

# IT 4823

## Information Security Administration

### Cryptography I



Notice: This session is  
being recorded.

# Codes and Ciphers

- Both codes and ciphers obfuscate communications to foil eavesdropping.
- **Codes** substitute words, phrases, or other symbols for other words. Codes operate at the level of meaning.
  - *One if by land and two if by sea.*
  - *Climb Mount Niitaka.*
- **Ciphers** operate at the level of symbols or small groups of symbols.

# Cryptography

- **Cryptography:** making secret ciphers
  - crypt (*kryptós*): hidden, secret
  - graph (*gráphein*): writing
- **Cryptanalysis:** analysis of cipher text (and other information) to attempt to derive plain text.
- **Cryptology:** the art and science of making and breaking secret ciphers. The union of cryptography and cryptanalysis.

# Goals of Cryptosystems

- Protection of **confidentiality**
- Protection of **integrity**
  - Message integrity
  - Non-repudiation (origin integrity)
  - Authentication (origin integrity)
- Do not address availability. (The official CISSP book says otherwise; it is wrong.)

# Plaintext and Ciphertext

The Cæsar Cipher:

**Now is the time!** Plaintext

**Opx jt uif ujn!** Ciphertext

# Cryptography = Algorithm + Key

**Algorithm:** A procedure for transforming plain text and a key to cipher text. The algorithm is generally assumed to be well-known.  
(Kerckhoffs' principle.)

**Key:** A secret, often random, collection of data that is combined with a plain text message to produce cipher text.

# Cryptosystem

Five-tuple  $(\mathcal{E}, \mathcal{D}, \mathcal{P}, \mathcal{K}, \mathcal{C})$

- $\mathcal{P}$  set of plaintexts
- $\mathcal{K}$  set of keys
- $\mathcal{C}$  set of ciphertexts
- $\mathcal{E}$  set of encryption functions  $e: \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}$
- $\mathcal{D}$  set of decryption functions  $d: \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{P}$

# Example

Example: Cæsar cipher

- $\mathcal{P} = \{ \text{sequences of letters} \}$
- $\mathcal{C} = \mathcal{P}$
- $\mathcal{K} = \{ i \mid i \text{ is an integer and } 0 < i \leq 25 \}$
- $\mathcal{E} = \{ E_k \mid k \in \mathcal{K} \text{ and for all letters } m, \\ E_k(m) = (m + k) \bmod 26 \}$
- $\mathcal{D} = \{ D_k \mid k \in \mathcal{K} \text{ and for all letters } c, \\ D_k(c) = (26 + c - k) \bmod 26 \}$

Cæsar cipher for  $k=3$

ABCDEFGHIJKLMNOPQRSTUVWXYZ

DEFGHIJKLMNOPQRSTUVWXYZABC



# Three Major Areas

- **Symmetric key (classical) cryptography:**  
Security depends upon a secret (the key) shared between sender and recipient
- **Cryptographic hash functions:** transform an input into a fixed-size output with properties that depend on the function.
- **Public key cryptography:** Depends upon a pair of keys, one public and one private

# Symmetric Key Cryptography

- Sender, receiver share common key (or the same principal.)
  - Keys may be the same, or trivial to derive from one another
  - Sometimes called *classical cryptography*
- Two basic types
  - Transposition ciphers
  - Substitution ciphers
  - Combinations are called *product ciphers*

