endicott-Popovsky, U.  Washington]

# Problems with DES

- Diffie, Hellman 1977 prediction: "In a few years, technology would allow DES to be broken in days."
- Key length is fixed (= 56)
  - $2^{56}$ keys ~ $10^{15}$ keys
  - "Becoming" too short for faster computers
    - 1997: 3,500 machines – 4 months
    - 1998: special "DES cracker" h/w – 4 days
- Design decisions not public
  - Suspected of having backdoors
    - Speculation: To facilitate government access?

# Double and Triple DES

- Double DES:
  - Use double DES encryption

    C = E(k2, E(k1, P) )
  - Expected to multiply difficulty of breaking the encryption
    - Not true!
      - In general, 2 encryptions are not better than one

        [Merkle, Hellman, 1981]
      - Only doubles the attacker's work

# Double and Triple DES (2)

- Triple DES:
  - Is it C = E(k3, E(k2, E(k1, P) ) ?
  - Not soooo simple!

# Double and Triple DES (3)

- Triple DES: *Is it C=E(k3, E(k2, E(k1, P))?*
  - Tricks used:

    D not E in the 2nd step, k1 used twice (in steps 1 & 3)
  - It is:

    C = E(k1, D(k2, E(k1, P) )

    and

    P = D(k1, E(k2, D(k1, C) )
  - Doubles the effective key length
    - 112-bit key is quite strong
      - Even for today’s computers
      - For all feasible known attacks

# Security of DES

- So, is DES insecure?
- No, not yet
  - 1997 attack required a lot of cooperation
  - The 1998 special-purpose machine is still very expensive
  - Triple DES still beyond the reach of these 2 attacks
- But ...
  - In 1995, NIST (formerly NBS) began search for new strong encryption standard

# The AES Contest (1)

- 1997 – NIST calls for proposals NIST (Nat'l Institute of Standards and Technology)
  - Criteria:
    - Unclassifed code
    - Publicly disclosed
    - Royalty-free worldwide
    - Symmetric block cipher for 128-bit blocks
    - Usable with keys of 128, 192, and 256 bits
- 1998 – 15 algorithms selected

# The AES Contest (2)

- 1999 – 5 finalists [cf. J. Leiwo]
  - MARS by IBM
  - RC6 by RSA Laboratories
  - Rijndael (RINE-dahl) by Joan Daemen and Vincent Rijmen
  - Serpent by Ross Anderson, Eli Biham and Lars Knudsen
  - Twofish by Bruce Schneier, John Kelsey, Doug Whiting, Dawid Wagner, Chris Hall and Niels Ferguson
- Evaluation of finalists
  - Public and private scrutiny
  - Key evaluation areas:

    security / cost or efficiency of operation /

    ease of software implementation

# The AES Contest (3)

- 2001- … and the winner is …

  Rijndael (RINE-dahl)

  Authors: Vincent Rijmen + Joan Daemen (Dutchmen)

- Adopted by US gov't as

  Federal Info Processing Standard 197 (FIPS 197)

# Overview of Rijndael/AES

- Similar to DES – cyclic type of approach
  - 128-bit blocks of P
  - # of iterations based on key length
    - 128-bit key => 9 "rounds" (called rounds, not cycles)
    - 192-bit key => 11 rounds
    - 256-bit key => 13 rounds
- Basic ops for a round:
  - Substitution – byte level (confusion)
  - Shift row (transposition) – depends on key length (diff.)
  - Mix columns – LSH and XOR (confusion +diffusion)
  - Add subkey – XOR used (confusion)

# Strengths of AES

- Extensive cryptanalysis by US gov’t and independent experts
- Dutch inventors have no ties to NSA or other US gov’t bodies (less suspicion of trapdoor)
- Solid math basis
  - Despite seemingly simple steps within rounds

# Comparison of DES & AES (1)

| | DES | AES |
|---|---|---|
| Date | 1976 | 1999 |
| Block size [bits] | 64 | 128 |
| Key length [bits] | 56 (effect.) | 128, 192, 256, or more |
| Encryption Primitives | substitution, permutation | substitution, shift, bit mixing |
| Cryptographic Primitives | confusion, diffusion | confusion, diffusion |
| Design | open | open |
| Design Rationale | closed | open |
| Selection process | secret | secret, but accepted public comments |
| Source | IBM, enhanced by NSA | independent Dutch cryptographers |

# Comparison of DES & AES (2)

- Weaknesses in AES?
  - 20+ yrs of experience with DES eliminated fears of its weakness (intentional or not)
    - Might be naïve…
  - Experts pored over AES for 2-year review period

# Comparison of DES & AES (3)

- Longevity of AES?
  - DES is nearly 40 yrs old (1976)
    - DES-encrypted message can be cracked in days
  - Longevity of AES more difficult to answer
    - Can extend key length to > 256 bits (DES: 56)
      - 2 * key length => 4 * number of keys
    - Can extend number of rounds (DES: 16)
  - Extensible AES seems to be significantly better than DES, but..
    - Human ingenuity is unpredicatble!

