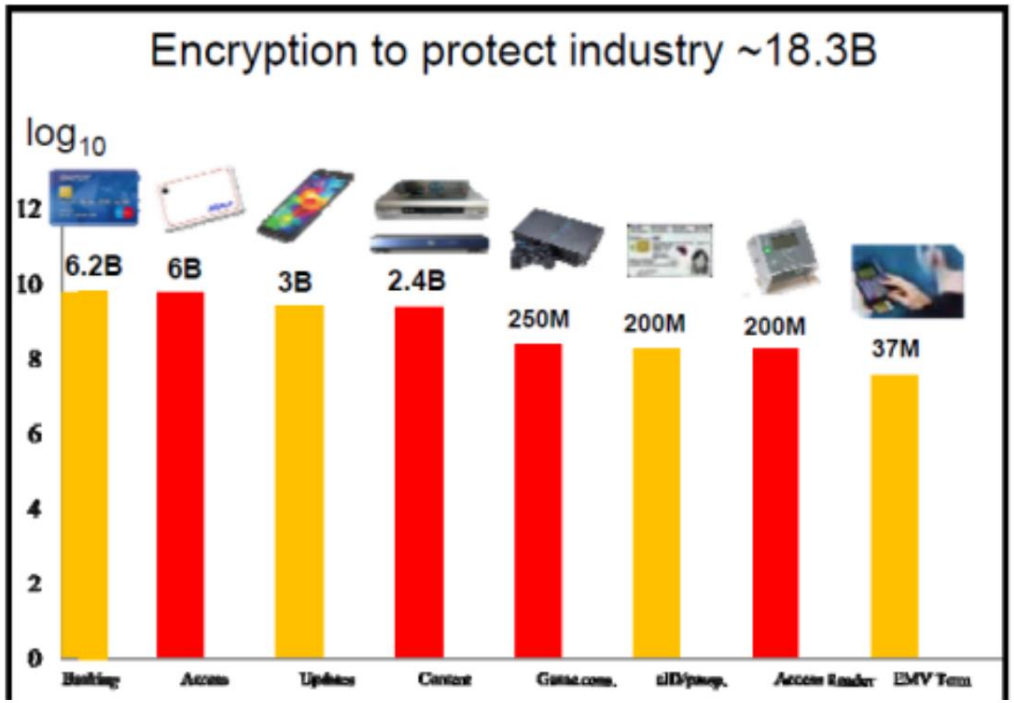
Introduction to Cryptography

Yu Bi

ELE594 – Special Topic on Hardware Security & Trust
University of Rhode Island

Cryptography Importance

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

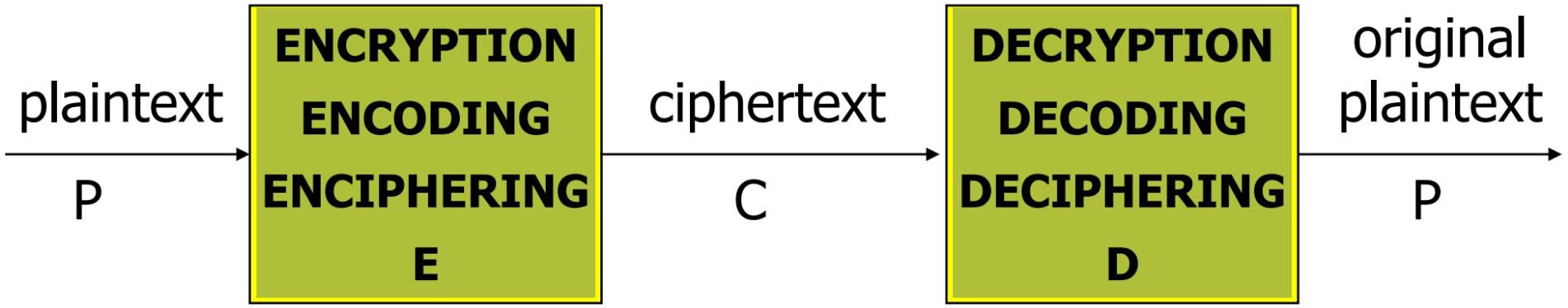


Cryptographic algorithms protects critical infrastructure and assets!