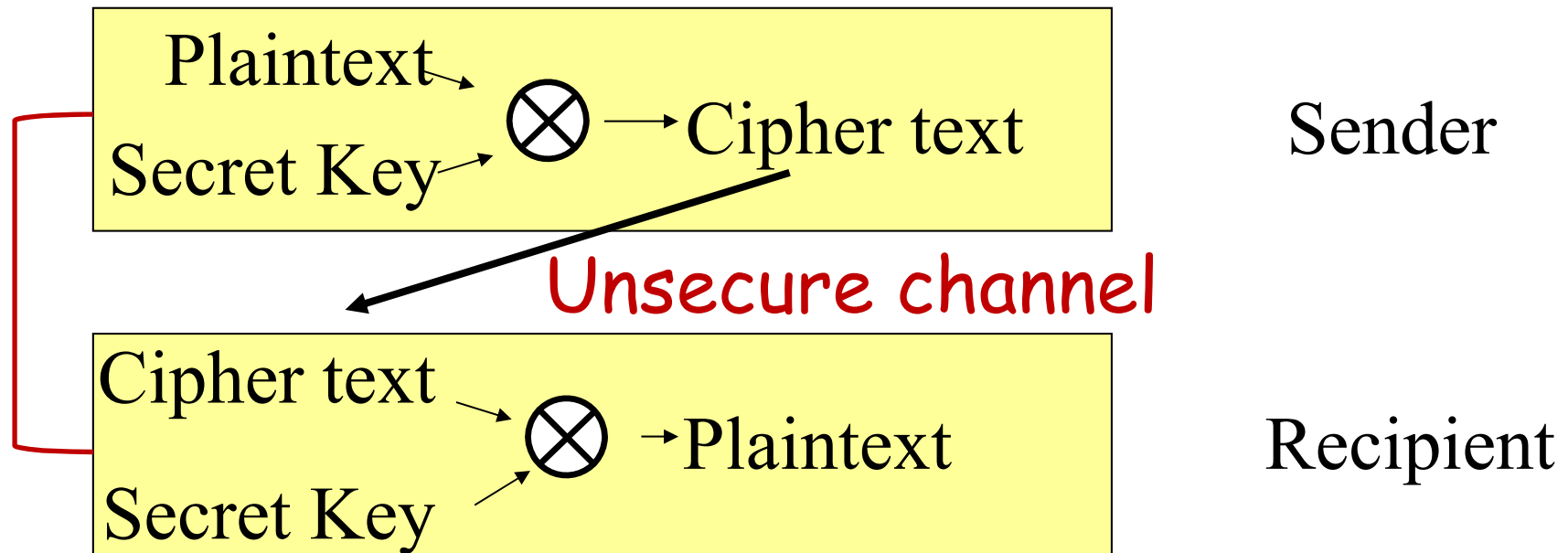
# Symmetric Key Cryptography

Encrypt by applying the key to the plaintext using an algorithm.

Decrypt by reversing the process using the *same* key and the inverse algorithm.

**Mnemonic:** *symmetric, shared key and secret key* are all the same thing and they all start with S.

# Symmetric Key Cryptography



The two “secret keys” are the same.

# Breakable Encryption

- A cryptosystem is “breakable” if, with enough time and information, it can theoretically be “cracked.”
- Either by determining the key or otherwise obtaining the plaintext.
- We must assume the algorithm is known. (Kerckhoffs’ Principle.)
- A cryptosystem that is breakable may require considerable effort. That is known as being “computationally secure.”

# Key Strength and Security

Encryption is a mathematical operation

*The strength is in the key, not the algorithm!*

(Assume that the bad guys know the algorithm.

That is *Kerckhoffs' Principle*.)

However, the algorithm must be free from “shortcut attacks.”

Keys should be *long* and *random*.

Keys are applied over blocks of data. Brute force attack: try all possible key combinations. 40-bit key:  $2^{40}$  combinations.

# Attacks Against Cryptography

- Opponent whose goal is to break cryptosystem is the *adversary*
  - Assume adversary knows the algorithm used, but not the key (Kerckhoffs' principle again)
- Three major types of attacks:
  - *ciphertext only*: adversary has only ciphertext; goal is to find plaintext and possibly the key
  - *known plaintext*: adversary has ciphertext, corresponding plaintext; goal is to find key
  - *chosen plaintext*: adversary may supply plaintext and obtain corresponding ciphertext; goal is to find key

# Computational vs. Absolute Security

- *Absolutely secure*: There is no way to recover the plaintext from the cipher text without the key. (And the algorithm)
- *Computationally secure*: The time to recover the plaintext from the cipher text is so great that the message has little or no value by the time it can be decrypted.

