

## 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
- 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

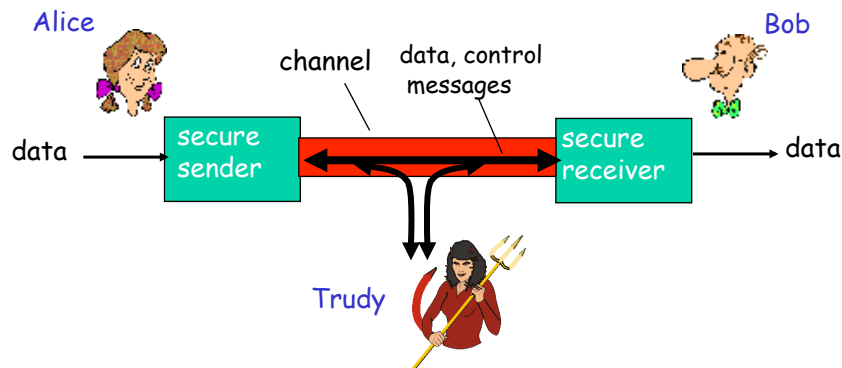
**Authentication:** sender, receiver want to confirm identity of each other

**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!

- *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

8.4 Securing e-mail

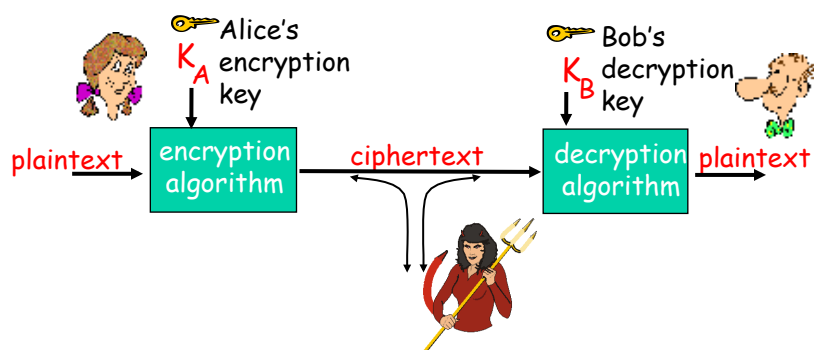
8.5 Securing TCP connections: SSL

8.6 Network layer security: IPsec

8.7 Securing wireless LANs

8.8 Operational security: firewalls and IDS

## The language of cryptography



$m$  plaintext message

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

$m = K_B(K_A(m))$

## 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

Key: the mapping from the set of 26 letters to the set of 26 letters

9

## Polyalphabetic encryption

- n monoalphabetic ciphers,  $M_1, M_2, \dots, M_n$
- 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 monoalphabetic pattern in cyclic pattern
  - dog: d from  $M_1$ , o from  $M_3$ , g from  $M_4$
- Key: the n ciphers and the cyclic pattern

10

## Breaking an encryption scheme

- **Cipher-text only attack:** Trudy has ciphertext that she can analyze
- **Two approaches:**
  - Search through all keys: must be able to differentiate resulting plaintext from gibberish
  - Statistical analysis
- **Known-plaintext attack:** trudy has some plaintext corresponding to some ciphertext
  - eg, in monoalphabetic cipher, trudy determines pairings for a,l,i,c,e,b,o,
- **Chosen-plaintext attack:** trudy can get the cyphertext for some chosen plaintext

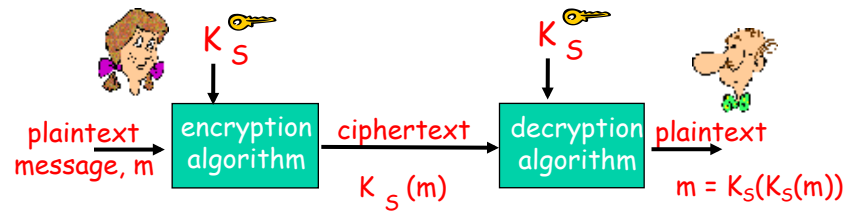
11

## Types of Cryptography

- Crypto often uses keys:
  - Algorithm is known to everyone
  - Only "keys" are secret
- Public key cryptography
  - Involves the use of two keys
- Symmetric key cryptography
  - Involves the use one key
- Hash functions
  - Involves the use of no keys
  - Nothing secret: How can this be useful?

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?

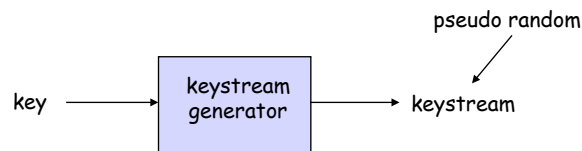
13

## Two types of symmetric ciphers

- Stream ciphers
  - encrypt one byte at time
- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

14

## Stream Ciphers



- ❑ Combine each bit of keystream with bit of plaintext to get bit of ciphertext
- ❑  $m(i)$  =  $i$ th bit of message
- ❑  $ks(i)$  =  $i$ th bit of keystream
- ❑  $c(i)$  =  $i$ th bit of ciphertext
- ❑  $c(i) = ks(i) \oplus m(i)$  ( $\oplus$  = exclusive or)
- ❑  $m(i) = ks(i) \oplus c(i)$

15

## RC4 Stream Cipher

- ❑ RC4 is a popular stream cipher
  - Extensively analyzed and considered good
  - Key can be from 1 to 256 bytes
  - Used in WEP for 802.11
  - Can be used in SSL

16



## Block ciphers

- ❑ Message to be encrypted is processed in blocks of  $k$  bits (e.g., 64-bit blocks).
- ❑ 1-to-1 mapping is used to map  $k$ -bit block of plaintext to  $k$ -bit block of ciphertext

### Example with $k=3$ :

<u>input</u>	<u>output</u>	<u>input</u>	<u>output</u>
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

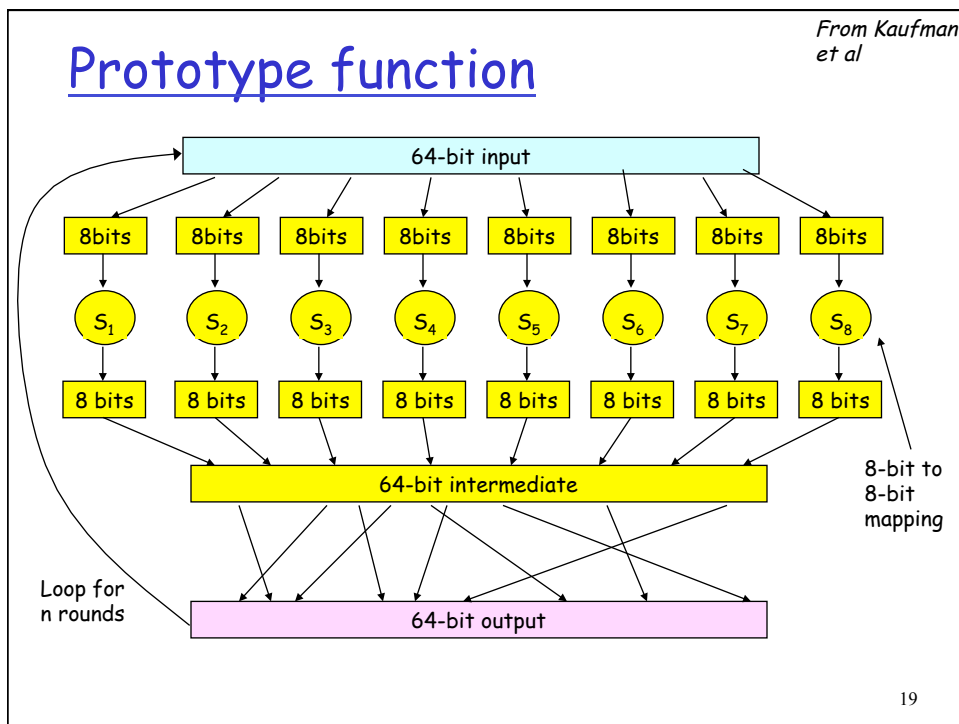
What is the ciphertext for 010110001111 ?

