ECE 356/COMPSI 356
Computer Network Architecture

Network Security

Monday December 2nd, 2019

Duke UNIVERSITY

1

Previous Lecture Recap

- Application layer protocols
  - Web and HTTP
  - Electronic mail
- Readings for this lecture: **PD Ch. 8**

2

Duke UNIVERSITY

2

Lecture Outline

- **What is network security?**
- Principles of cryptography
- Authentication
- Message integrity
- Securing e-mail
- Securing TCP connections: SSL
- Operational security: firewalls and IDS

3

The Internet is Insecure

- A network is a shared resource
- Attackers may eavesdrop, modify, or drop your packets!
- Can mount attacks on massive scales

4

Network Security Concepts (1/2)

- **Confidentiality**
  - Do you want to send your credit card #, login password over the Internet in plaintext?
  - *Traffic* confidentiality: knowing that communication has taken place might give away information as well
- **Integrity**
  - Data integrity: Imagine an Amazon transaction. Do you want your payment to be modified from $10.0 to $100?
  - Replay attack: You do not want the same transaction confirmation to be sent multiple times!
    - **Encryption does not prevent replay attacks**
  - Timeliness: delay a stock purchase

5

Duke UNIVERSITY

5

Network Security Concepts (2/2)

- **Authenticity**
  - Entity authentication: who are you talking to? Phishing attack
  - Message authentication: who sent this message?
- **Availability**
  - Denial of service attacks
- **Non-repudiation**
  - You've clicked the confirmation button!

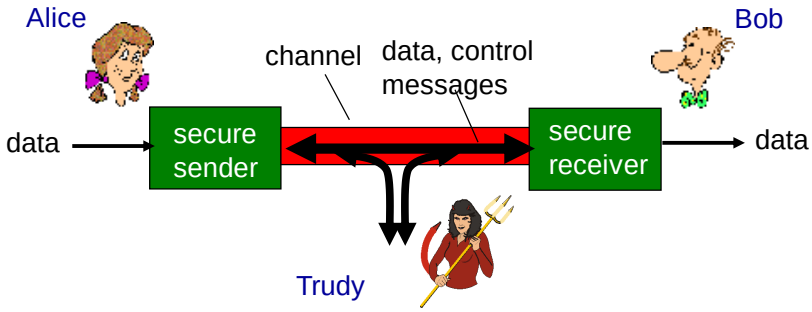
6

Duke UNIVERSITY

6

Friends and Enemies: Alice, Bob, Trudy, Eve

- Well-known in network security world
- Bob, Alice want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages
- Eve (eavesdropper) listens in on traffic



7

Duke UNIVERSITY

7

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
- …

8

Duke UNIVERSITY

8

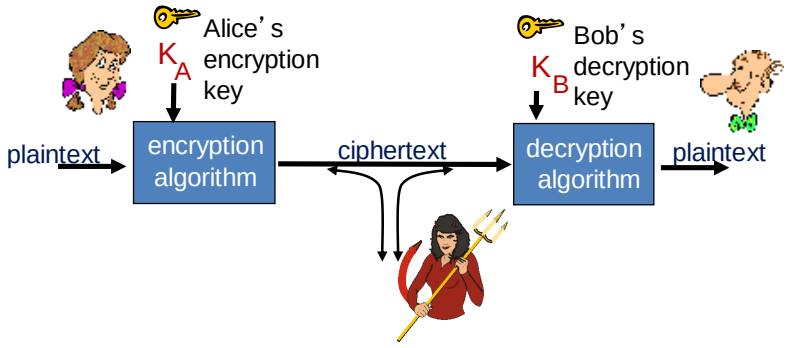
Who are Trudy and Eve?

- Who mounts network attacks and why?

Lecture Outline

- What is network security?
- **Principles of cryptography**
  - Symmetric keys
  - Public/private keys
- Authentication
- Message integrity
- Securing e-mail
- Securing TCP connections: SSL
- Operational security: firewalls and IDS

The Language of Cryptography



m plaintext message

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

$m = K_B(K_A(m))$

11

Duke UNIVERSITY

11

Breaking an Encryption Scheme

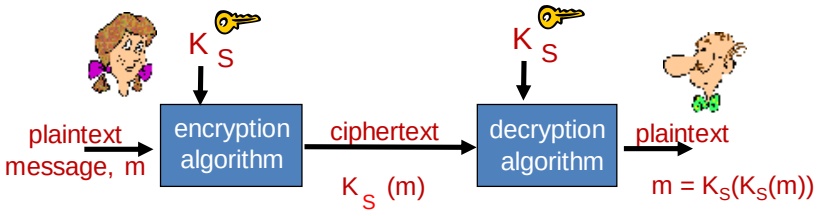
- Cipher-text only attack: Trudy has ciphertext she can analyze
- Two approaches:
  - Brute force: search through all keys
  - Statistical analysis
- Known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - E.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- Chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

12

Duke UNIVERSITY

12

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?