How long does “*We attack at dawn!*” need to remain secure?



# Computational Security

- Brute force: Try every possible key combination to see which one appears to decrypt the message.
- Shortcut attacks: For a particular algorithm, there exists a mechanism that is less work than brute force for extracting the plain text from the ciphertext. Example: letter frequency analysis in the Cæsar cipher.
- Cracking such a system could take thousands of years even though a mechanism is known.

# Representing Messages

- In computers, all data are encoded as binary numbers.
- We can perform mathematical transforms on these numbers.
- So, modern encryption is essentially a mathematical operation.
- We can easily perform transformations on ASCII.  
(Or Unicode.)

# Example

Message: DOG 010001000100011101001111

Key (simple) 010101010101010101010101

Algorithm: XOR

Cipher text 000100010001001000011010

Key (again) 010101010101010101010101

Plain text 010001000100011101001111

We've decoded the original text: "DOG" !

# The Cæsar Cipher

Now is the time! Plaintext

Opx jt uif ujnfn! Ciphertext

This is the Cæsar Cipher with a key of 1 (B).

ABCDEFGHIJKLMNOPQRSTUVWXYZ Plaintext

BCDEFGHIJKLMNOPQRSTUVWXYZA Ciphertext

Enciphering is performed by replacing each plaintext character with the character “one down” in the alphabet.

The Cæsar Cipher is a *substitution cipher*.

# Cæsar's Problem

- Key is too short:
  - Can be found by exhaustive search
  - Statistical frequencies not concealed
  - They look too much like regular English (or Latin!) words
- So make it longer...
  - Use multiple letters in key
  - The idea is to smooth the statistical frequencies to make cryptanalysis harder

# Vigenère Cipher

- Like Cæsar cipher, but use a phrase for the key
  - Example:
    - Message **THE BOY HAS THE BAG**
    - Key **BCD** (1, 2, and 3 letters down)
    - Encipher using Cæsar cipher for each letter:
- |        |                          |
|--------|--------------------------|
| key    | <b>BCDBCDBCDBCDBCDCD</b> |
| plain  | <b>THEBOYHASTHEBAG</b>   |
| cipher | <b>UJHCQBJCSUJHCCJ</b>   |
- The Vigenère cipher is a *polyalphabetic* substitution cipher.

# Cryptographic Terms

- *period*: length of key
  - In the “BCD” example, the period is 3
- *tableau*: table used to encipher and decipher
  - Vigenere cipher tableau has key letters on top, plaintext letters on the left
- *polyalphabetic*: the key has several different letters (Cæsar cipher is monoalphabetic)

# Vigenère Tableau

BCD B C D B C D B C D

# THEBOYHASTHEBAG

UJHCQBJCSUJHCCJ

		B	C	D
A		B	C	D
B		C	D	E
C		D	E	F
D		E	F	G
E		F	G	H
F		G	H	I
G		H	I	J
H		I	J	K
I		J	K	L
J		L	L	M
K		L	M	N
L		M	N	O
M		N	O	P

		B	C	D
N		O	P	Q
O		P	Q	R
P		Q	R	S
Q		R	S	T
R		S	T	U
S		T	U	V
T		U	V	W
U		V	W	Z
V		W	X	Y
W		X	Y	Z
X		Y	Z	A
Y		Z	A	B
Z		A	B	C



# One-Time Pad

- A cipher with a random key at least as long as the message.
- Provably unbreakable.
- Why? Any part of the ciphertext is equally likely to correspond to any plaintext.
- Example: HELLO produces ciphertext EQNVZ
- A brute force attack with infinite computing power determines that  $\text{EQNVZ} = \text{HELLO}$
- But it *also* determines that  $\text{EQNVZ} = \text{LATER}$
- Both possibilities, and every other five letter word, are equally probable!