17

## Block ciphers

- ❑ What should be the block size?
  - small blocks vulnerable to analysis attacks
  - large blocks - say 64 bits in length
    - random substitution requires  $2^{64}$  table entries
    - random permutation requires  $64 \times 8$  entries
- ❑ substitution very secure
- ❑ permutations - hardly any security
- ❑ goal: input/output should be uncorrelated, any change in input must not appear at specific locations in the output

18



## Why rounds?

- ❑ If only a single round, then one bit of input affects at most 8 bits of output.
- ❑ In 2<sup>nd</sup> round, the 8 affected bits get scattered and inputted into multiple substitution boxes.
- ❑ How many rounds?
  - How many times do you need to shuffle cards
  - Becomes less efficient as n increases

## Symmetric key crypto: DES

### DES: Data Encryption Standard

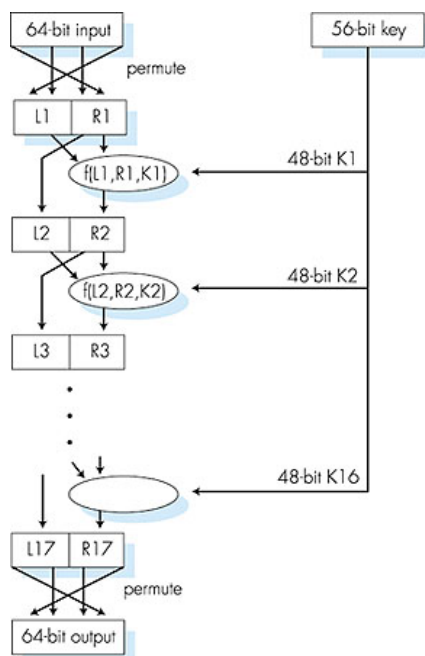
- ❑ US encryption standard [NIST 1993]
- ❑ 56-bit symmetric key, 64-bit plaintext input
- ❑ Block cipher with cipher block chaining
- ❑ How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - No known good analytic attack
- ❑ making DES more secure:
  - 3DES: encrypt 3 times with 2 different keys (actually encrypt, decrypt, encrypt)

21

## Symmetric key crypto: DES

### DES operation

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



22

## 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

23

## Encrypting a large message

- ❑ Why not just break message in 64-bit blocks, encrypt each block separately?
  - If same block of plaintext appears twice, will give same cyphertext.
- ❑ How about:
  - Generate random 64-bit number  $r(i)$  for each plaintext block  $m(i)$
  - Calculate  $c(i) = K_S(m(i) \oplus r(i))$
  - Transmit  $c(i), r(i), i=1,2,\dots$
  - At receiver:  $m(i) = K_S(c(i)) \oplus r(i)$
  - Problem: inefficient, need to send  $c(i)$  and  $r(i)$

24

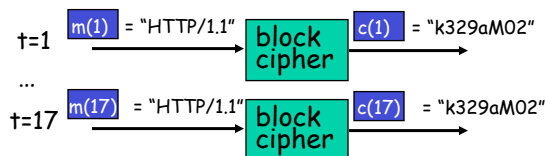
## Cipher Block Chaining (CBC)

- CBC generates its own random numbers
  - Have encryption of current block depend on result of previous block
  - $c(i) = K_S(m(i) \oplus c(i-1))$
  - $m(i) = K_D(c(i)) \oplus c(i-1)$
- How do we encrypt first block?
  - Initialization vector (IV): random block =  $c(0)$
  - IV does not have to be secret
- Change IV for each message (or session)
  - Guarantees that even if the same message is sent repeatedly, the ciphertext will be completely different each time

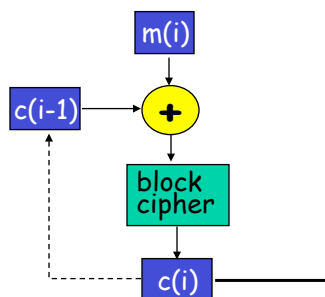
25

## Cipher Block Chaining

- cipher block: if input block repeated, will produce same cipher text:



- *cipher block chaining:*  
XOR ith input block,  $m(i)$ , with previous block of cipher text,  $c(i-1)$ 
  - $c(0)$  transmitted to receiver in clear
  - what happens in "HTTP/1.1" scenario from above?



## 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")?

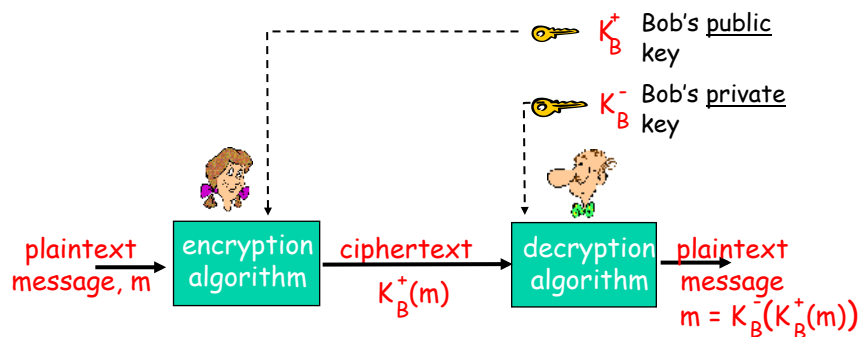
### public key cryptography

- 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



27

## Public key cryptography



28

## 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, Adelman algorithm

29

## Prerequisite: modular arithmetic

- $x \bmod n$  = remainder of  $x$  when divide by  $n$

- Facts:

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

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

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

- Thus

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

- Example:  $x=14$ ,  $n=10$ ,  $d=2$ :

$$(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$$

$$x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$$

30

## RSA: getting ready

- A message is a bit pattern.
- A 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 cyphertext).

31

## 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:  $ed \bmod z = 1$ ).
5. Public key is  $(n, e)$ . Private key is  $(n, d)$ .  

$\underbrace{\hspace{1.5em}}_{K_B^+}$

$\underbrace{\hspace{1.5em}}_{K_B^-}$

32



## RSA: Encryption, decryption

0. Given  $(n,e)$  and  $(n,d)$  as computed above
1. To encrypt message  $m (<n)$ , compute  

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

$$m = c^d \bmod n$$

Magic happens!

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

33

## 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:	<u>bit pattern</u>	<u>m</u>	<u>m<sup>e</sup></u>	<u>c = m<sup>e</sup> mod n</u>
	00001000	12	24832	17
decrypt:	<u>c</u>	<u>c<sup>d</sup></u>	<u>m = c<sup>d</sup> mod n</u>	
	17	481968572106750915091411825223071697	12	

34

## Why does RSA work?

- Must show that  $c^d \bmod n = m$   
where  $c = m^e \bmod n$
- Fact: for any  $x$  and  $y$ :  $x^y \bmod n = x^{(y \bmod z)} \bmod n$ 
  - where  $n = pq$  and  $z = (p-1)(q-1)$
- Thus,
 
$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(ed \bmod z)} \bmod n \\ &= m^1 \bmod n \\ &= m \text{ (assuming } m < n) \end{aligned}$$

35

## RSA: another important property

The following property will be *very* useful later:

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

use public key  
first, followed  
by private key

use private key  
first, followed  
by public key

*Result is the same!*

36

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

Follows directly from modular arithmetic:

$$\begin{aligned} (m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n \end{aligned}$$

37

## 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.

## Generating RSA keys

- Have to find big primes  $p$  and  $q$
- Approach: make good guess then apply  
testing rules (see Kaufman)

38

## Session keys

- ❑ Exponentiation is computationally intensive
- ❑ DES is at least 100 times faster than RSA

### 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

39

## Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 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

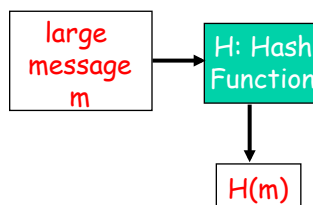
## Message Integrity

- Allows communicating parties to verify that received messages are authentic.
  - Content of message has not been altered
  - Source of message is who/what you think it is
  - Message has not been replayed
  - Sequence of messages is maintained
- Let's first talk about message digests

41

## Message Digests

- Function  $H()$  that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- Note that  $H()$  is a many-to-1 function
- $H()$  is often called a "hash function"



- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine  $m$  from  $H(m)$
  - Collision resistance: Computationally difficult to produce  $m$  and  $m'$  such that  $H(m) = H(m')$
  - Seemingly random output

42

## Internet checksum: poor message digest

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of input
- is many-to-one
- But given message with given hash value, it is easy to find another message with same hash value.
- Example: Simplified checksum: add 4-byte chunks at a time:

<u>message</u>	<u>ASCII format</u>		<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31		I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39		0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42		9 B O B	39 42 D2 42
	<u>B2 C1 D2 AC</u>	← different messages		<u>B2 C1 D2 AC</u>
		but identical checksums!		

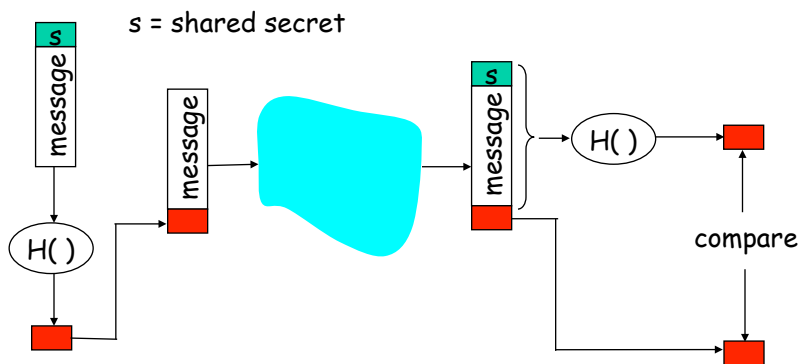
43

## Hash Function Algorithms

- MD5 hash function was widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
- now mostly SHA-\* is used
  - US standard [NIST, FIPS PUB 180-1]
  - SHA-1: 160-bit message digest

44

## Message Authentication Code (MAC)



- ❑ **Authenticates sender**
- ❑ **Verifies message integrity**
- ❑ No encryption !
- ❑ Also called "keyed hash"
- ❑ Notation:  $MD_m = H(s||m)$  ; send  $m||MD_m$

45

## HMAC

- ❑ Popular MAC standard
  - ❑ Addresses some subtle security flaws
1. Concatenates secret to front of message.
  2. Hashes concatenated message
  3. Concatenates the secret to front of digest
  4. Hashes the combination again.

