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

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity, authentication
8.4 Securing e-mail
8.5 Securing TCP connections: SSL
8.6 Network layer security: IPsec
8.7 Securing wireless LANs
8.8 Operational security: firewalls and IDS

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
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

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity, authentication
8.4 Securing e-mail
8.5 Securing TCP connections: SSL
8.6 Network layer security: IPsec
8.7 Securing wireless LANs
8.8 Operational security: firewalls and IDS

The language of cryptography

plaintext → encryption algorithm → ciphertext → decryption algorithm → plaintext

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
- known-plaintext attack:
  Trudy has plaintext corresponding to ciphertext
  e.g., in monoalphabetic cipher, Trudy determines pairings for a,i,c,e,b,o,b
- chosen-plaintext attack:
  Trudy can get ciphertext for chosen plaintext

Symmetric key cryptography

plaintext message, m

plaintext → encryption algorithm → ciphertext → decryption algorithm → plaintext

symmetric key crypto: Bob and Alice share same (symmetric) key: $K_S$
- e.g., key is knowing substitution pattern in monoalphabetic 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: mnbindzrftghjklpoqwsx
- 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, ..., M_n
- cycling pattern:
  - e.g., n=4: M_1, M_3, M_2, M_1, M_3, M_4, M_1, M_3, ...
- 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
  - 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)
    - no known good analytic attack
    - making DES more secure:
      - 3DES: encrypt 3 times with 3 different keys

AES: Advanced Encryption Standard
- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
  - 10 cycles for 128 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 take 149 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 \( K_B^+() \) and \( K_B^-() \) 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

- \( x \mod n = \text{remainder of } x \text{ when divide by } n \)
- facts:
  - \( [(a \mod n) + (b \mod n)] \mod n = (a + b) \mod n \)
  - \( [(a \mod n) - (b \mod n)] \mod n = (a - b) \mod n \)
  - \( [(a \mod n) \times (b \mod n)] \mod n = (a \times b) \mod n \)
- thus \( (a \mod n)^d \mod n = a^d \mod n \)
- example: \( x=14, n=10, d=2: (x \mod n)^2 \mod n = 4^2 \mod 10 = 6 \)

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: encryption, decryption

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

\[ m = (m^e \mod n)^d \mod n \]

magic happens!

See a proof later.

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 \( ed \mod z = 1 \) (in other words: \( ed \mod z = 1 \)).
5. public key is \((n,e)\), private key is \((n,d)\).

RSA example:

Bob chooses \( p=5, q=7 \). Then \( n=35, z=24 \).
\( e=5 \) (so \( e, z \) relatively prime).
\( d=29 \) (so \( ed \mod z = 1 \)).

encrypting 8-bit messages.

<table>
<thead>
<tr>
<th>encrypt letter 'L'</th>
<th>bit pattern</th>
<th>( m )</th>
<th>( m^e \mod n )</th>
<th>( c )</th>
<th>( c^d \mod n )</th>
<th>( m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001100</td>
<td>12</td>
<td>248832</td>
<td>17</td>
<td>12</td>
<td>48166870190769159411816253871897</td>
<td>12</td>
</tr>
</tbody>
</table>

decrypt:

<table>
<thead>
<tr>
<th>decrypt</th>
<th>( c )</th>
<th>( c^d \mod n )</th>
<th>( m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>48166870190769159411816253871897</td>
<td>17</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

4/25/2016
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 \( c^d \mod n = m \)
  - where \( c = m^e \mod n \)
- fact: for any \( x \) and \( y \):
  - \( x^{(y \mod z)} \mod n = x^{(y \mod z)} \mod n \)
  - where \( n = pq \) and \( z = (p-1)(q-1) \)
  - Fact 4 in earlier set: \( ed \mod n = 1 \)

- thus,
  - \( c^d \mod n = (m^e \mod n)^d \mod n = m^{ed} \mod n = m^{ed \mod z} \mod n = m^1 \mod n = m \)

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, followed by public key

\[ \text{result is the same!} \]

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 crypto 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