13

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 pay you. alice`
`ciphertext: nkn. s lmw wky. mgsbc`

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

14

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

15

Duke UNIVERSITY

Block Ciphers (1/2)

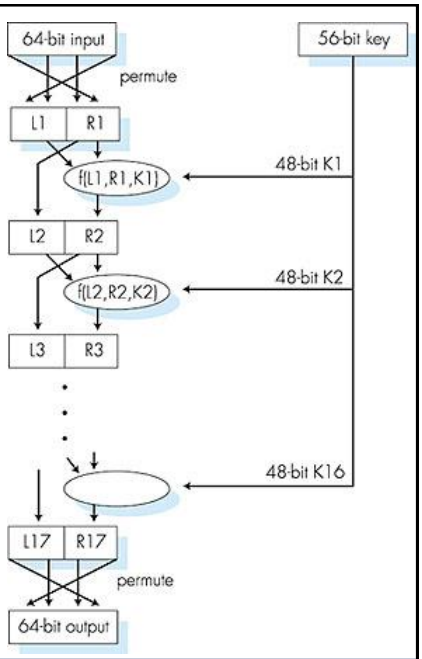
- **Block ciphers**: cleartext processed in blocks
- k-bit block of cleartext mapped directly to k-bit block of ciphertext
- E.g., for k = 3:
  - Input {0,0,0} mapped to output {1,1,0}
  - 2^3 = 8 possible inputs, permuted in 8! = 40,320 different ways
- Typically k is large, e.g.,64
  - 2^k! possible mappings
  - For k = 64, **18,446,744,073,709,551,616!** possible mappings
- In practice, cannot maintain and regenerate direct mappings of this scale

16

Duke UNIVERSITY

Block Ciphers (2/2)

- In practice:
  - Use functions that simulate randomly permuted tables
  - Use keys to customize the transformations
- E.g., DES:
  - Multiple rounds of expansion permutation (duplicating some of the bits), key mixing, substitution, permutation
- **Brute-force attacks**: cycle through all keys
  - Key of length n: 2^n possible keys



Symmetric Key Crypto: DES

DES: Data Encryption Standard

- US encryption standard, first published in 1977
- 56-bit symmetric key, 64-bit plaintext input
- Weakness: small key size
  - Concerns about the key size were raised early on
- Making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

AES: Advanced Encryption Standard

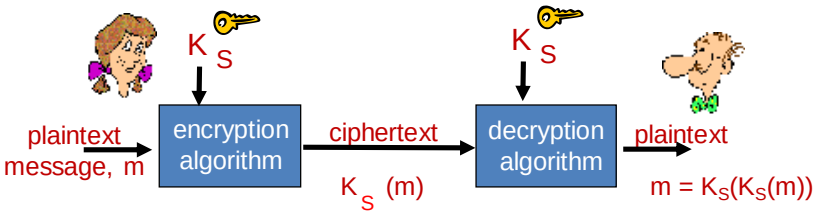
- Symmetric-key NIST standard, replaced DES (Nov 2001)
- Processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- A machine that could crack DES in 1 second would **take 149 trillion years for AES**

19

Duke UNIVERSITY

19

Symmetric Keys: Key Points to Remember



- Symmetric key cryptography: Bob and Alice share the same **secret key**
- Ciphertext usually generated from plaintext and key by using functions that simulate randomly permuted tables
  - Standards: DES, AES

20

Duke UNIVERSITY

20

Public Key Cryptography

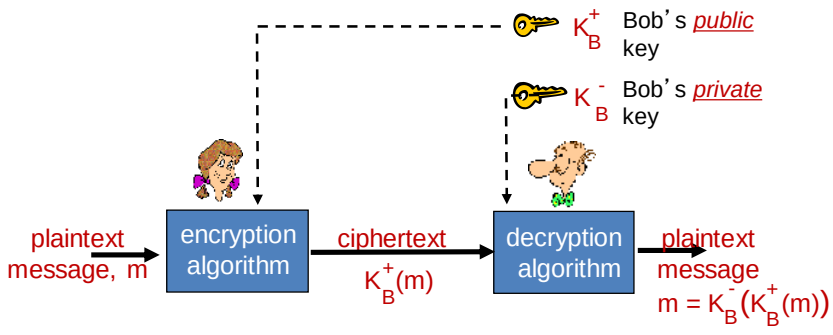
Symmetric key crypto

- 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



Public Key Encryption Algorithms

Requirements:

1. Need to have:

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

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

RSA: Rivest, Shamir, Adelson algorithm

23

Duke UNIVERSITY

Prerequisite: Modular Arithmetic

- x mod n = remainder of x when divided by n
  - E.g., 19 mod 5 = 4
- Facts:

(1) $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$

(2) $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$

(3) $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$

- Thus, from (3):

$$(a \bmod n)^d \bmod n = a^d \bmod n$$

- Example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

24

Duke UNIVERSITY

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)

25

RSA: Creating Public/Private Key Pair (1/2)

- Choose two large **prime numbers** *p, q*
  - These need to be kept secret
  - e.g., 1024 bits each
- From them, calculate **n**, **e** that will be public, and **d** that will be private
  - We share (n,e), calculated from p,q, but not p,q themselves

public key is (n,e). K_B^+ *private* key is (n,d). K_B^-

26

RSA: Creating Public/Private Key Pair

- Compute $n = pq$
  - Counting on the difficulty of factorization of the product of large prime numbers
- Compute $z = (p-1)(q-1)$
- Choose e (with $e<n$) that has no common factors with z
  - e, z are "relatively prime"
- Choose d such that $ed-1$ is exactly divisible by z
  - In other words: ed mod $z = 1$

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

27

Duke UNIVERSITY

RSA: Encryption, Decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m, compute
   $c = m^e \bmod n$

2. to decrypt received bit pattern c, compute
   $m = c^d \bmod n$

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

28

Duke UNIVERSITY

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

encrypt: m → m^e → $c = m^e \bmod n$

12 → 24832 → 17

decrypt: c → c^d → $m = c^d \bmod n$

17 → 48196857210675091509141182522307169 7 → 12

29

Duke UNIVERSITY

Why Does RSA Work?