# More About One-Time Pads

- Warning: keys *must* be random, or you can attack the cipher by trying to regenerate the key
- Approximations, such as using pseudorandom number generators to generate keys, are *not* random, but may be adequate in some cases.

# Vernam Cipher

- Punched paper tape with a sequence of non-repeating numbers at least as long as the message.
- The message and the key tape are combined mathematically. (Addition mod 26.)
- Combining cipher text with the (same) key tape decrypts the message.

# Product Ciphers

- A “product cipher” is a combination of transposition and substitution.
- Two applications of encryption actually do help
- Further applying ciphers serially does not necessarily make the encryption stronger.

# Shannon's Characteristics

- The amount of secrecy needed determines the amount of work that's appropriate.
- The key space and algorithm should be free of artificial constraints.
- Implementation should be as simple as possible.
- Errors in enciphering should not propagate
- Enciphering should not increase message size.

# “Trustworthy” Encryption Systems

- Based on sound mathematics
- Analyzed by experts
- Withstand the test of time

“It’s not as easy as it looks!”

# Secret-Key Encryption

- Provides confidentiality for the message.
- Provides authentication. (Assuming the key is really secret.)

# The Problem with Secret Keys

- You use secret-key cryptography to protect unsecure transmission or storage
- Problem: how does the recipient get the secret key!?!

## Key Exchange Problem

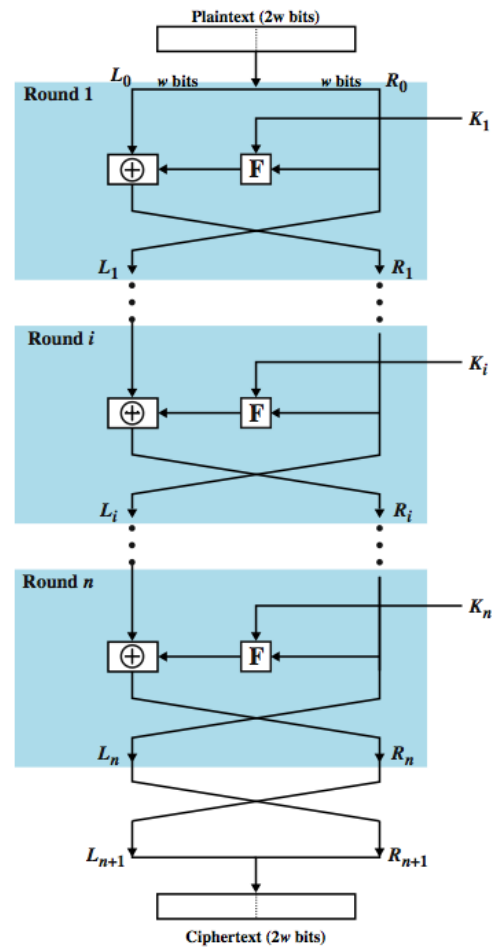
(Out of band transmission is one secure way, but often one could simply transmit the message, and not just the key, if a secure channel exists. )



# Weaknesses

- Bribery, coercion, etc.
- Release of plaintext
- Error and carelessness
- Painstaking analysis

# Feistel Cipher Structure



*Illustration Copyright © 2008 by Prentice-Hall*

# Block Cipher Structure

- General iterative block cipher structure
  - with a sequence of rounds
  - with substitutions / permutations controlled by key
- Parameters and design features:
  - block size
  - key size
  - number of rounds
  - subkey generation algorithm
  - round function
  - also: fast software encrypt/decrypt
  - ease of analysis

# Triple DES (3DES)

- First used in financial applications
- In DES FIPS PUB 46-3 standard of 1999
- Uses three keys & three DES executions:
$$C = E(K_3, D(K_2, E(K_1, P)))$$
- Decryption same with keys reversed
- Use of decryption in second stage gives compatibility with original DES users
- Effective 112-bit key length, slow but secure
- AES has replaced 3DES