    => Need to incessantly search for better and better encryption algorithms

# Motivation for PKE (1)

- So far - cryptosystems with secret keys
- Problems:
  - A lot of keys
    - o($n^2$) keys for n users (n * (n-1) /2 keys)

      — if each must be able to communicate with each
  - Distributing so many keys securely
  - Secure storage for the keys
    - User with n keys can’t just memorize them
- Can have a system with significantly fewer keys?

  Yes!

# Motivation for PKE (2)

Whitfield Diffie

- 1976 — Diffie and Hellman — new kind of cryptosystem:

  public key cryptosystem = asymmetric cryptosystem

  – Key pairs: < $k_{PRIVATE}$, $k_{PUBLIC}$>

  Martin E. Hellman

  – Each user owns one private key

  – Each user shares the corresponding public key with n-1 remaining users => n users share each public key

  – Only 2n keys for n users $\rightarrow$ 2n = n * (1 + n * 1/n)

  » Since public key is shared by n people: 1 "owner" + (n-1) others = n

  » 1/n since each part "owns" 1/n of the public key

  - Even if each communicates with each
  - Reduction from $o(n^2)$ to $o(n)$ !
  - n key pairs are:

<$k_{PRIV-1}$, $k_{PUB-1}$ >, <$k_{PRIV-2}$, $k_{PUB-2}$>, ..., <$k_{PRIV-n}$, $k_{PUB-n}$>

# Characteristics of PKE (1)

- PKE requirements
  1. It must be computationally easy to encipher or decipher a message given the appropriate key
  2. It must be computationally infeasible to derive $k_{PRIV}$ from $k_{PUB}$
  3. It must be computationally infeasible to determine $k_{PRIV}$ from a chosen plaintext attack

[cf. Barbara Endicott-Popovsky, U. Washington]

# Characteristics of PKE (2)

- Key pair characteristics
  - One key is inverse of the other key of the pair
    - i.e., it can undo encryption provided by the other:
      - $D(k_{PRIV}, E(k_{PUB}, P)) = P$
      - $D(k_{PUB}, E(k_{PRIV}, P)) = P$
  - One of the keys can be public since each key does only half of E ”+” D
    - As shown above – need both E and D to get P back

# Characteristics of PKE (3)

- Two E/D possibilities for key pair <$k_{PRIV}$, $k_{PUB}$ >
  - – $P = D(k_{PRIV}, E(k_{PUB}, P))$
    - User encrypts msg with $k_{PUB}$ ($k_{PUB''}$ "locks")
    - Recipient decrypts msg with $k_{PRIV}$ ($k_{PRIV}$ "unlocks")

  OR

  - – $P = D(k_{PUB}, E(k_{PRIV}, P))$ (e.g., in RSA)
    - User encrypts msg with $k_{PRIV}$ ($k_{PRIV}$ "locks")
    - Recipient decrypts msg with key $k_{PUB}$ ($k_{PUB}$ "unlocks")
- Do we still need symmetric encryption (SE) systems?
  - – Yes, PKEs are 10,000+ times (!) slower than SEs
    - PKEs use exponentiation – involves multiplication and division
    - SEs use bit operations (add,XOR< substitute, shift)–much faster

# RSA Encryption (1)

- RSA = Rivest, Shamir, and Adelman (MIT), 1978
- **RSA** is one of the first practical public-key cryptosystems and is widely used for secure data transmission.
- Underlying hard problem:
  - Number theory – determining prime factors of a given (large) number (ex. factoring of small #: 5 → 5, 6 → 2 *3)
  - Arithmetic modulo n
- How secure is RSA?
  - So far remains secure (after all these years...)
  - Will quantum computing break it? TBD

# RSA Encryption (2)

- In RSA:

  P = E (D(P)) = D(E(P))  (order of D/E does not matter)

  - More precisely: $P = E(k_E, D(k_D, P)) = D(k_D, E(k_E, P))$

- Encryption: $C = P^e \bmod n$  $K_E = e$
  - Given C, it is very difficult to find P without knowing $K_D$

- Decryption: $P = C^d \bmod n$  $K_D = d$