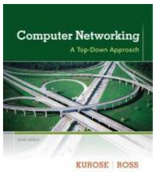
# Chapter 8 Security

A note on the use of these ppt slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:
- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR



The course notes are adapted for CSCI 363 at Bucknell
Spring 2016, Xiannong Meng




Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012



## Chapter 8: Network Security

*Chapter goals:*
- understand principles of network security:
  - cryptography and its *many* uses beyond "confidentiality"
  - authentication
  - message integrity
- security in practice:
  - firewalls and intrusion detection systems
  - security in application, transport, network, link layers



## Chapter 8 roadmap





## What is network security?

*confidentiality:* only sender, intended receiver should "understand" message contents
- sender encrypts message
- receiver decrypts message

*authentication:* sender, receiver want to confirm identity of each other

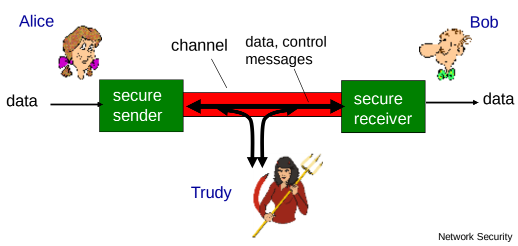
*message integrity:* sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

*access and availability:* services must be accessible and available to users



## Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages






## Who might Bob, Alice be?

- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

## There are bad guys (and girls) out there!

*Q:* What can a "bad guy" do?

*A:* A lot! See section 1.6

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service:* prevent service from being used by others (e.g., by overloading resources)

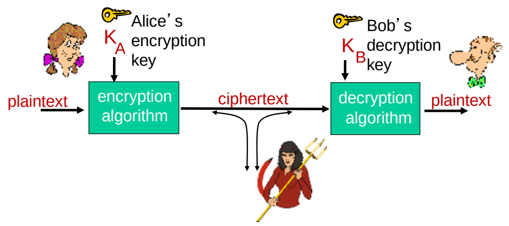


## Chapter 8 roadmap





## The language of cryptography




m plaintext message

$K_A(m)$ ciphertext, encrypted with key $K_A$

$m = K_B(K_A(m))$

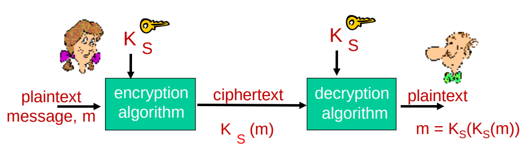


## Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,b
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext



## Symmetric key cryptography




symmetric key crypto: Bob and Alice share same (symmetric) key: $K_S$

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

*Q:* how do Bob and Alice agree on key value?



## Simple encryption scheme

*substitution cipher:* substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

e.g.:
```
 Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

*Encryption key:* mapping from set of 26 letters to set of 26 letters

## A more sophisticated encryption approach

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- cycling pattern:
  - e.g., n=4: $M_1,M_3,M_4,M_3,M_2$; $M_1,M_3,M_4,M_3,M_2$; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - bob: b using $M_1$, o using $M_3$, b using $M_4$
  - b -> 'k', o -> 'm', b -> 'y'

*Encryption key:* n substitution ciphers, and cyclic pattern
- key need not be just n-bit pattern



## Symmetric key crypto: DES

DES: Data Encryption Standard
- US encryption standard, published 1975, originally adopted 1976, reaffirmed 1983, 1988, 1993, 1999, withdrew 2004
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day (2008)
    - http://en.wikipedia.org/wiki/Data_Encryption_Standard
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys



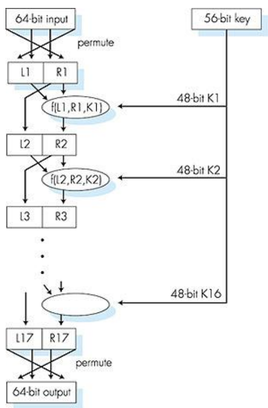
## Symmetric key crypto: DES

*DES operation*

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation






## AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
  - 10 cycles for128 bit keys
  - 12 cycles for 192 bit keys
  - 14 cycles for 256 bit keys
- brute force decryption (try each key) takes 1 sec on DES, would take149 trillion years for AES
- See http://en.wikipedia.org/wiki/Advanced_Encryption_Standard



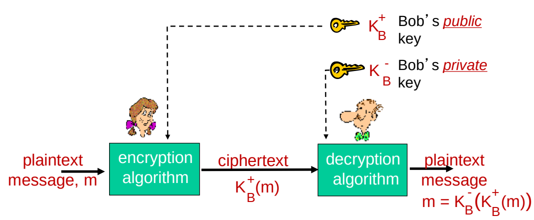
## How to agree on "key(s)"?

*symmetric key crypto*
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

*public key crypto*
- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver



## Public key cryptography



## Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that
   $K_B^-(K_B^+(m)) = m$
2. given public key $K_B^+$, it should be impossible to compute private key $K_B^-$

*RSA:* Rivest, Shamir, Adelson algorithm



## Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:
  [(a mod n) + (b mod n)] mod n = (a+b) mod n
  [(a mod n) - (b mod n)] mod n = (a-b) mod n
  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus
  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$



## RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number.

*example:*

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).



## RSA: Creating public/private key pair

1. choose two large prime numbers *p, q.* (e.g., 1024 bits each)
2. compute *n = pq, z = (p-1)(q-1)*
3. choose *e* (with e<*n)* that has no common factors with z (*e, z* are "relatively prime").
4. choose *d* such that *ed-1* is exactly divisible by z. (in other words: *ed* mod *z = 1* ).
5. *public* key is *(n,e)*. *private* key is *(n,d)*.
   $(n,e) = K_B^+ \qquad (n,d) = K_B^-$



## RSA: encryption, decryption

0. given (*n,e*) and (*n,d*) as computed above
1. to encrypt message *m* (<*n*), compute
   $c = m^e \bmod n$
2. to decrypt received bit pattern, *c*, compute
   $m = c^d \bmod n$

*magic happens!* $m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$

*See a proof later.*



## RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.
*e=5* (so *e, z* relatively prime).
*d=29* (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.

| | bit pattern | m | $m^e$ | $c = m^e \bmod n$ |
|---|---|---|---|---|
| encrypt letter 'L' or value 12: | 00001100 | 12 | 248832 | 17 |

| | c | $c^d$ | $m = c^d \bmod n$ |
|---|---|---|---|
| decrypt: | 17 | 481968572106750915091411825223071697 | 12 |

## RSA implementation in openssl

- At the Linux command prompt, where openssl has been installed
  - To generate private key
    - openssl genrsa -out private.pem 2048
  - To generate public key based on the private key
    - openssl rsa -in private.pem -outform PEM -pubout -out public.pem
- PEM format
  - The PEM format is the most common format that Certificate Authorities issue certificates in.
  - They are Base64 encoded ASCII files.



## Why does RSA work?

- must show that $c^d \bmod n = m$ where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where n= pq and z = (p-1)(q-1)
  - Fact 4 in earlier set : ed mod z = 1
- thus,

$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(ed \bmod z)} \bmod n \\ &= m^1 \bmod n \\ &= m \end{aligned}$$



## RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

use private key first, followed by public key

*result is the same!*



## Why $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$ ?

follows directly from modular arithmetic:

$$\begin{aligned} (m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n \end{aligned}$$



## Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard



## RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key cryto to establish secure connection, then establish second key – symmetric session key – for encrypting data

*session key, $K_S$*

- Bob and Alice use RSA to exchange a symmetric key $K_S$
- once both have $K_S$, they use symmetric key cryptography