- Must show that $c^d \bmod n = m$, where $c = m^e \bmod n$
  - From $(a \bmod n)^d \bmod n = a^d \bmod n$, $(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$
- Fact: if p,q are prime, for n= pq and z = (p-1)(q-1), for any x and y:
  - $x^y \bmod n = x^{(y \bmod z)} \bmod n$
- Thus,

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

Selected *ed* mod *z* = *1*

30

Duke UNIVERSITY

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!

31

Duke UNIVERSITY

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

follows directly from modular arithmetic:

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

32

Duke UNIVERSITY

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

33

Duke UNIVERSITY

33

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

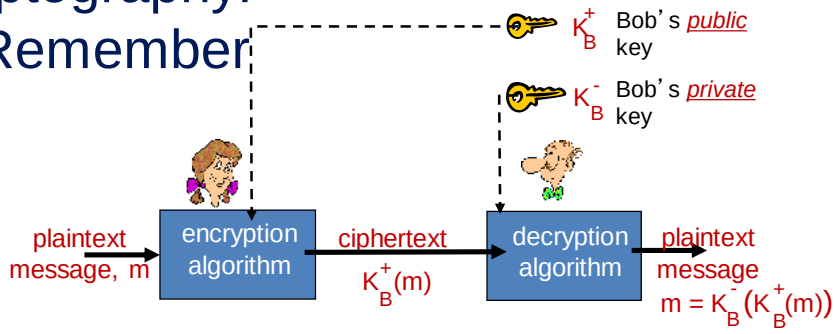
- Bob and Alice use RSA to exchange a symmetric key K_S
- Once both have K_S, they use symmetric key cryptography

34

Duke UNIVERSITY

34

Public Key Cryptography: Key Points to Remember



- Public key known to **everyone**
- Private key is known only to Bob
- Cryptographic algorithms ensure that private key cannot be calculated from the public one
- RSA is a widely used for secure data transmission

35

Duke UNIVERSITY

35

Lecture Outline

- What is network security?
- Principles of cryptography
- **Authentication**
- Message integrity
- Securing e-mail
- Securing TCP connections: SSL
- Operational security: firewalls and IDS

36

Duke UNIVERSITY

36

Authentication Protocols

- Authentication protocols: identify communicating parties
  - In real life, identify each other by face, voice, posture, …
  - Customs official authenticates us based on a picture in our passport
- Typically run before communication protocols

Authentication

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

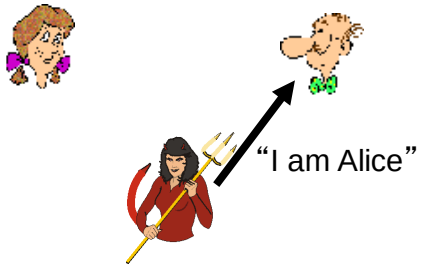
Protocol ap1.0: Alice says "I am Alice"

Failure scenario??

Authentication

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

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



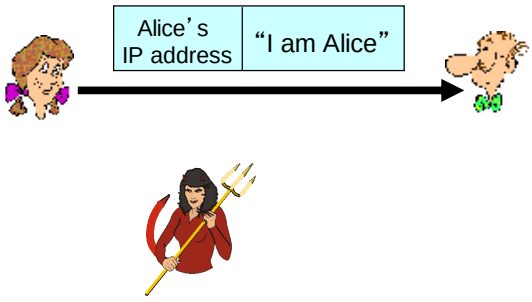
In a network,
Bob cannot "see" Alice,
so Trudy simply
declares
herself to be Alice

39

Duke UNIVERSITY

Authentication: Another Try

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



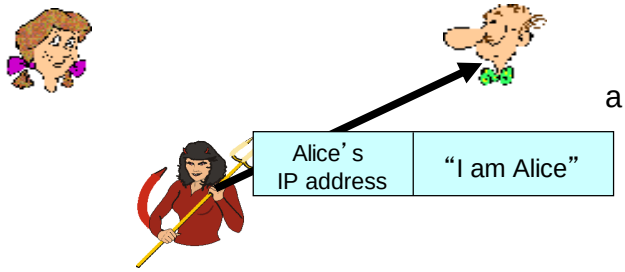
Failure scenario?

40

Duke UNIVERSITY

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

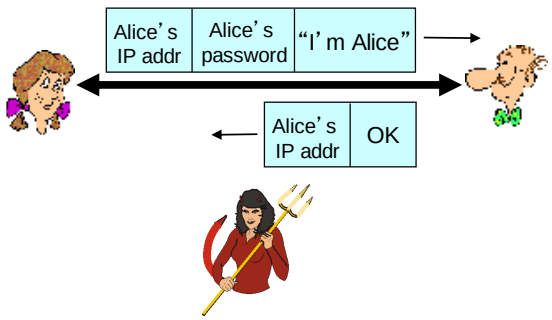
41

Duke UNIVERSITY

41

Authentication: Another Try

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



Failure scenario?