46

## Example: OSPF

- Recall that OSPF is an intra-AS routing protocol
  - Each router creates map of entire AS (or area) and runs shortest path algorithm over map.
  - Router receives link-state advertisements (LSAs) from all other routers in AS.
- Attacks:**
- Message insertion
  - Message deletion
  - Message modification
  - How do we know if an OSPF message is authentic?

47

## OSPF Authentication

- Within an Autonomous System, routers send OSPF messages to each other.
- OSPF provides authentication choices
  - No authentication
  - Shared password: inserted in clear in 64-bit authentication field in OSPF packet
  - Cryptographic hash
- Cryptographic hash with MD5
  - 64-bit authentication field includes 32-bit sequence number
  - MD5 is run over a concatenation of the OSPF packet and shared secret key
  - MD5 hash then appended to OSPF packet; encapsulated in IP datagram

48

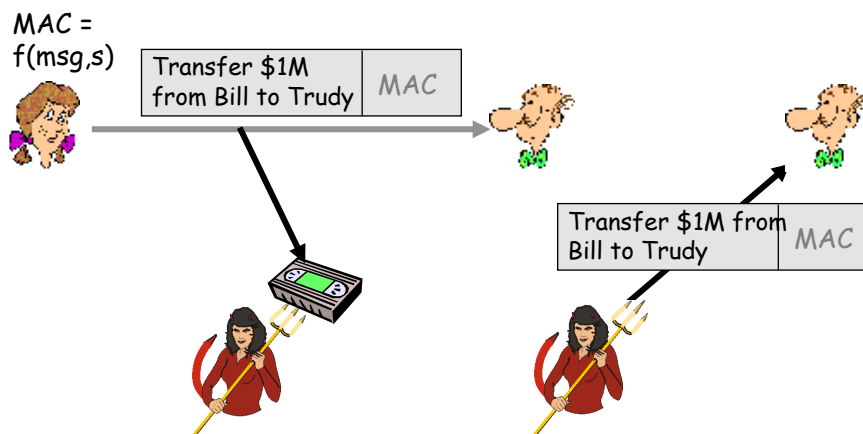


## End-point authentication

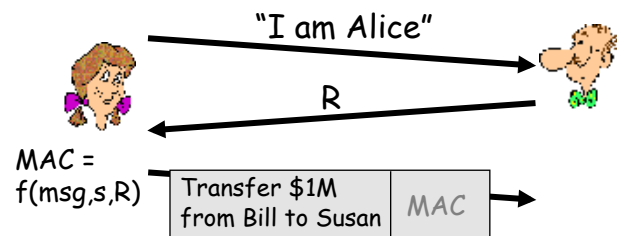
- Want to be sure of the originator of the message - *end-point authentication*.
- Assuming Alice and Bob have a shared secret, use a MAC for authentication
  - We do know that Alice created the message.
  - But did she send it?

49

## Playback attack



## Defending against playback attack: nonce



## Digital Signatures

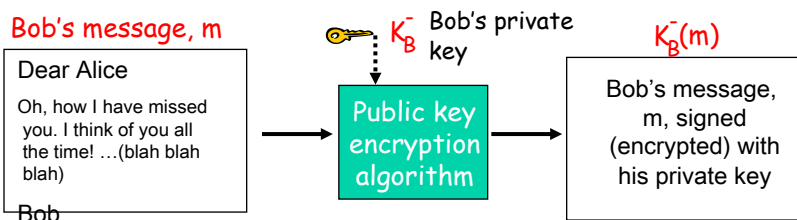
**Cryptographic technique analogous to hand-written signatures.**

- ❑ sender (Bob) digitally signs document, establishing he is document owner/creator.
- ❑ Goal is similar to that of a MAC, except now use public-key cryptography
- ❑ **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 $m$ :

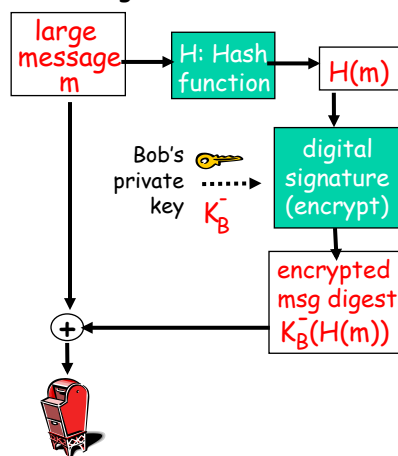
- Bob signs  $m$  by encrypting with his private key  $K_B^-$ , creating "signed" message,  $K_B^-(m)$



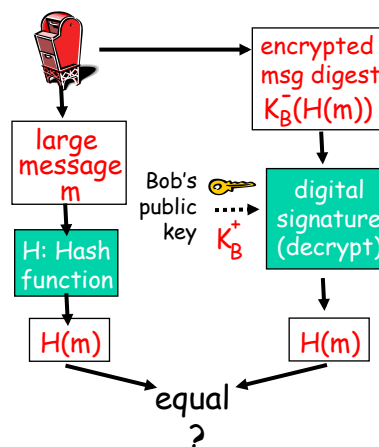
53

## Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



54

## 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  $m$ .

55

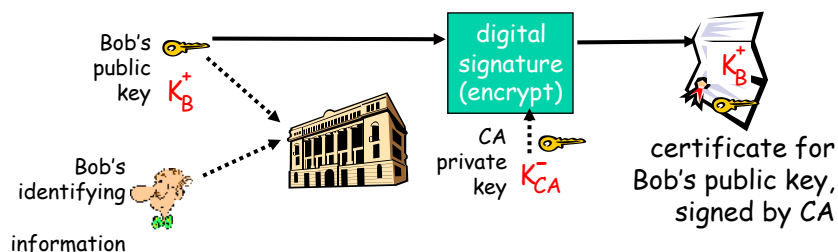
## Public-key certification

- 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 pizzas to Bob.
  - Bob doesn't even like Pepperoni

56

## Certification Authorities

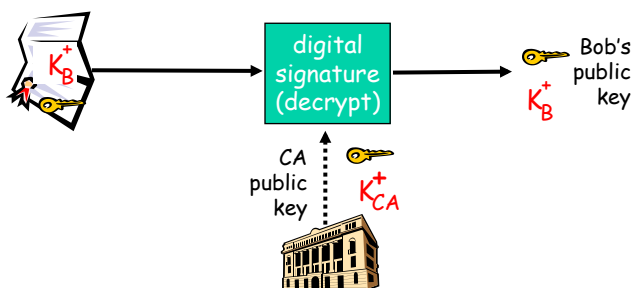
- **Certification authority (CA):** binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA - CA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



57

## Certification Authorities

- 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



58

## Certificates: summary

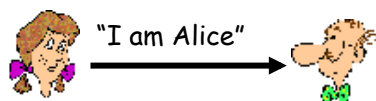
- ❑ Primary standard X.509 (RFC 2459)
- ❑ Certificate contains:
  - Issuer name
  - Entity name, address, domain name, etc.
  - Entity's public key
  - Digital signature (signed with issuer's private key)
- ❑ Public-Key Infrastructure (PKI)
  - Certificates and certification authorities
  - Often considered "heavy"

59

## Authentication

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

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



Failure scenario??