# Stream Ciphers

- Process input elements continuously
- Key input to a pseudorandom bit generator
  - produces stream of random-like numbers
  - unpredictable without knowing input key
  - XOR keystream output with plaintext bytes
- Stream ciphers are faster and use far less code
- Design considerations:
  - encryption sequence should have a large period
  - keystream approximates random number properties
  - uses a sufficiently long key

# Symmetric (Secret Key) Encryption

- DES 56 bits (1973) **Obsolete!**
- 3DES 3 keys, multiple encryption (strength equivalent to 112 bits)
- AES (Rijndael) 128 (and more) bit keys (2001)
- Others

# The Advanced Encryption Standard

- DES is too weak for use now
- NIST selected Rijndael as Advanced Encryption Standard, successor to DES effective in May, 2002
- Designed to withstand attacks that were successful on DES

# Key Distribution

- Symmetric cryptography needs a shared key:
- Two parties, A and B can achieve this by:
  - A selects key, physically delivers to B
  - Third party select keys, physically delivers to A and B; reasonable for link encryption; does not scale well.
  - A selects new key, sends encrypted using previous old key to B; good for either, but security fails if any key discovered
  - Third party C selects key, sends encrypted to each of A and B using existing key with each
  - Distribution using public key cryptography



# Notation

- $X \rightarrow Y : \{ Z \parallel W \} k_{X,Y}$ 
  - $X$  sends  $Y$  the message produced by concatenating  $Z$  and  $W$  enciphered by key  $k_{X,Y}$ , which is shared by users  $X$  and  $Y$
- $A \rightarrow T : \{ Z \} k_A \parallel \{ W \} k_{A,T}$ 
  - $A$  sends  $T$  a message consisting of the concatenation of  $Z$  enciphered using  $k_A$ ,  $A$ 's key, and  $W$  enciphered using  $k_{A,T}$ , the key shared by  $A$  and  $T$
- $r_1, r_2$  nonces (nonrepeating random numbers)

# Goal: Alice and Bill Get Shared Key

- Key cannot be sent in clear
  - Attacker can listen in
  - Key can be sent enciphered, or derived from exchanged data plus data that can't be determined by an eavesdropper
- Alice, Bill may trust third party
- Assume all cryptosystems and protocols publicly known (Kerchoff's principle)
  - The only secret data is the keys and any ancillary information known only to Alice and Bill, needed to derive keys
  - Anything transmitted is assumed known to attacker

# Classical Key Exchange

- Bootstrap problem: how do Alice, Bill begin? Alice can't send it to Bill in the clear!
- Assume trusted third party, Cathy
  - Alice and Cathy share secret key  $k_A$
  - Bill and Cathy share secret key  $k_B$
- Use this to exchange shared key  $k_s$

# Simple Protocol

Alice  $\xrightarrow{\{ \text{request for session key for Bill} \} k_{A,C}}$  Cathy

