Chapter 8
Security

A note on the use of these Powerpoint 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

All material copyright 1996-2016
© J.F Kurose and K.W. Ross, All Rights Reserved
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
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

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 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,
- **chosen-plaintext attack:** Trudy can get ciphertext for chosen plaintext
Symmetric key cryptography

plaintext message, m → encryption algorithm → ciphertext K_S(m) → decryption algorithm → plaintext m = K_S(K_S(m))

**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
  - dog: d from $M_1$, o from $M_3$, g from $M_4$

**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 [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

$ciphertext = K_B^+(m)$

decryption algorithm

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

Bob’s public key

Bob’s private key
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) \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)^d \mod n = 4^2 \mod 10 = 6
  
  x^d = 14^2 = 196, \quad x^d \mod 10 = 6
  
Security 8-20
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 $(n, d)$.

\[
\begin{align*}
K_B^+ \\
K_B^-
\end{align*}
\]
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
\[ m = c^d \mod n \]

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

\textit{magic happens!}
RSA example:


$e=5$ (so $e$, $z$ relatively prime).

$d=29$ (so $ed-1$ exactly divisible by $z$).

encrypting 8-bit messages.

Encrypt:

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

Decrypt:

$c = 17$

$m = c^d \mod n$

$m = 12$

$481968572106750915091411825223071697$
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:

\[ K_B^{-1}(K_B^+(m)) = m = K_B^+(K_B^-(m)) \]

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:

$$(m^e \mod n)^d \mod n = m^{ed} \mod n$$

$$= m^{de} \mod n$$

$$= (m^d \mod n)^e \mod n$$
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 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
This discussion of authentication protocols is actually 8.4 in the book, and differs slightly. So for the remaining sections (8.4-8.8) according to the slides, be aware this is actually 8.5 and on!

In the interest of minimizing edits, for now I've left the numbering the way Kurose and Ross (the book authors) prepared the slides.
Authentication

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

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

Failure scenario??
Authentication

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

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

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

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??

Authentication: another try
**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?

Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!
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”
```

```
Bob computes
K^+(K^-_A(R)) = R
```

and knows only Alice could have the private key, that encrypted R such that

```
K^+_A(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)

I am Alice

\[ R \]

Send me your public key

\[ K^-_A(R) \]

I am Alice

Send me your public key

\[ K^+_A \]

Trudy gets

\[ m = K^-_A(K^+_A(m)) \]

sends m to Alice encrypted with Alice’s public key

\[ K^+_T(m) \]
ap5.0: security hole

**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!
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
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
Digital signatures

simple digital signature for message m:

- Bob signs m by encrypting 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

Public key encryption algorithm

Bob’s message, m, signed (encrypted) with his private key

$m, K_B^{-1}(m)$
Digital signatures

- Suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob’s public key $K_B$ to $K_B^-(m)$ then checks $K_B(K_B^+(m^-)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob’s private key.

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
Message digests

Computationally expensive to public-key-encrypt long messages.

**Goal:** fixed-length, easy-to-compute digital "fingerprint"

- Apply hash function $H$ to $m$, get fixed size message digest, $H(m)$.

Hash function properties:
- Many-to-1
- Produces fixed-size msg digest (fingerprint)
- Given message digest $x$, computationally infeasible to find $m$ such that $x = H(m)$.
Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:
- produces fixed length digest (16-bit sum) of message
- is many-to-one

But given message with given hash value, it is easy to find another message with same hash value:

<table>
<thead>
<tr>
<th>message</th>
<th>ASCII format</th>
</tr>
</thead>
<tbody>
<tr>
<td>I O U 1</td>
<td>49 4F 55 31</td>
</tr>
<tr>
<td>0 0 . 9</td>
<td>30 30 2E 39</td>
</tr>
<tr>
<td>9 B O B</td>
<td>39 42 D2 42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>message</th>
<th>ASCII format</th>
</tr>
</thead>
<tbody>
<tr>
<td>I O U 9</td>
<td>49 4F 55 39</td>
</tr>
<tr>
<td>0 0 . 1</td>
<td>30 30 2E 31</td>
</tr>
<tr>
<td>9 B O B</td>
<td>39 42 D2 42</td>
</tr>
</tbody>
</table>

B2 C1 D2 AC  different messages  but identical checksums!  B2 C1 D2 AC
Digital signature = signed message digest

Bob sends digitally signed message:

large message $m$ $\xrightarrow{H: \text{Hash function}}$ $H(m)$

$\xrightarrow{\text{digital signature (encrypt)}}$ $K^{-}_B(H(m))$

encrypted msg digest $K^{-}_B(H(m))$

Bob’s private key $K^{-}_B$

Alice verifies signature, integrity of digitally signed message:

large message $m$ $\xrightarrow{H: \text{Hash function}}$ $H(m)$

$\xrightarrow{\text{digital signature (decrypt)}}$ $K^{+}_B$

Bob’s public key $K^{+}_B$

digital signature $H(m)$

equal? $H(m)$ $\xrightarrow{\text{encrypted msg digest}}$ $K^{-}_B(H(m))$
Hash function algorithms

- **MD5 hash function widely used (RFC 1321)**
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x

- **SHA-1 is also used**
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest
Recall: ap5.0 security hole

**man (or woman) in the middle attack:** Trudy poses as Alice (to Bob) and as Bob (to Alice)

I am Alice

Send me your public key

Send me your public key

Trudy gets

m = \( K_A^- (K_A^+ (m)) \)
sends m to Alice encrypted with Alice’s public key
Public-key certification

- motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    *Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob*
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it’s Bob’s public key
  - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
  - Bob doesn’t even like pepperoni

Security 8-53
Certification authorities

- **certification authority (CA):** binds public key to particular entity, E.

- E (person, router) registers its public key with CA.
  - E provides “proof of identity” to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E’s public key digitally signed by CA – CA says “this is E’s public key”
when Alice wants Bob’s public key:
  • gets Bob’s certificate (Bob or elsewhere).
  • apply CA’s public key to Bob’s certificate, get Bob’s public key