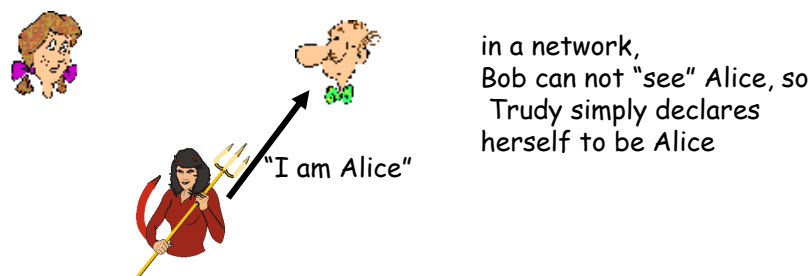
8: Network Security

8-60

## Authentication

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

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

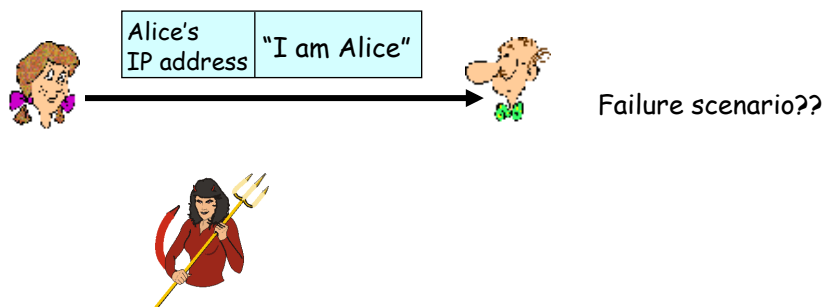


8: Network Security

8-61

## Authentication: another try

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

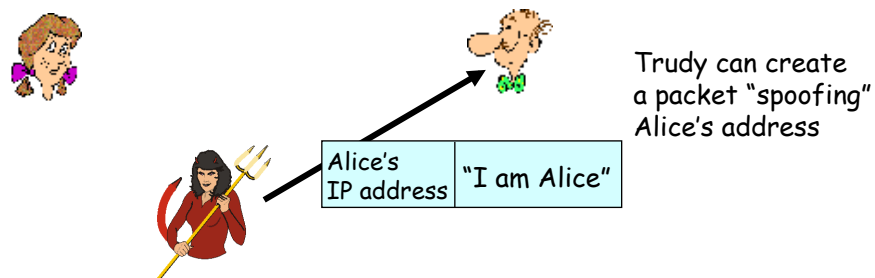


8: Network Security

8-62

## Authentication: another try

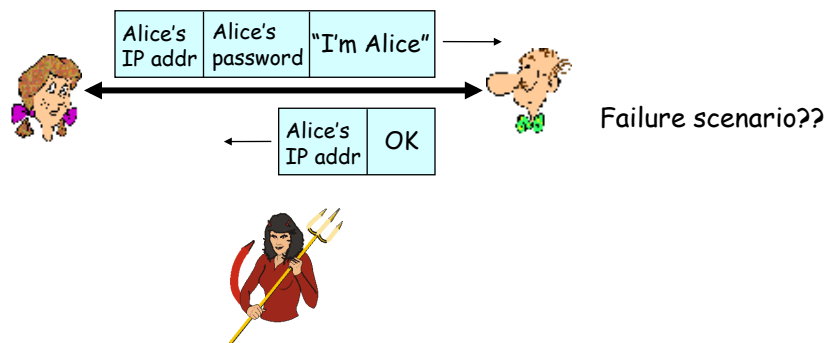
**Protocol ap2.0:** Alice says "I am Alice" in an IP packet containing her source IP address



8: Network Security 8-63

## Authentication: another try

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

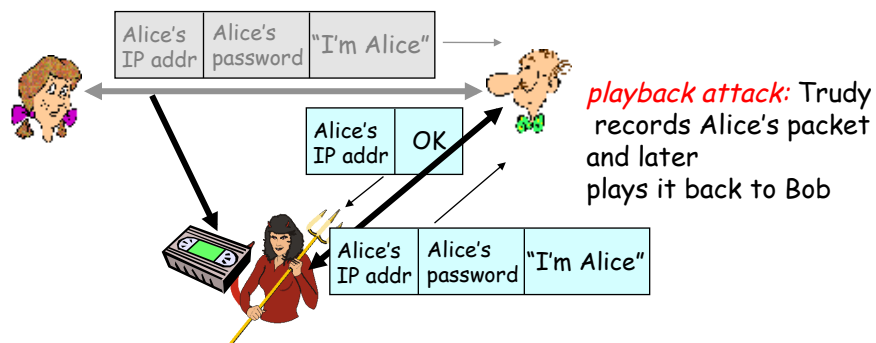


8: Network Security 8-64



## Authentication: another try

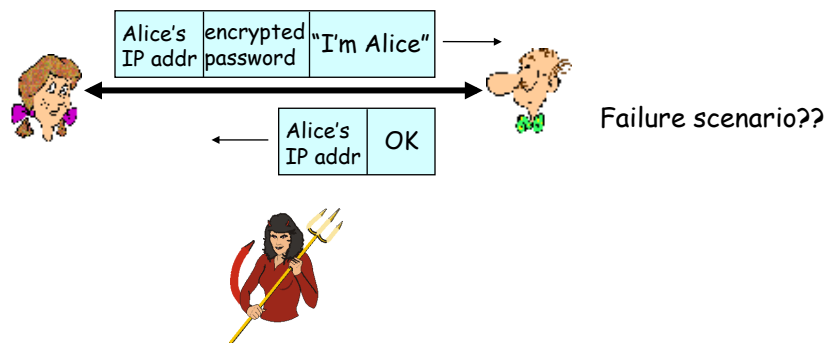
**Protocol ap3.0:** Alice says "I am Alice" and sends her secret password to "prove" it.



8: Network Security 8-65

## Authentication: yet another try

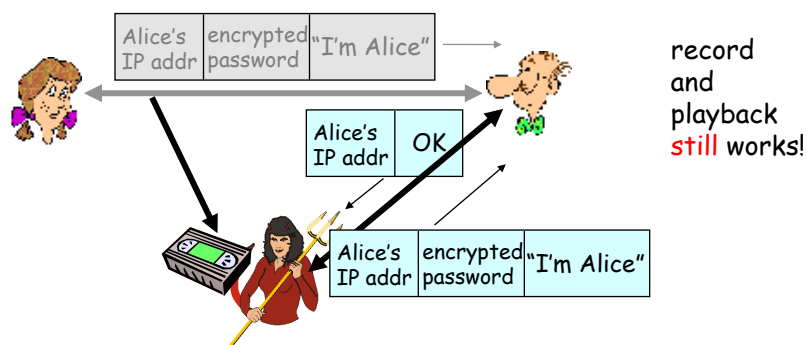
**Protocol ap3.1:** Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



8: Network Security 8-66

## Authentication: another try

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



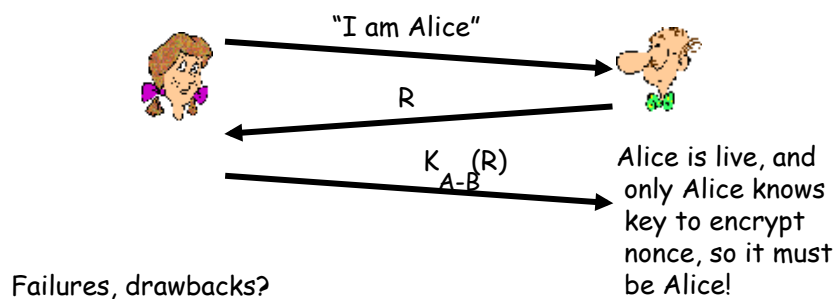
8: Network Security 8-67

## 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



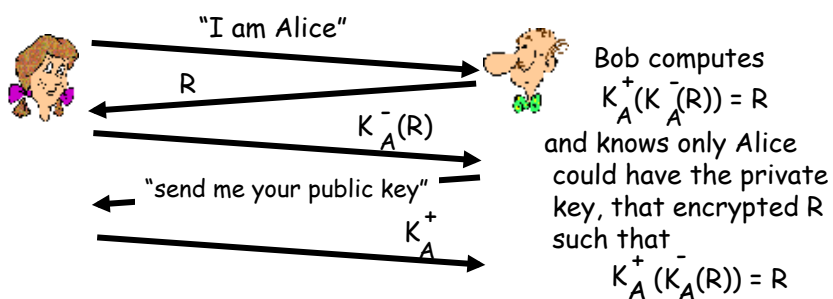
8: Network Security 8-68

## Authentication: ap5.0

ap4.0 requires shared symmetric key

□ can we authenticate using public key techniques?

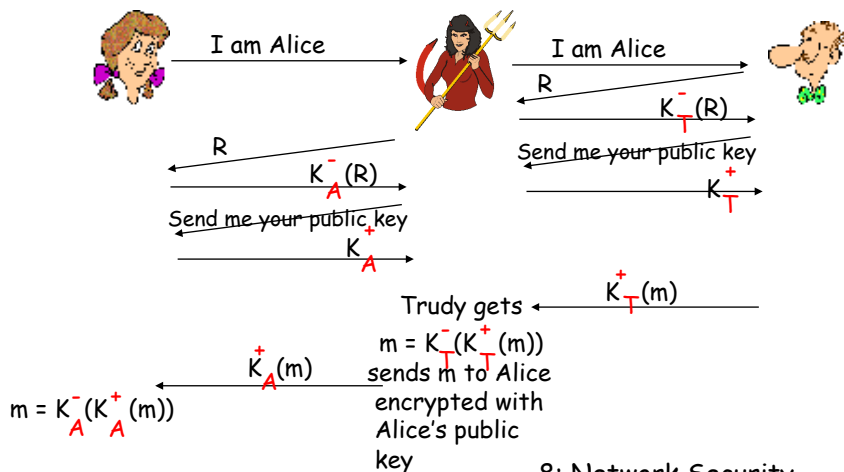
**ap5.0:** use nonce, public key cryptography



8: Network Security 8-69

## ap5.0: security hole

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



8: Network Security 8-70

## 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!

8: Network Security

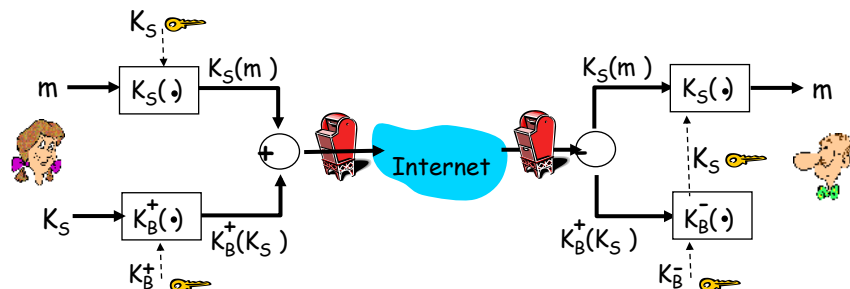
8-71

## Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 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

## Secure e-mail

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

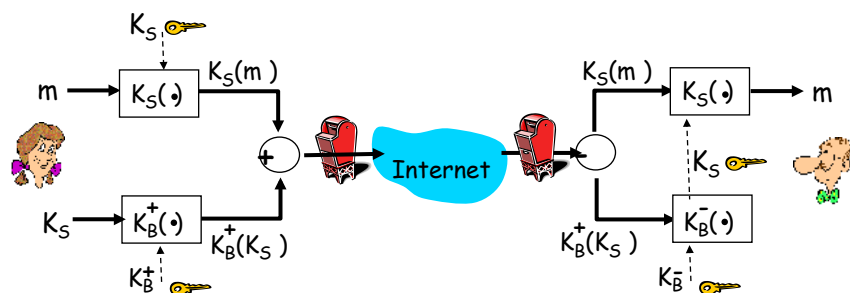


### Alice:

- generates random *symmetric* private key,  $K_S$ .
- encrypts message with  $K_S$  (for efficiency)
- also encrypts  $K_S$  with Bob's public key.
- sends both  $K_S(m)$  and  $K_B(K_S)$  to Bob.