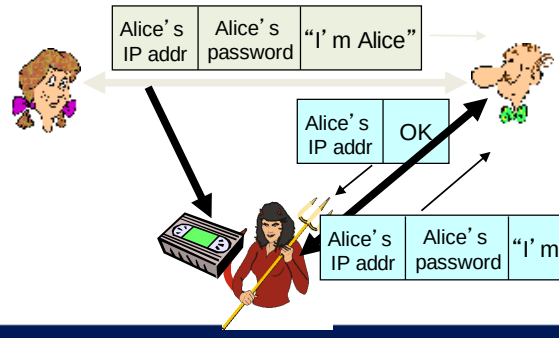
42

Duke UNIVERSITY

42

Authentication: Another Try

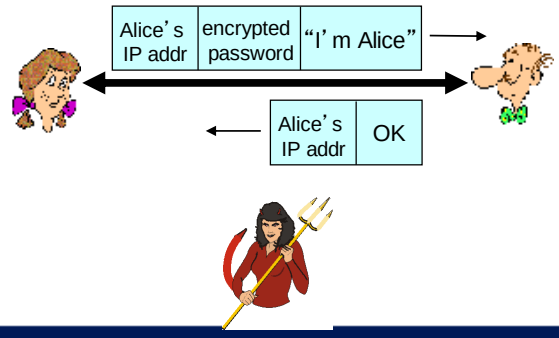
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to prove it



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

Authentication: Yet Another Try

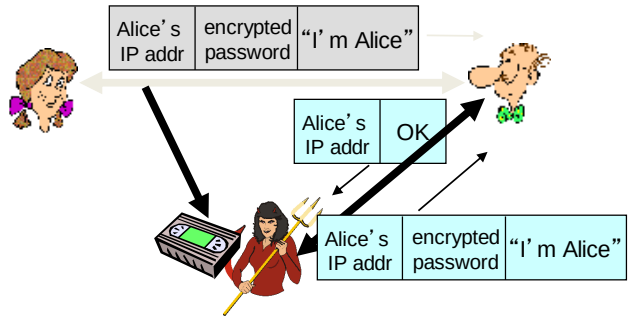
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario?

Authentication: Yet Another Try

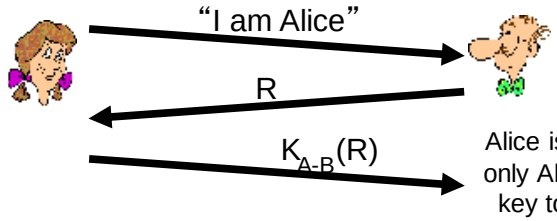
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



record and playback *still* works!

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!

Authentication Protocols: Key Points to Remember

- Verify that you are talking to the person you believe
- Typically run before communication protocols
- Approach: shared secret key + a nonce
  - Use of a nonce prevents a replay attack

Final Policy

- Date, location: on Duke Hub
  - December 15th, 7 – 10 PM, HH 125
- Covers all lectures, with an emphasis on post-midterm material
  - Includes the invited lecture
- Closed book/notes
- No Internet
- Allowed:
  - One two-sided hand-written page, written by you (letter-size)
  - A calculator

You Need to Know The Material Covered in Class

- Will not ask questions on material covered in the book but not in class
- Will ask questions on lecture materials not covered in the book

49

Duke UNIVERSITY

49

Additional Study Materials

- HW1, HW2, HW3 solutions
- Lab2, Lab3 concepts and materials
  - If you are working in a group, please make sure to understand the parts done by your lab partner
- In-class quiz answers
  - Make sure you understand everything you did not get right
- Midterm and makeup midterm solutions
  - Make sure you understand everything you did not get right
  - Note that final emphasizes post-midterm materials

50

Duke UNIVERSITY

50

Lecture Outline

- What is network security?
- Principles of cryptography
- Authentication
- **Message integrity**
- Securing e-mail
- Securing TCP connections: SSL
- Operational security: firewalls and IDS

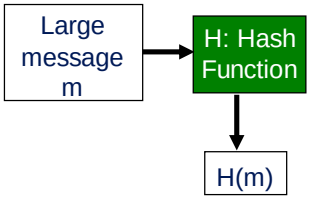
51

Message Integrity

- Need to know that the message was not modified in transit
- Possible approach:
  - Sender calculates a checksum and appends it to the message
  - Receiver verifies that the received checksum is correct
- Seen a similar approach for error checking before
- Need to keep *malicious modifications* in mind in integrity verification

52

Cryptographic Hash Functions



Hash function properties:

- Many-to-1
- Produces fixed-size message "fingerprint" that can be used to verify that the message has not been tampered with

- **Cryptographic hash function** has to have an additional property
  - ➢ It should be computationally infeasible to find any two different messages x and y such that H(x) = H(y)

53

Duke UNIVERSITY

Widely Used Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - ➢ Computes 128-bit message digest in 4-step process
- SHA-1 is also used
  - ➢ US standard [NIST, FIPS PUB 180-1]
  - ➢ 160-bit message digest

54

Duke UNIVERSITY

Message Authentication Codes

- Rely on a shared secret s + hash calculation
  - Alice creates message m, concatenates s to create m+s, and calculates **message authentication code** H(m+s)
  - Alice appends the MAC to the message
    - Sends (m, H(m+s))
  - Bob, knowing s, calculates H(m+s), verifies the correctness of the message
- Note that MAC calculations do not require encryption
  - E.g., router communications need to be authenticated but may not need to be encrypted

55

Digital Signatures

Cryptographic technique analogous to hand-written signatures:

