Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Authentication
8.4 Integrity
8.5 Key Distribution and certification

1

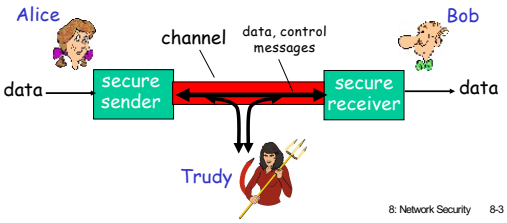
What is network security?

1. Confidentiality:
   only Tx and Rx should "understand" message contents
   - Tx encrypts message
   - Rx decrypts message
2. Authentication:
   sender, receiver want to confirm identity of each other
3. Message Integrity:
   sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
4. Access and Availability:
   services must be accessible and available to users

2

Friends and enemies: Alice, Bob, Trudy

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



3

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
- Smartphone and smartwatches
- Amazon Echo talking to WiFi base station

4

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)

more on this later ……

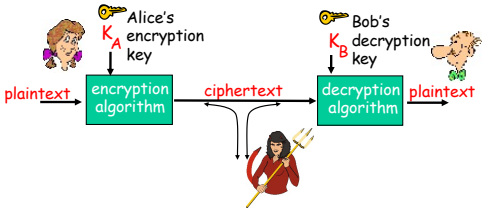
5

Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Authentication
8.4 Integrity
8.5 Key Distribution and certification
8.6 Access control: firewalls
8.7 Attacks and counter measures
8.8 Security in many layers

6

The language of cryptography



symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

7

Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

E.g.: `Plaintext:` [redacted]

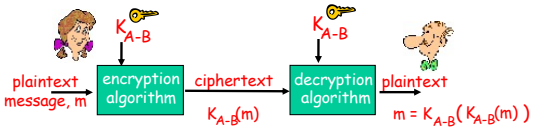
`ciphertext: nkn. s gktc wky. mgsbc`

Q: How hard to break this simple cipher?:

- brute force (how hard?)
- other?

8

Symmetric key cryptography



symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{A-B}

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: how do Bob and Alice agree on key value?

9

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase ("Strong cryptography makes the world a safer place") decrypted (brute force) in 4 months
  - no known "backdoor" decryption approach
- making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

10

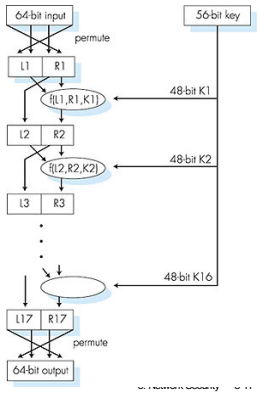
Symmetric key crypto: DES

DES operation

initial permutation

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

final permutation



11

AES: Advanced Encryption Standard

- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- 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

12

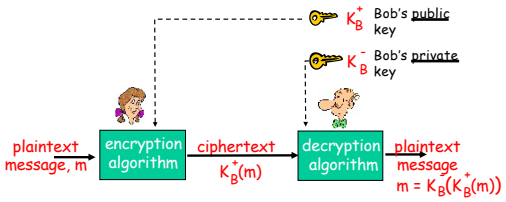
Public Key Cryptography

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 cryptography
- 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

13

Public key cryptography



14

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

15

RSA: Choosing keys

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 $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

16

RSA: Encryption, decryption

0. Given (*n,e*) and (*n,d*) as computed above
1. To encrypt bit pattern, *m*, compute
   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by *n*)
2. To decrypt received bit pattern, *c*, compute
   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by *n*)

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

17

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).

	letter	m	m^e	$c = m^e \bmod n$
encrypt:	L	12	1524832	17

	c	c^d	$m = c^d \bmod n$	letter
decrypt:	17	481968572106750915091411825223071697	12	L

18

RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

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

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

19

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

Result is the same!

20

Chapter 8 roadmap

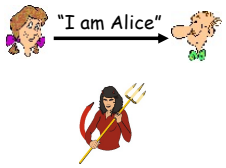
8.1 What is network security?
8.2 Principles of cryptography
8.3 Authentication
8.4 Integrity
8.5 Key Distribution and certification
8.6 Access control: firewalls
8.7 Attacks and counter measures
8.8 Security in many layers

21

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"



Failure scenario??