## Secure e-mail

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

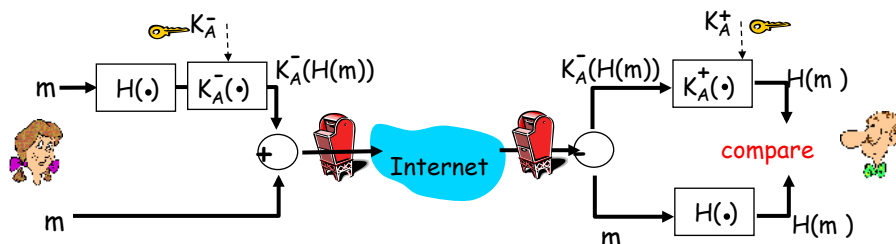


### Bob:

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

## Secure e-mail (continued)

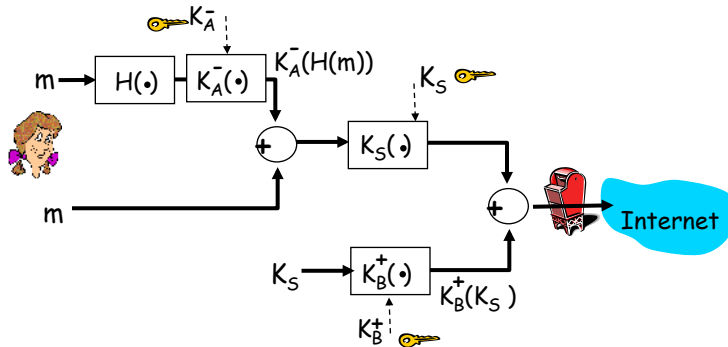
- Alice wants to provide sender authentication message integrity.



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

## Secure e-mail (continued)

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



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

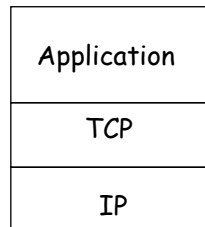
## Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 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

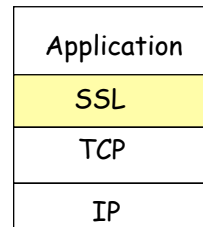
## SSL: Secure Sockets Layer

- Widely deployed security protocol
  - Supported by almost all browsers and web servers
  - https
  - Tens of billions \$ spent per year over SSL
- Originally designed by Netscape in 1993
- Number of variations:
  - TLS: transport layer security, RFC 2246
- Provides
  - Confidentiality
  - Integrity
  - Authentication
- Original goals:
  - Had Web e-commerce transactions in mind
  - 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

## 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

79

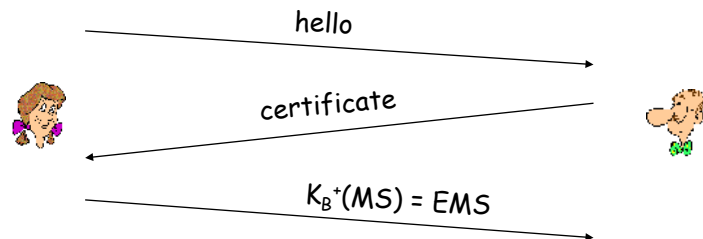
## Toy SSL: a simple secure channel

- Handshake: Alice and Bob use their certificates and 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 a series of records
- Connection Closure: Special messages to securely close connection

80



## Toy: A simple handshake



- MS = master secret
- EMS = encrypted master secret

81

## 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
- Four keys:
  - $K_c$  = encryption key for data sent from client to server
  - $M_c$  = MAC key for data sent from client to server
  - $K_s$  = encryption key for data sent from server to client
  - $M_s$  = 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

82

## Toy: Data Records

- ❑ 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.
  - For example, with instant messaging, how can we do integrity check over all bytes sent before displaying?
- ❑ 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



83

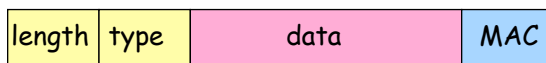
## Toy: Sequence Numbers

- ❑ Attacker can capture and replay record or re-order records
- ❑ Solution: put sequence number into MAC:
  - $MAC = MAC(M_x, \text{sequence} || \text{data})$
  - Note: no sequence number field
- ❑ Attacker could still replay all of the records
  - Use random nonce

84

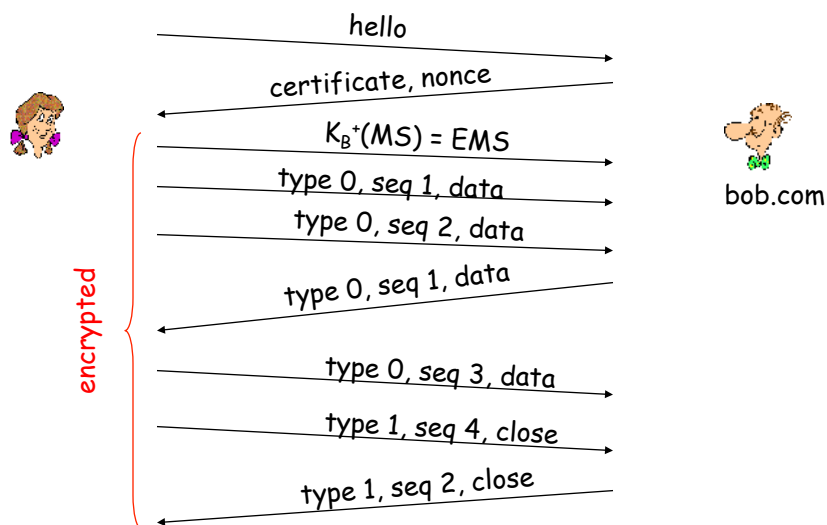
## Toy: Control information

- Truncation attack:
  - attacker forges TCP connection close segment
  - One or both sides thinks there is less data than there actually is.
- Solution: record types, with one type for closure
  - type 0 for data; type 1 for closure
- $MAC = MAC(M_x, \text{sequence} || \text{type} || \text{data})$



85

## Toy SSL: summary



86

## Toy SSL isn't complete

- ❑ How long are the fields?
- ❑ What encryption protocols?
- ❑ No negotiation
  - Allow client and server to support different encryption algorithms
  - Allow client and server to choose together specific algorithm before data transfer

87

## Most common symmetric ciphers in SSL

- ❑ DES - Data Encryption Standard: block
- ❑ 3DES - Triple strength: block
- ❑ RC2 - Rivest Cipher 2: block
- ❑ RC4 - Rivest Cipher 4: stream

### Public key encryption

- ❑ RSA

88

## SSL Cipher Suite

- ❑ Cipher Suite
  - Public-key algorithm
  - Symmetric encryption algorithm
  - MAC algorithm
- ❑ SSL supports a variety of cipher suites
- ❑ Negotiation: client and server must agree on cipher suite
- ❑ Client offers choice; server picks one

89

## Real SSL: Handshake (1)

### Purpose

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

90

## Real SSL: Handshake (2)

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

91

## Real SSL: Handshaking (3)

Last 2 steps protect handshake from tampering

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

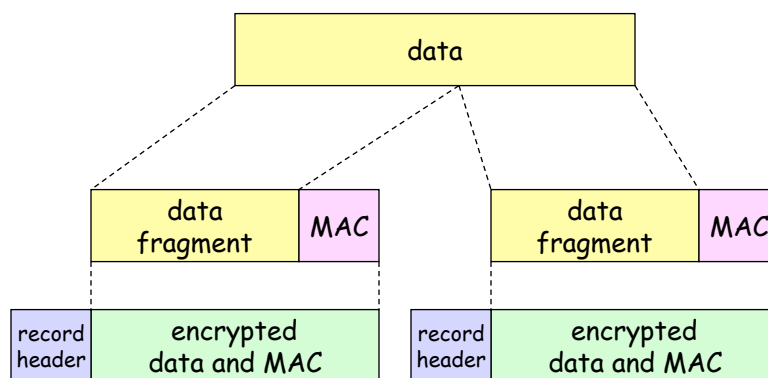
92

## Real SSL: Handshaking (4)

- ❑ Why the two random nonces?
- ❑ Suppose Trudy sniffs all messages between Alice & Bob.
- ❑ Next day, Trudy sets up TCP connection with Bob, sends the 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.