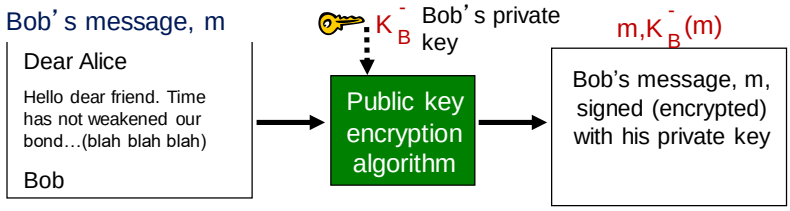
- Sender (Bob) digitally signs document, establishing he is document owner/creator
- *Verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
  - Cannot use message authentication codes since they rely on a shared secret

56

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating a "signed" message, $K_B^-(m)$



57

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

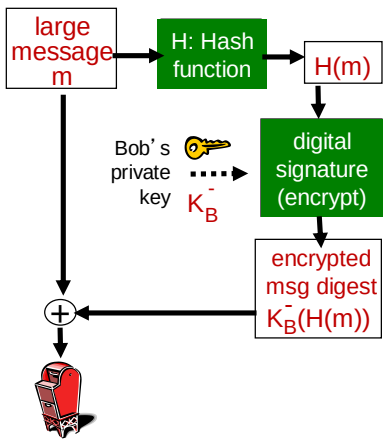
58

Message Digests

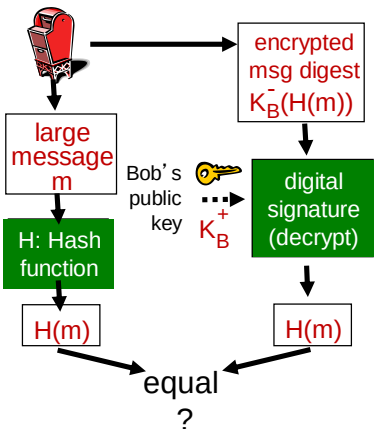
- Computationally expensive to public-key-encrypt long messages
- *Goal:* fixed-length, easy- to-compute digital "fingerprint"
  - Apply hash function H to *m*, get fixed size **message digest**, *H(m)*

Digital Signature = Signed Message Digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Message Authentication Codes and Digital Signatures: Key Points to Remember

- Create "checksums" of documents using **cryptographic hash functions**
  - Internet checksum and the CRC are not suitable for these calculations
  - It needs to be difficult to find any two different messages x,y such that H(x) = H(y)
- MAC calculations rely on a shared secret
- Digital signatures use public-private keys

61

Duke UNIVERSITY

61

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

62

Duke UNIVERSITY

62

Certification Authorities (1/2)

- *Certification authority (CA):* binds public key to particular entity, E
- So-called trusted 3rd party
- Commercial entities
  - Symantec, Comodo, GoDaddy, GlobalSign, DigitCert, …

63

Duke UNIVERSITY

63

Certification Authorities

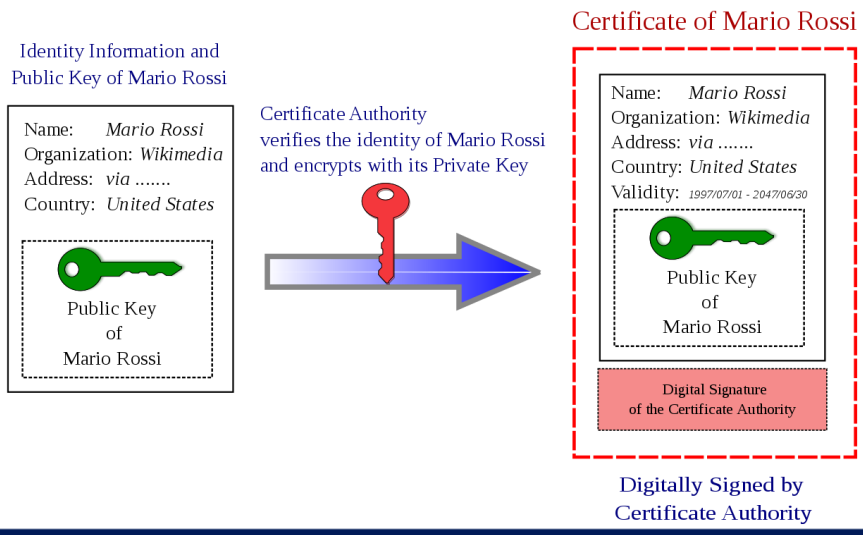
- *Certification authority (CA):* binds public key to particular entity, E
- E (person, router) registers its public key with CA
  - E provides "proof of identity" to CA
  - CA creates a **certificate** binding E to its public key
  - Certificate containing E's public key digitally signed by CA: CA says "*this is E's public key*"

64

Duke UNIVERSITY

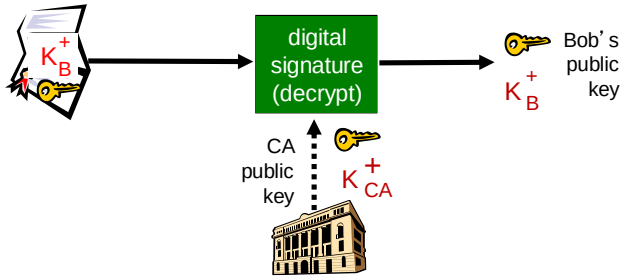
64

Obtaining a Public Key Certificate



Verifying Identity via Certificates

- When Alice wants Bob's public key:
  - Gets Bob's certificate
  - Apply CA's public key to Bob's certificate, get Bob's public key