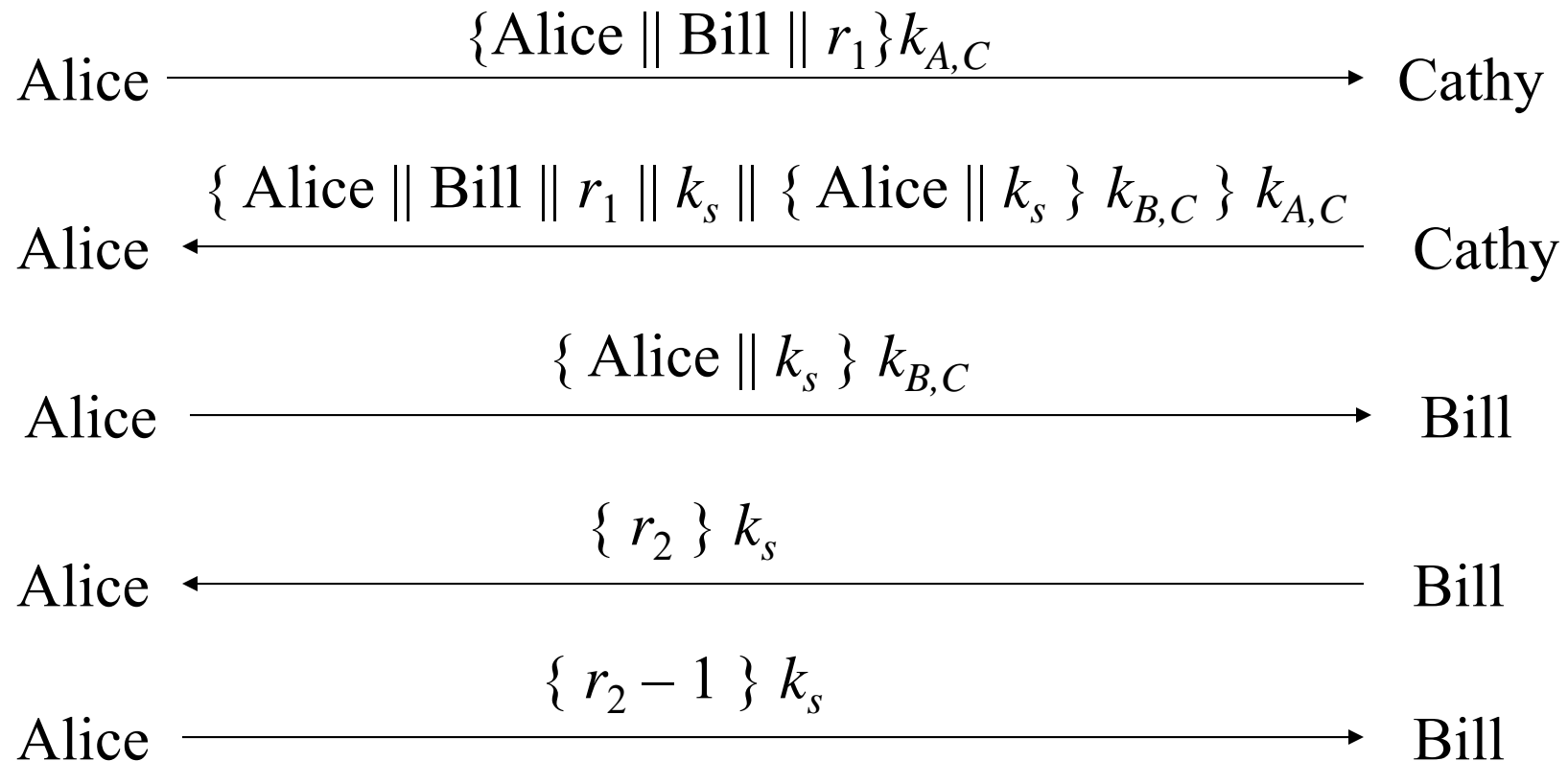
Alice  $\xleftarrow{\{ k_s \} k_{A,C} \parallel \{ k_s \} k_{B,C}}$  Cathy

Alice  $\xrightarrow{\{ k_s \} k_{B,C}}$  Bill

# Problems

- How does Bill know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bill, later replays it; Bill may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bill, so Bill re-uses session key
- Protocols must provide authentication and defense against replay

# Needham-Schroeder



# Argument: Alice talking to Bill

- Second message
  - Enciphered using key only she and Cathy know
    - So Cathy enciphered it
  - Response to first message
    - As  $r_1$  in it matches  $r_1$  in first message
- Third message
  - Alice knows only Bill can read it as only Bill can derive session key from message
  - Any messages enciphered with that key are from Bill

# Argument: Bill talking to Alice

- Third message
  - Enciphered using key only he and Cathy know
    - So Cathy enciphered it
  - Names Alice, session key
    - Cathy provided session key, says Alice is other party
- Fourth message
  - Uses session key to determine if it is replay from Eve
    - If not, Alice will respond correctly in fifth message
    - If so, Eve can't decipher  $r_2$  and so can't respond, or responds incorrectly



# Questions