Terminology

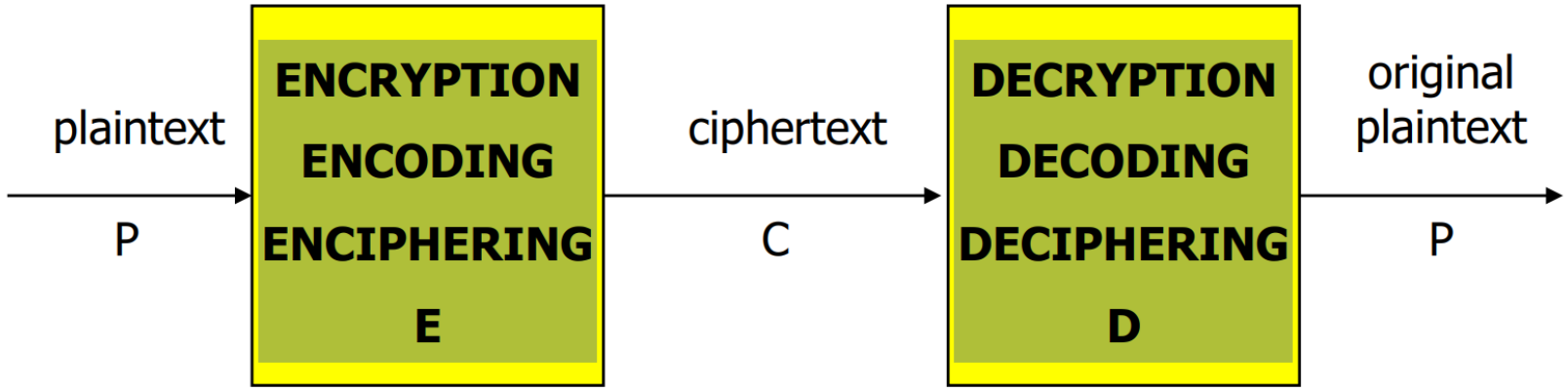
- **Cryptology:**
  - cryptography + cryptanalysis
- **Cryptography:**
  - art/science of keeping message secure
- **Cryptanalysis:**
  - art/science of breaking ciphertext
    - *Enigma* in world war II
      - Read the real story – not fabrications!

Basic Cryptography Scheme



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

Formal Notation



- C = E(P)
- P = D(C)