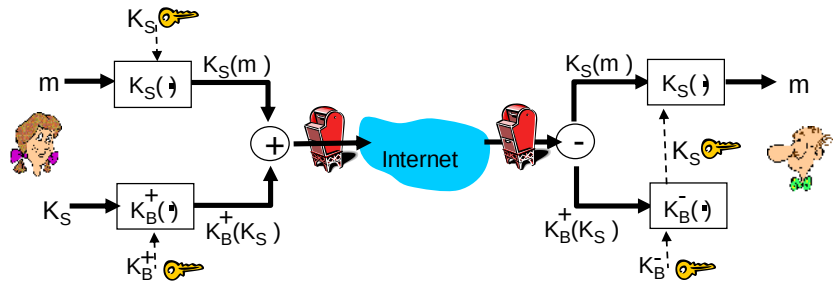
Lecture Outline

- What is network security?
- Principles of cryptography
- Authentication
- Message integrity
- **Securing e-mail**
- Securing TCP connections: SSL
- Operational security: firewalls and IDS

67

Secure E-mail: Confidentiality (1/2)

Alice wants to send **confidential e-mail**, m, to Bob.



Session key approach

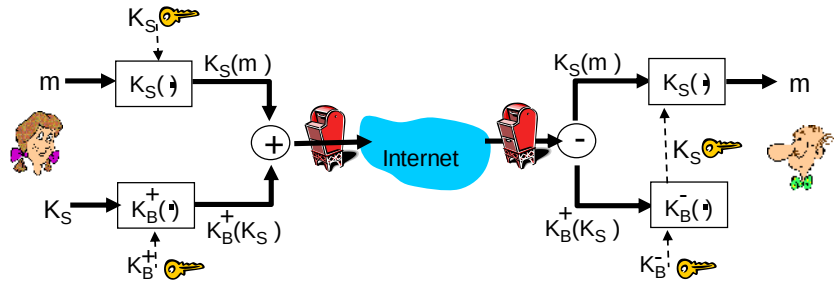
Alice:

1. Generates random symmetric session key, K_S
2. Encrypts message with K_S
3. Encrypts K_S with Bob's public key
4. Sends both $K_S(m)$ and $K_B(K_S)$ to Bob

68

Secure E-mail: Confidentiality (2/2)

Alice wants to send confidential e-mail, m, to Bob.



Session key approach

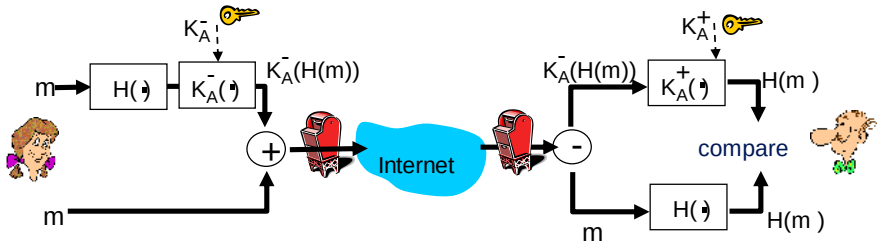
Bob:

- Uses his private key to decrypt and recover K_S
- Uses K_S to decrypt $K_S(m)$ to recover m

69

Secure E-mail: Sender Authentication And Message Integrity

Alice wants to provide sender authentication and message integrity

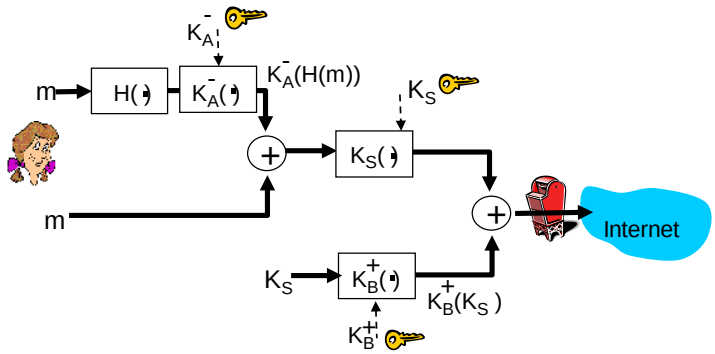


- Alice digitally signs message
- Sends both message (in the clear) and digital signature

70

Secure E-mail: Secrecy and Integrity

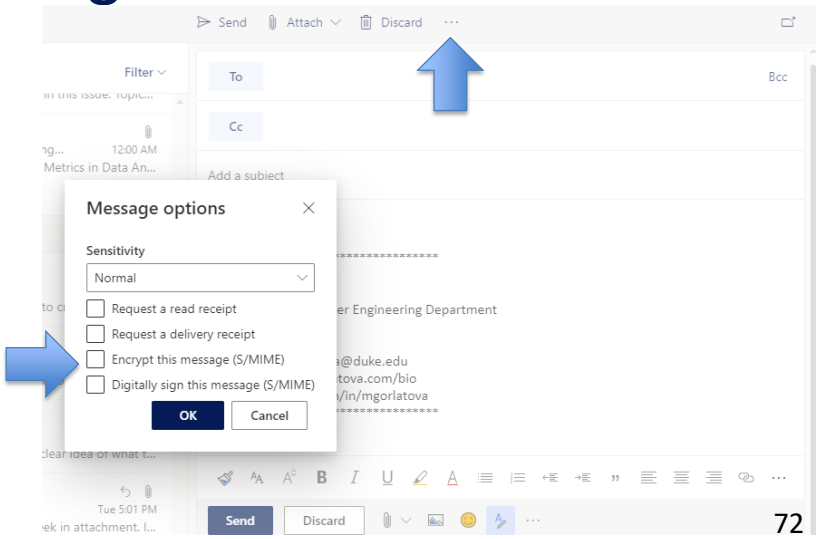
Alice wants to provide secrecy, sender authentication, and message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Securing Duke E-mails