93

## SSL Record Protocol



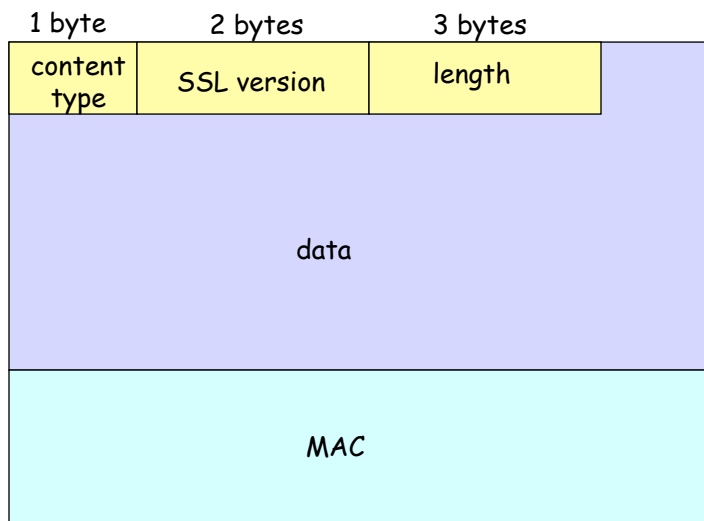
**record header:** content type; version; length

**MAC:** includes sequence number, MAC key  $M_x$

**Fragment:** each SSL fragment  $2^{14}$  bytes (~16 Kbytes)

94

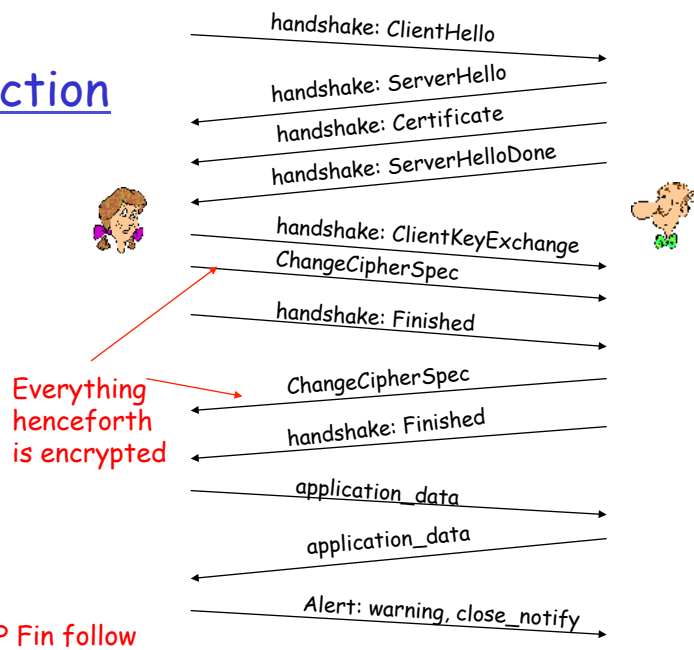
## SSL Record Format



Data and MAC encrypted (symmetric algo)

95

## Real Connection



96



## Key derivation

- Client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - Produces master secret
- Master secret and new nonces input into another random-number generator: "key block"
  - Because of resumption: TBD
- Key block sliced and diced:
  - client MAC key
  - server MAC key
  - client encryption key
  - server encryption key
  - client initialization vector (IV)
  - server initialization vector (IV)

97

## Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 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 confidentiality at the network-layer?

### Between two network entities:

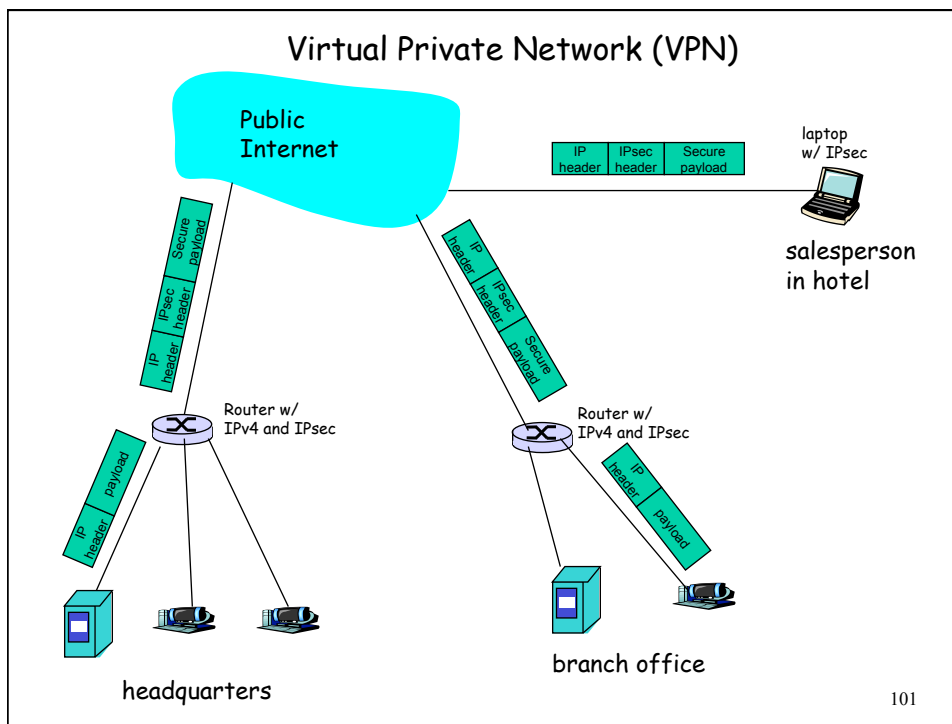
- ❑ Sending entity encrypts the payloads of datagrams. Payload could be:
  - TCP segment, UDP segment, ICMP message, OSPF message, and so on.
- ❑ All data sent from one entity to the other would be hidden:
  - Web pages, e-mail, P2P file transfers, TCP SYN packets, and so on.
- ❑ That is, "blanket coverage".

99

## Virtual Private Networks (VPNs)

- ❑ Institutions often want private networks for security.
  - Costly! Separate routers, links, DNS infrastructure.
- ❑ With a VPN, institution's inter-office traffic is sent over public Internet instead.
  - But inter-office traffic is encrypted before entering public Internet

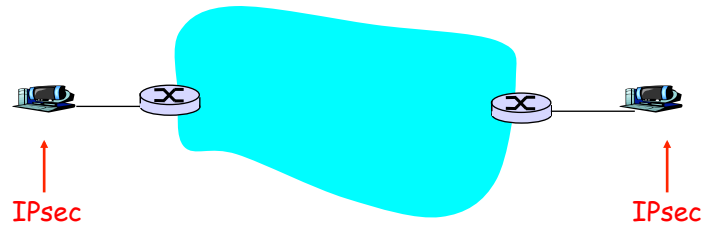
100



## IPsec services

- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality
  
- Two protocols providing different service models:
  - AH
  - ESP

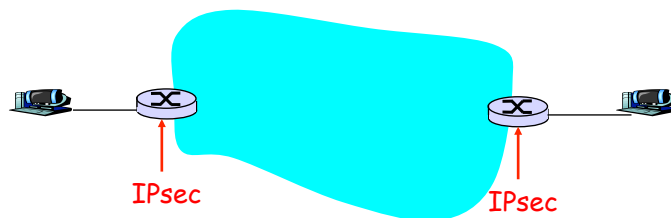
## IPsec Transport Mode



- ❑ IPsec datagram emitted and received by end-system.
- ❑ Protects upper level protocols

103

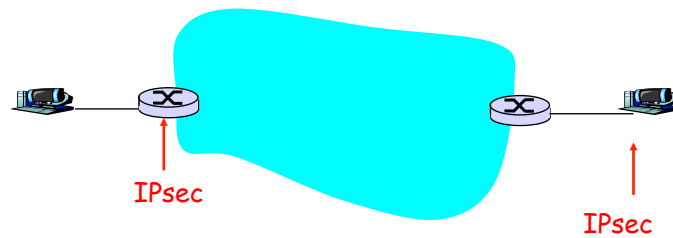
## IPsec - tunneling mode (1)



- ❑ End routers are IPsec aware. Hosts need not be.

104

## IPsec - tunneling mode (2)



- Also tunneling mode.

105

## Two protocols

- Authentication Header (AH) protocol
  - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP)
  - provides source authentication, data integrity, and confidentiality
  - more widely used than AH

106

## Four combinations are possible!

Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

Most common and  
most important

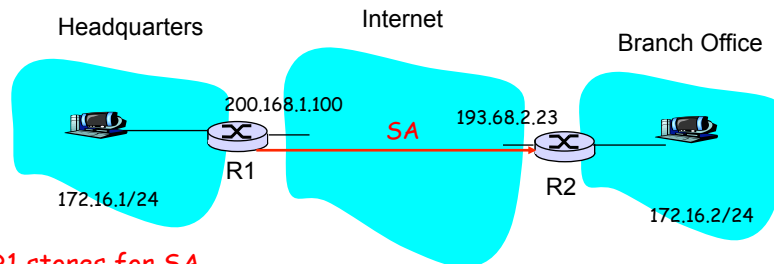
107

## Security associations (SAs)

- ❑ Before sending data, a virtual connection is established from sending entity to receiving entity.
- ❑ Called "security association (SA)"
  - SAs are simplex: for only one direction
- ❑ Both sending and receiving entities maintain *state information* about the SA
  - Recall that TCP endpoints also maintain state information.
  - IP is connectionless; IPsec is connection-oriented!
- ❑ How many SAs in VPN w/ headquarters, branch office, and n traveling salesperson?