22

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

23

Authentication: another try

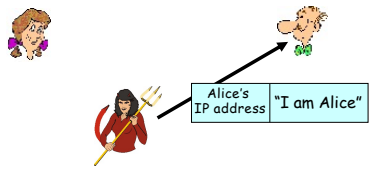
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Failure scenario??

24

Authentication: another try

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

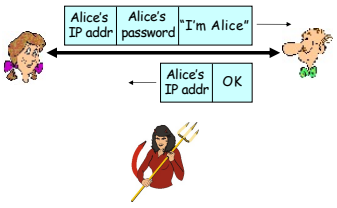


Trudy can create a packet "spoofing" Alice's address

25

Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

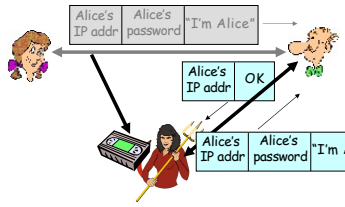


Failure scenario??

26

Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

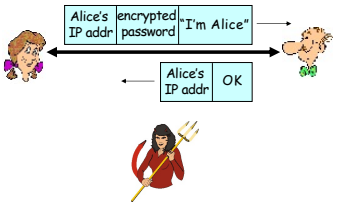


playback attack: Trudy records Alice's packet and later plays it back to Bob

27

Authentication: yet another try

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

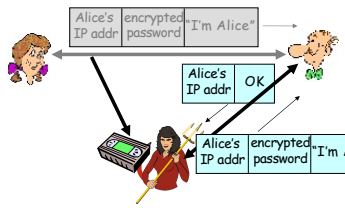


Failure scenario??

28

Authentication: another try

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



record and playback still works!

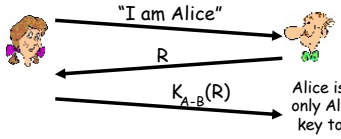
29

Authentication: yet another try

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



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

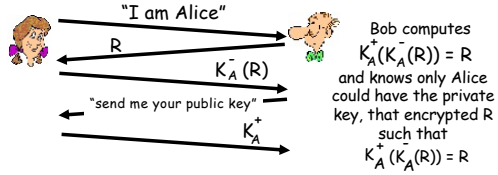
Failures, drawbacks?

30

Authentication: ap5.0

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

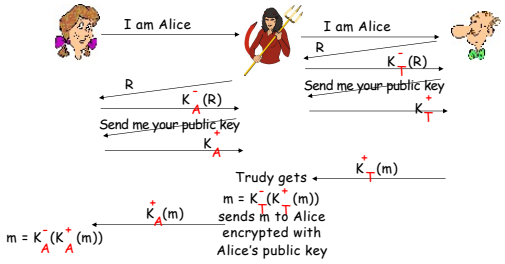
ap5.0: use nonce, public key cryptography



31

ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



32

ap5.0: security hole

Man (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!

33

Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Authentication
8.4 Message integrity
8.5 Key Distribution and certification

34

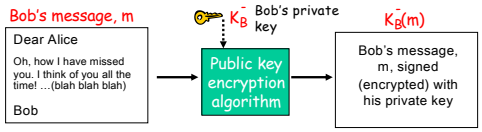
Digital Signatures

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

35

Digital Signatures

Simple digital signature for message m:
- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



36

Digital Signatures (more)

- Suppose Alice receives msg m, digital signature $K_B^-(m)$
- 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.

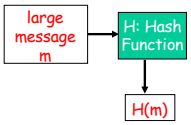
37

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)*.

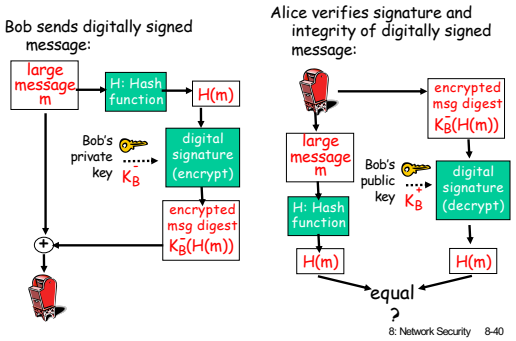


Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

38

Digital signature = signed message digest



40

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

41