- May require installing an extension



Lecture Outline

- What is network security?
- Principles of cryptography
- Authentication
- Message integrity
- Securing e-mail
- **Securing TCP connections: SSL**
- Operational security: firewalls and IDS

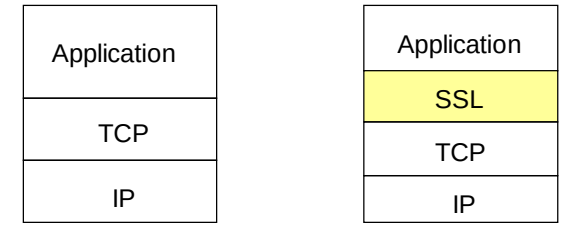
73

Secure Sockets Layer (SSL)

- Originally designed by Netscape
- Widely deployed security protocol
  - Supported by almost all browsers, web servers
  - **Https**
  - Billions $/year over SSL
- 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

74

SSL and TCP/IP



Normal application *Application with SSL*

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

75

Duke UNIVERSITY

75

Toy SSL: A Simple Secure Channel

- *Handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *Key derivation:* Alice and Bob use shared secret to derive set of keys
- *Data transfer:* data to be transferred is broken up into series of records
- *Connection closure:* special messages to securely close connection

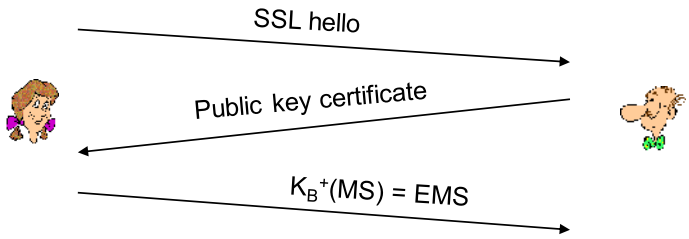
76

Duke UNIVERSITY

76

Toy: A Simple Handshake

- First, a TCP handshake. Then:



MS: master secret generated by Alice

EMS: encrypted master secret

- Bob decrypts it with his private key to get MS

77

Duke UNIVERSITY

77

Toy: Key Derivation

- Considered bad to use same key for more than one cryptographic operation
  - Use different keys for message authentication code (MAC) and encryption
- Alice and Bob use the MS to generate **four keys**:
  - K_c = encryption key for data sent from client to server
  - M_c = *integrity verification* MAC key for data sent from client to server
  - K_s = encryption key for data sent from server to client
  - M_s = *integrity verification* MAC key for data sent from server to client
- Keys derived from *key derivation function* (KDF)
  - Takes master secret and (possibly) some additional random data and creates the keys

78

Duke UNIVERSITY

78

Toy: Data Records (1/2)

- Why not encrypt data in constant stream as we write it to TCP?
  - Where would we put the MAC? If at end, no message integrity until all data processed
  - E.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?

79

Toy: Data Records (2/2)

- Instead, break stream in series of records
  - Each record carries a MAC
  - Receiver can act on each record as it arrives
- Issue: in record, receiver needs to distinguish MAC from data
  - Want to use variable-length records

80

Toy: Sequence Numbers

- *Problem:* attacker can capture and replay record or re-order records
  - TCP sequence numbers are not encrypted
- *Solution:* put sequence number in the MAC calculations
  - MAC = MAC(M_x, sequence number)
  - Note: no sequence number field

SSL: Cipher Suites

- Cipher suite
  - Public key algorithm
  - Symmetric encryption algorithm
  - MAC algorithm
- SSL supports several cipher suites
- Negotiation: client, server agree on cipher suite
  - Client offers choice
  - Server picks one

Real SSL: Handshake (1/4)

Purpose

1. Server authentication
2. Negotiation: agree on crypto algorithms
3. Establish keys
4. Client authentication (optional)

Real SSL: Handshake (2/4)

1. Client sends list of algorithms it supports, along with client nonce
2. Server chooses algorithms from list; sends back: choice + certificate + server nonce
3. Client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. Client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. Client sends a MAC of all the handshake messages
6. Server sends a MAC of all the handshake messages

Real SSL: Handshaking (3/4)

Last 2 steps protect handshake from tampering

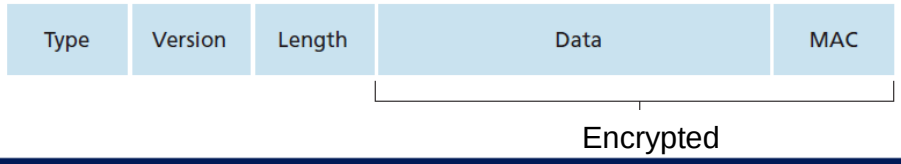
- Client typically offers range of algorithms, some strong, some weak
- Man-in-the middle could delete stronger algorithms from list
- Last 2 steps prevent this
  - Last two messages are encrypted

Real SSL: Handshaking (4/4)

- Why two random nonces?
- Suppose Trudy sniffs all messages between Alice & Bob
- Next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - Solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

Closing an SSL Connection

- Problem: **truncation attack**
  - Attacker forges TCP connection close segment
  - One or both sides thinks there is less data than there actually is
- Solution:
  - Indicate closure in an SSL fragment
  - Use type field: Type 0 for data; type 1 for closure