108

## Example SA from R1 to R2



### R1 stores for SA

- ❑ 32-bit identifier for SA: *Security Parameter Index (SPI)*
- ❑ the origin interface of the SA (200.168.1.100)
- ❑ destination interface of the SA (193.68.2.23)
- ❑ type of encryption to be used (for example, 3DES with CBC)
- ❑ encryption key
- ❑ type of integrity check (for example, HMAC with MD5)
- ❑ authentication key

109

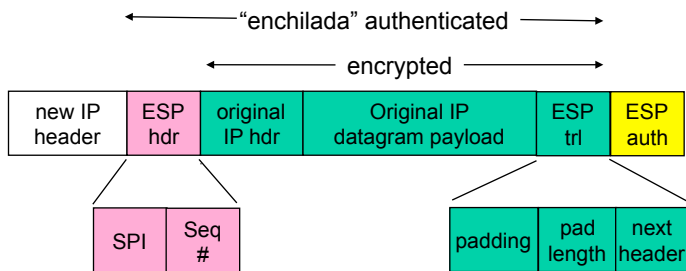
## Security Association Database (SAD)

- ❑ Endpoint holds state of its SAs in a SAD, where it can locate them during processing.
- ❑ When sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- ❑ When IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

110

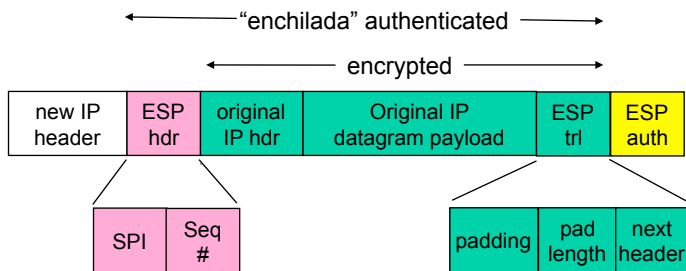
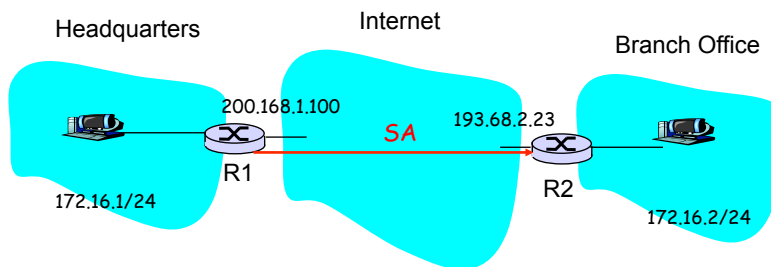
## IPsec datagram

Focus for now on tunnel mode with ESP



111

## What happens?



112

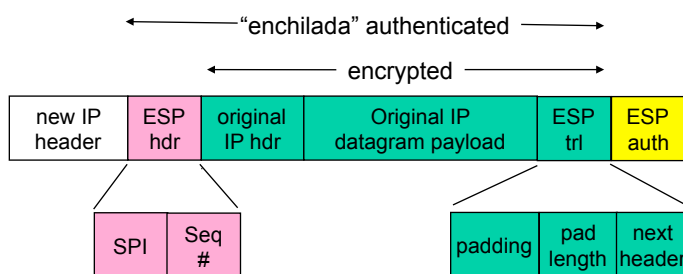


## R1 converts original datagram into IPsec datagram

- ❑ Appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- ❑ Encrypts result using algorithm & key specified by SA.
- ❑ Appends to front of this encrypted quantity the "ESP header, creating "enchilada".
- ❑ Creates authentication MAC over the *whole enchilada*, using algorithm and key specified in SA;
- ❑ Appends MAC to back of enchilada, forming *payload*;
- ❑ Creates brand new IP header, with all the classic IPv4 header fields, which it appends before payload.

113

## Inside the enchilada:



- ❑ ESP trailer: Padding for block ciphers
- ❑ ESP header:
  - SPI, so receiving entity knows what to do
  - Sequence number, to thwart replay attacks
- ❑ MAC in ESP auth field is created with shared secret key

114

## IPsec sequence numbers

- ❑ For new SA, sender initializes seq. # to 0
- ❑ Each time datagram is sent on SA:
  - Sender increments seq # counter
  - Places value in seq # field
- ❑ Goal:
  - Prevent attacker from sniffing and replaying a packet
    - Receipt of duplicate, authenticated IP packets may disrupt service
- ❑ Method:
  - Destination checks for duplicates
  - But doesn't keep track of ALL received packets; instead uses a window

115

## Security Policy Database (SPD)

- ❑ Policy: For a given datagram, sending entity needs to know if it should use IPsec.
- ❑ Needs also to know which SA to use
  - May use: source and destination IP address; protocol number.
- ❑ Info in SPD indicates "what" to do with arriving datagram;
- ❑ Info in the SAD indicates "how" to do it.

116

## Summary: IPsec services

- ❑ Suppose Trudy sits somewhere between R1 and R2. She doesn't know the keys.
  - Will Trudy be able to see contents of original datagram? How about source, dest IP address, transport protocol, application port?
  - Flip bits without detection?
  - Masquerade as R1 using R1's IP address?
  - Replay a datagram?

117

## Internet Key Exchange

- ❑ In previous examples, we manually established IPsec SAs in IPsec endpoints:

### Example SA

SPI: 12345  
 Source IP: 200.168.1.100  
 Dest IP: 193.68.2.23  
 Protocol: ESP  
 Encryption algorithm: 3DES-cbc  
 HMAC algorithm: MD5  
 Encryption key: 0x7aeaca...  
 HMAC key: 0xc0291f...

- ❑ Such manually keying is impractical for large VPN with, say, hundreds of sales people.
- ❑ Instead use *IPsec IKE (Internet Key Exchange)*

118

## IKE: PSK and PKI

- ❑ Authentication (proof who you are) with either
  - pre-shared secret (PSK) or
  - with PKI (public/private keys and certificates).
- ❑ With PSK, both sides start with secret:
  - then run IKE to authenticate each other and to generate IPsec SAs (one in each direction), including encryption and authentication keys
- ❑ With PKI, both sides start with public /private key pair and certificate.
  - run IKE to authenticate each other and obtain IPsec SAs (one in each direction).
  - Similar with handshake in SSL.

119

## Summary of IPsec

- ❑ IKE message exchange for algorithms, secret keys, SPI numbers
- ❑ Either the AH or the ESP protocol (or both)
- ❑ The AH protocol provides integrity and source authentication
- ❑ The ESP protocol (with AH) additionally provides encryption
- ❑ IPsec peers can be two end systems, two routers /firewalls, or a router/firewall and an end system

120

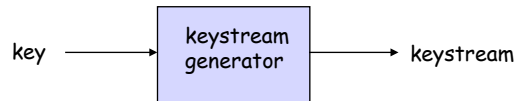
## Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 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

## WEP Design Goals

- Symmetric key crypto
  - Confidentiality
  - Station authorization
  - Data integrity
- Self synchronizing: each packet separately encrypted
  - Given encrypted packet and key, can decrypt; can continue to decrypt packets when preceding packet was lost
  - Unlike Cipher Block Chaining (CBC) in block ciphers
- Efficient
  - Can be implemented in hardware or software

## Review: Symmetric Stream Ciphers

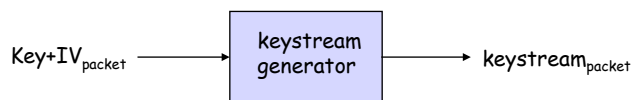


- ❑ Combine each byte of keystream with byte of plaintext to get ciphertext
- ❑  $m(i)$  =  $i$ th unit of message
- ❑  $ks(i)$  =  $i$ th unit of keystream
- ❑  $c(i)$  =  $i$ th unit of ciphertext
- ❑  $c(i) = ks(i) \oplus m(i)$  ( $\oplus$  = exclusive or)
- ❑  $m(i) = ks(i) \oplus c(i)$
- ❑ WEP uses RC4

123

## Stream cipher and packet independence

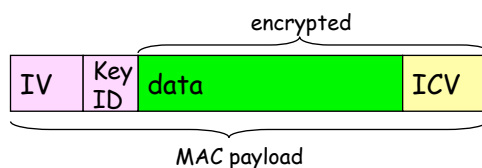
- ❑ Recall design goal: each packet separately encrypted
- ❑ If for frame  $n+1$ , use keystream from where we left off for frame  $n$ , then each frame is not separately encrypted
  - Need to know where we left off for packet  $n$
- ❑ WEP approach: initialize keystream with key + new IV for each packet:



124

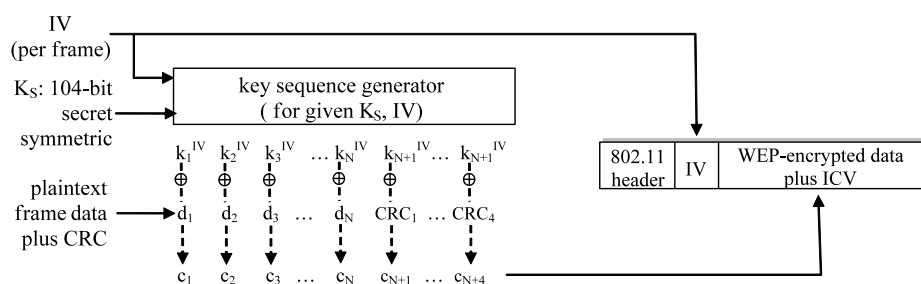
## WEP encryption (1)