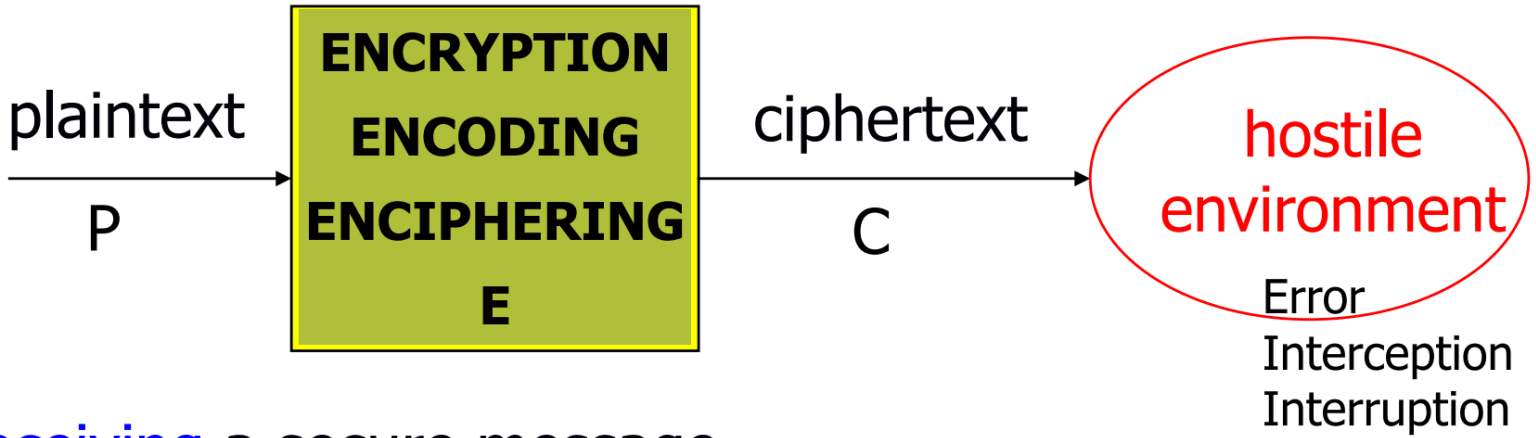
E – encryption rule/algorithm

D – decryption rule/algorithm

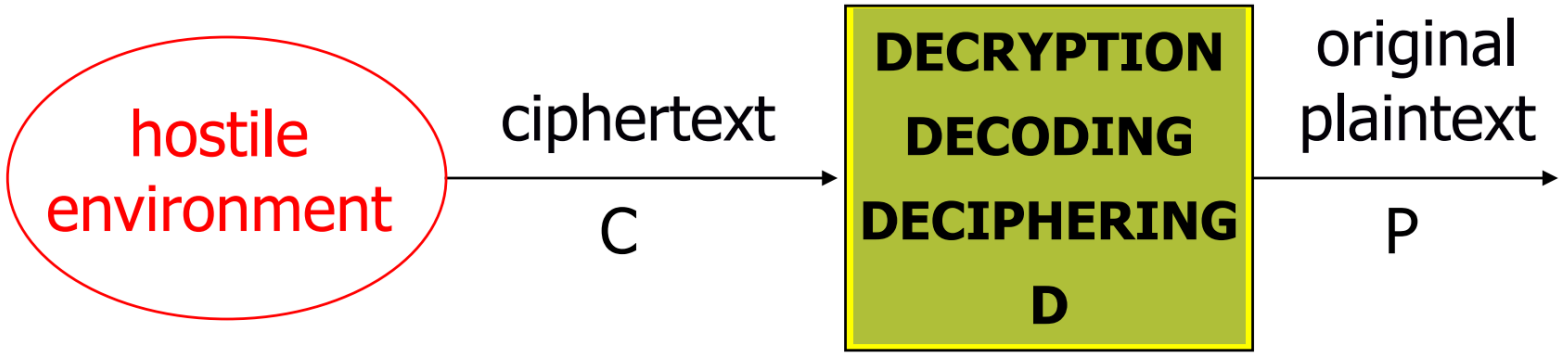
- We need a cryptosystem, where:
  - P = D(C)= D(E(P))
    - i.e., able to get the original message back

Cryptography in Practice

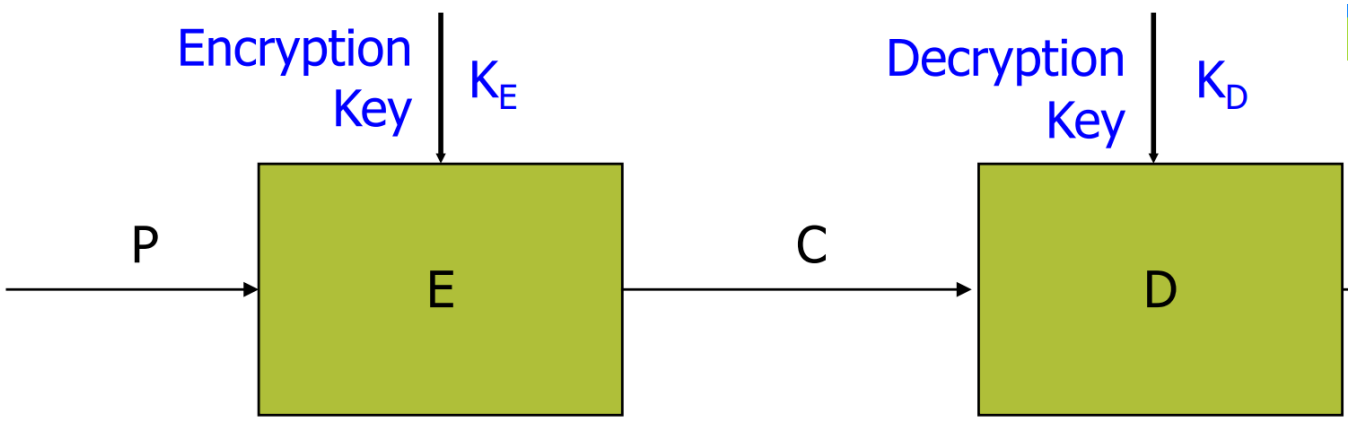
- Sending a secure message



- Receiving a secure message



Cryptography 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 a
- We need: $P = D(K_D, E(K_E, P))$

Key Classifications

- 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

Cryptanalysis

- **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

- **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!!!

