7. Network Security
Chapter 7: 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 7: Outline

- What is Network Security?
- Principles of Cryptography
- Authentication, Message Integrity
- Securing TCP: SSL/TLS
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
• Many other examples!
There are bad guys (and girls) out there!

**Q:** What can a “bad guy” do?

**A:** A lot!

- **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 7: Outline

- What is Network Security?
- Principles of Cryptography
- Authentication, Message Integrity
- Securing TCP: SSL/TLS
The language of cryptography

plaintext \rightarrow \text{encryption algorithm} \rightarrow \text{ciphertext} \rightarrow \text{decryption algorithm} \rightarrow \text{plaintext}

- **m** plaintext message
- **E(K_A,m)** ciphertext, encrypted with key **K_A**
- **m = D(K_B,E(K_A,m))**
Symmetric key cryptography

plaintext message, \( m \) \( \rightarrow \) encryption algorithm \( \rightarrow \) ciphertext \( E(K_S,m) \) \( \rightarrow \) decryption algorithm \( \rightarrow \) plaintext \( m = D(K_S,E(K_S,m)) \)

**symmetric key crypto**: Bob and Alice share same (symmetric) key: \( K_S \)
Symmetric key crypto: DES

DES: Data Encryption Standard

• US encryption standard [NIST 1993]
• 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
    • 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
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES
Public Key Cryptography

**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

plaintext message, $m$

encryption algorithm

$ciphertext$ $E(K_B^+, m)$

decryption algorithm

$m = D(K_B^-, E(K_B^+, m))$

Bob's $public$ key

$K_B^+$

Bob's $private$ key

$K_B^-$
Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that
   
   $$D(K_B^-, E(K_B^+, m)) = m$$

2. given public key $K_B^+$, it should be impossible* to compute private key $K_B^-$

*RSA: Rivest, Shamir, Adelson algorithm
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 \( n = pq, \quad z = (p-1)(q-1) \)

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 \( (n,d) \).
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!*
RSA example:


- $e=5$ (so $e$, $z$ relatively prime).
- $d=29$ (so $ed-1$ exactly divisible by $z$).

Encrypting 8-bit messages.

<table>
<thead>
<tr>
<th>encrypt:</th>
<th>bit pattern</th>
<th>$m$</th>
<th>$m^e$</th>
<th>$c = m^e \mod n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00001000</td>
<td>12</td>
<td>24832</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>decrypt:</th>
<th>$c$</th>
<th>$c^d$</th>
<th>$m = c^d \mod n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>

$m = 00001000$, $c = 17$, $m^e = 24832$.
Why does RSA work?

• must show that \( c^d \mod n = m \) where \( c = m^e \mod n \)

• fact: for any \( x \) and \( y \): \( x^y \mod n = x^{(y \mod z)} \mod n \)
  • where \( n = pq \) and \( z = (p-1)(q-1) \)

• 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 first, followed by public key

result is the same!
Why \( K_B^{-1}(K_B^+(m)) = m = K_B^+(K_B^{-1}(m)) \) ?

follows directly from modular arithmetic:

\[
(m^e \bmod n)^d \bmod n = m^{ed} \bmod n \\
= m^{de} \bmod n \\
= (m^d \bmod n)^e \bmod n
\]
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
Chapter 7: Outline

- What is Network Security?
- Principles of Cryptography
- Authentication, Message Integrity
- Securing TCP: SSL/TLS
Authentication

**Goal:** Bob wants Alice to “prove” her identity to him

**Protocol ap1.0:** Alice says “I am Alice”

“I am Alice” →  

Failure scenario??
**Authentication**

**Goal:** Bob wants Alice to “prove” her identity to him

**Protocol ap1.0:** Alice says “I am Alice”

in a network, Bob can not “see” Alice, so Trudy simply declares herself to be Alice
Authentication: another try

**Protocol ap2.0:** Alice says “I am Alice” in an IP packet containing her source IP address

<table>
<thead>
<tr>
<th>Alice’s IP address</th>
<th>“I am Alice”</th>
</tr>
</thead>
</table>

Failure scenario??
Authentication: another try

*Protocol ap2.0:* Alice says “I am Alice” in an IP packet containing her source IP address

Trudy can create a packet “spoofing” Alice’s address
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.

Failure scenario??
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.

Playback attack: Trudy records Alice’s packet and later plays it back to Bob.
Authentication: yet another try

*Protocol ap3.1:* Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.

- Alice’s IP addr
- Encrypted password
- “I’m Alice”

Failure scenario??
Authentication: yet another try

*Protocol ap3.1:* Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.

Record and playback *still* works!
**Goal:** avoid playback attack

**nonce:** number (R) used only *once-in-a-lifetime*

**ap4.0:** to prove Alice “live”, Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key

Failures, drawbacks?
Authentication: ap5.0

ap4.0 requires shared symmetric key
• can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography

“I am Alice”
“send me your public key”
$K_A$
$E(K_A^+, R)$
$R$
ap5.0: security hole

*man (or woman) in the middle attack:* Trudy poses as Alice (to Bob) and as Bob (to Alice)
ap5.0: security hole

\textit{man (or woman) in the middle attack}: Trudy poses as Alice (to Bob) and as Bob (to Alice)

difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!
Digital signatures

cryptographic technique analogous to hand-written signatures:

• sender (Bob) digitally signs document, establishing he is document owner/creator.

• *verifiable, nonforgeable*: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
simple digital signature for message $m$:

- Bob signs $m$ by signing with his private key $K_B$, creating signed message, $K_B(m)$

Dear Alice
Oh, how I have missed you. I think of you all the time! …(blah blah blah)
Bob

Bob’s message, $m$

$K_B$ Bob’s private key

Public key signature algorithm

$m, S(K_B^-, m)$

Bob’s message, m, signed with his private key
Digital signatures

- suppose Alice receives msg m, with signature: m, S(K_B^-,m)
- Alice verifies m signed by Bob by using Bob’s public key K_B^+
to verify V(K_B^+,S(K_B^-,m),m) = True

Alice thus verifies that:
- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:
✓ Alice can take m, and signature K_B(m) to court and prove that Bob signed m
Chapter 7: Outline

- What is Network Security?
- Principles of Cryptography
- Authentication, Message Integrity
- Securing TCP: SSL/TLS
SSL/TLS: Transport Layer Security

- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over TLS
- mechanisms: [Woo 1994], implementation: Netscape
- Current version: TLS1.2
  - TLS1.3 (aka TLS2 aka TLS4 aka TLS7 on the horizon)
- provides
  - confidentiality
  - integrity
  - authentication
- original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- available to all TCP applications
  - secure socket interface
TLS and TCP/IP

- TLS provides application programming interface (API) to applications
- C and Java TLS libraries/classes readily available
Chapter 7: Outline

✓ What is Network Security?
✓ Principles of Cryptography
✓ Authentication, Message Integrity
✓ Securing TCP: SSL/TLS
Network Security (summary)

basic techniques......

• cryptography (symmetric and public)
• message integrity
• end-point authentication

.... used in many different security scenarios

• secure email
• secure transport (SSL)

operational security: firewalls and IDS