Secure Sockets Layer: Key Points to Remember

- Widely deployed protocol
  - Anything done over HTTPS is done over SSL
- Encrypted data transmitted in segments, which include a MAC

- Uses handshaking to agree on a cipher suite and to establish encryption and MAC keys

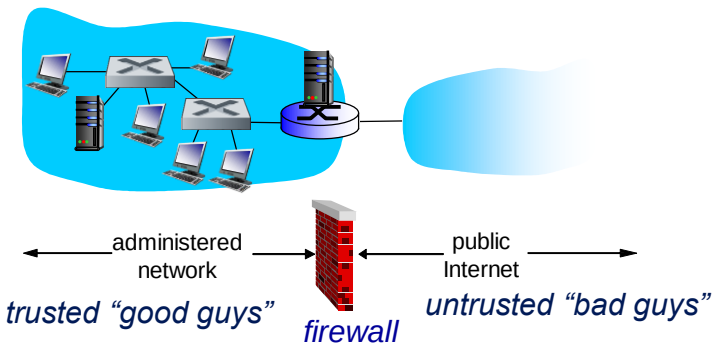
Lecture Outline

- What is network security?
- Principles of cryptography
- Authentication
- Message integrity
- Securing e-mail
- Securing TCP connections: SSL
- **Operational security: firewalls and IDS**

Firewalls

Firewall

Isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others



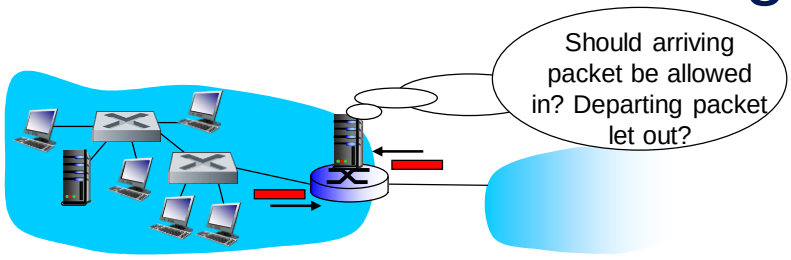
Firewall Goals

- All traffic from outside to inside, and vise versa, passes through the firewall
- Only authorized traffic, as defined by the local access policy, will be allowed to pass
- The firewall itself is immune from penetration

Three types of firewalls:

- Stateless packet filters
- Stateful packet filters
- Application gateways

Stateless Packet Filtering



- Internal network connected to Internet via *router firewall*
- Router *filters packet-by-packet,* decision to forward/drop packet based on:
  - Source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

Stateless Packet Filtering: Example

- *Example 1:* block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23
  - *Result:* all incoming, outgoing UDP flows and telnet connections are blocked
- *Example 2:* block inbound TCP segments with ACK=0.
  - *Result:* prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside

Stateless Packet Filtering: More Examples

Policy	*Firewall Setting*
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Stateful Packet Filtering (1/2)

- *Stateless packet filter:* heavy handed tool
  - Admits packets that "make no sense," e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- *Stateful packet filter:* track status of every TCP connection
  - Track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets "makes sense"
  - Timeout inactive connections at firewall: no longer admit packets

Stateful Packet Filtering (2/2)

ACL augmented to indicate need to check connection state table before admitting packet

action	source address	dest address	proto	source port	dest port	flag bit	check conxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	X
deny	all	all	all	all	all	all	

Limitations of Firewalls

- *Tradeoff:* degree of communication with outside world, level of security
- Offer only perimeter defense
- *IP spoofing:* router cannot know if data "really" comes from claimed source
- Many highly protected sites still suffer from attacks

Intrusion Detection Systems (1/3)

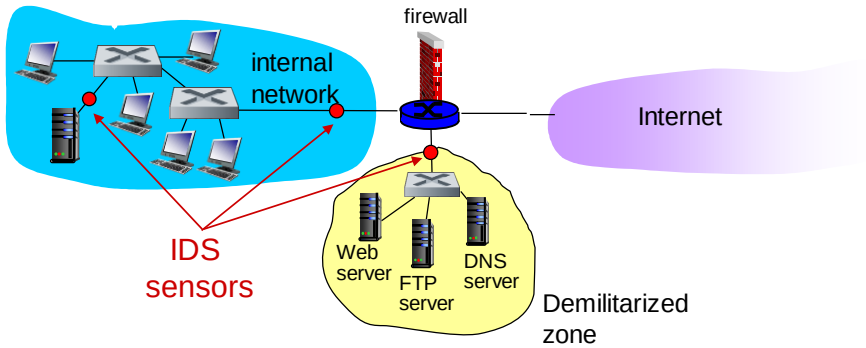
- Packet filtering:
  - Operates on TCP/IP headers only
  - No correlation check among sessions
- *IDS: intrusion detection system*
  - *Deep packet inspection:* look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
  - Examine correlation among multiple packets
    - Port scanning
    - Network mapping
    - DoS attack

Intrusion Detection Systems (2/3)

- Signature-based systems
  - Compare traffic against a set of attack signatures
  - Con: cannot detect attacks that have not been seen before
- Anomaly-based systems
  - Compare traffic to traffic under “normal operation”
  - Subject of much machine learning research

Intrusion Detection Systems (3/3)

Multiple IDSs: different types of checking at different locations



Lecture Summary

- What is network security?
- Principles of cryptography
- Authentication
- Message integrity
- Securing e-mail
- Securing TCP connections: SSL
- Operational security: firewalls and IDS

Next Lecture

- Final review

Duke UNIVERSITY

103