Cryptanalysis

- **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

Cryptanalysis

- 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

Requirement 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 Cipher Types

- 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

Caeser Cipher

- $c_i = E(p_i) = p_i + 3 \bmod 26$ (*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 π: $\pi(i) = i+3 \bmod 26$

  $\pi(0)=3$, $\pi(1)=4$, …,

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

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

Caeser Cipher

- 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 Substitute Ciphers

- 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

1-grams (unigrams) in 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

Statistical Analysis

- 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

Statistical Analysis

- phi $\varphi(i)$ - correlation of frequency of letters *in ciphertext* with frequency of corresponding letters *in English* —for key i
- For key i: $\varphi(i) = \Sigma_{0 \le c \le 25} f(c) * p(c - i)$
  - *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

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

Statistical Analysis

- Correlation $\varphi(i)$ for $0 \leq i \leq 25$

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

Statistical Analysis

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

Caeser'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

Other Substitute Ciphers

n-char key:

- Polyalphabetic substitution ciphers
- Vigenere Tableaux cipher

Polyalphabetic Substitution

- 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
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

- **Question**:

  How Key1 and Key2 were defined?

[cf. J. Leiwo, VU, NL]

Polyalphabetic Substitution

- 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

  - 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)

Polyalphabetic Substitution

	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

Vigenere Tableaux

	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

Vigenere Tableaux

- 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

Vigenere Tableaux

- 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'

Transposition Ciphers

- 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

    (b) onto 2 columns:

    HE
    LL
    OW
    OR
    LD

  - Ciphertext (read column-by column):

    (a) **hlodeorxlwlx**

    (b) **hloolelwrd**

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

Transposition Ciphers

- 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)

Stream and Block Ciphers

a. Stream ciphers

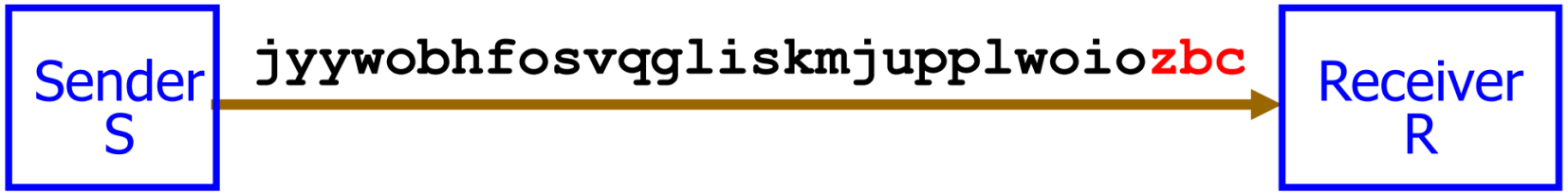
b. Problems with stream ciphers

c. Block ciphers

d. Pros / cons for stream and block ciphers

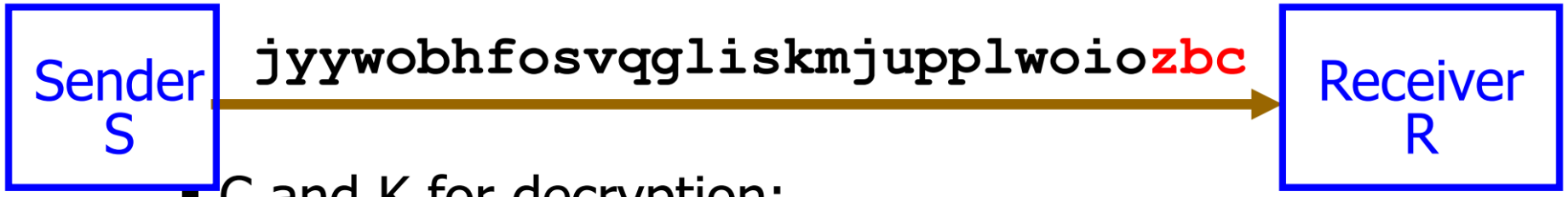
Stream Ciphers

- 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):



Stream Ciphers

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



  - 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. . .

Problem with Stream Ciphers

- 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

Problem with Stream Ciphers

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

- 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!

What if message is corrupted in a noisy area?

Problem with Stream Ciphers

- 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