- ❑ Sender calculates Integrity Check Value (ICV) over data
  - four-byte hash/CRC for data integrity
- ❑ Each side has 104-bit shared key
- ❑ Sender creates 24-bit initialization vector (IV), appends to key: gives 128-bit key
- ❑ Sender also appends keyID (in 8-bit field)
- ❑ 128-bit key inputted into pseudo random number generator to get keystream
- ❑ data in frame + ICV is encrypted with RC4:
  - Bytes of keystream are XORed with bytes of data & ICV
  - IV & keyID are appended to encrypted data to create payload
  - Payload inserted into 802.11 frame



125

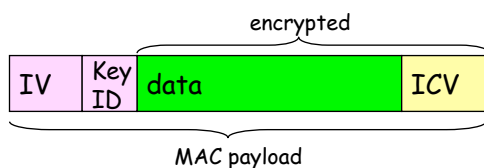
## WEP encryption (2)



**New IV for each frame**

126

## WEP decryption overview



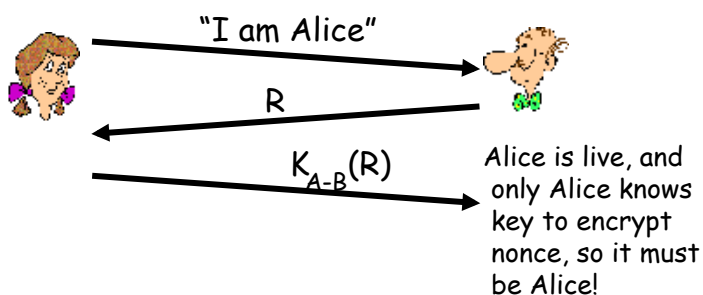
- ❑ Receiver extracts IV
- ❑ Inputs IV and shared secret key into pseudo random generator, gets keystream
- ❑ XORs keystream with encrypted data to decrypt data + ICV
- ❑ Verifies integrity of data with ICV
  - Note that message integrity approach used here is different from the MAC (message authentication code) and signatures (using PKI).

127

## End-point authentication w/ nonce

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

**How:** to prove Alice "live", Bob sends Alice **nonce**, R. Alice must return R, encrypted with shared secret key

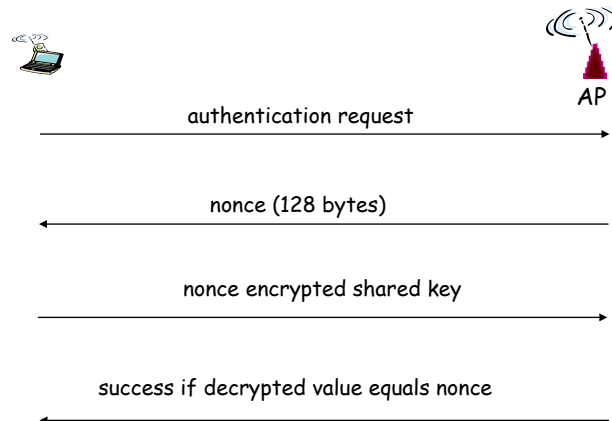


128



## WEP Authentication

*Not all APs do it, even if WEP is being used. AP indicates if authentication is necessary in beacon frame. Done before association.*



129

## Breaking 802.11 WEP encryption

### security hole:

- ❑ 24-bit IV, one IV per frame, -> IV's eventually reused
- ❑ IV transmitted in plaintext -> IV reuse detected

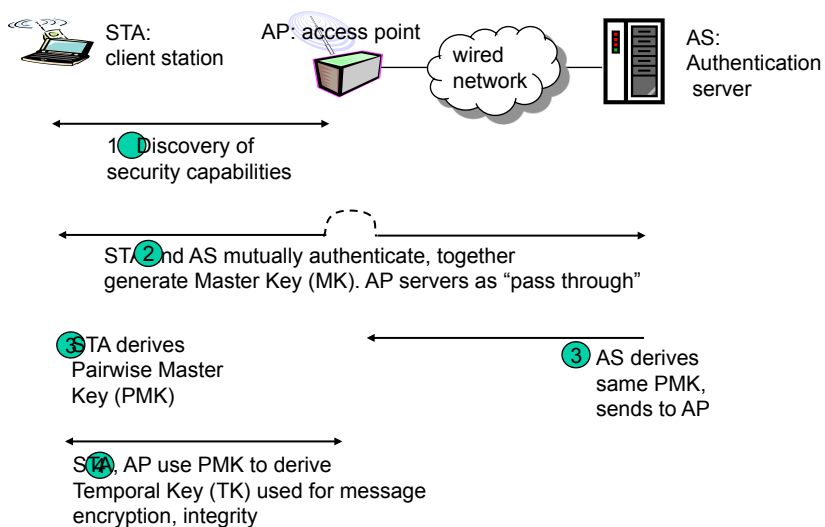
### ❑ attack:

- Trudy causes Alice to encrypt known plaintext  $d_1 d_2 d_3 d_4 \dots$
- Trudy sees:  $c_i = d_i \text{ XOR } k_i^{\text{IV}}$
- Trudy knows  $c_i d_i$ , so can compute  $k_i^{\text{IV}}$
- Trudy knows encrypting key sequence  $k_1^{\text{IV}} k_2^{\text{IV}} k_3^{\text{IV}} \dots$
- Next time IV is used, Trudy can decrypt!

## 802.11i: improved security

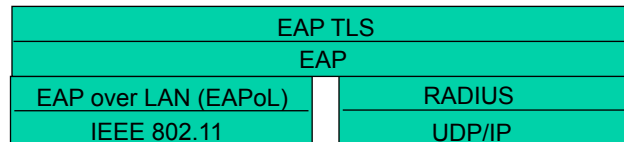
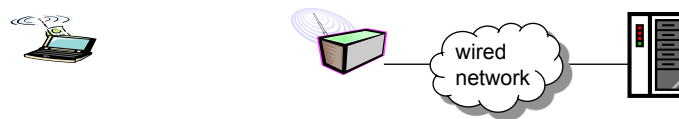
- ❑ numerous (stronger) forms of encryption possible
- ❑ provides key distribution
- ❑ uses authentication server separate from access point

## 802.11i: four phases of operation



## EAP: extensible authentication protocol

- ❑ EAP: end-end client (mobile) to authentication server protocol
- ❑ EAP sent over separate "links"
  - mobile-to-AP (EAP over LAN)
  - AP to authentication server (RADIUS over UDP)



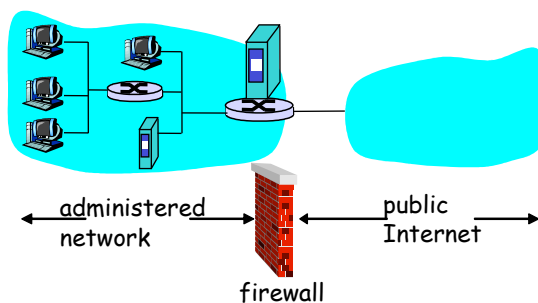
## Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 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

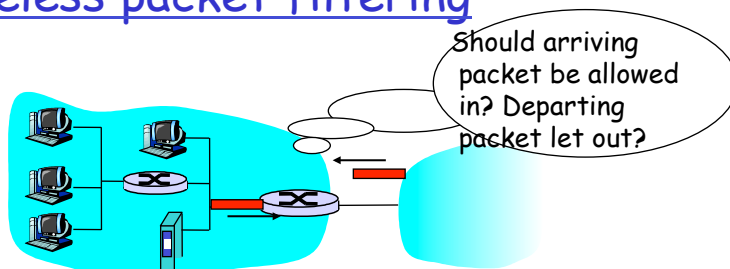
## Firewalls

### firewall

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



## 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.
  - all incoming, outgoing UDP flows and telnet connections are blocked.
- ❑ example 2: Block inbound TCP segments with ACK=0.
  - prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

## Stateless packet filtering: more examples

<u>Policy</u>	<u>Firewall Setting</u>
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 (eg 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

- 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): can determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets

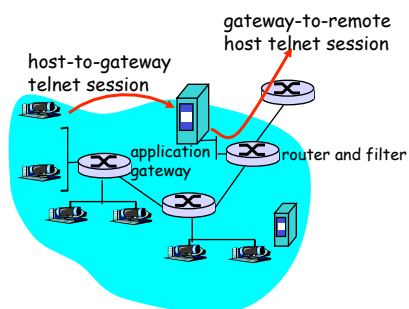
## Stateful packet filtering

- 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	×
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	

## Application gateways

- filters packets on application data as well as on IP/TCP/UDP fields.
- example:** allow select internal users to telnet outside.
  - require all telnet users to telnet through gateway.
  - for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
  - router filter blocks all telnet connections not originating from gateway.



## Limitations of firewalls and gateways

- **IP spoofing:** router can't know if data "really" comes from claimed source
- if multiple app's. need special treatment, each has own app. gateway.
- client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP.
- tradeoff: **degree of communication with outside world, level of security**
- many highly protected sites still suffer from attacks.

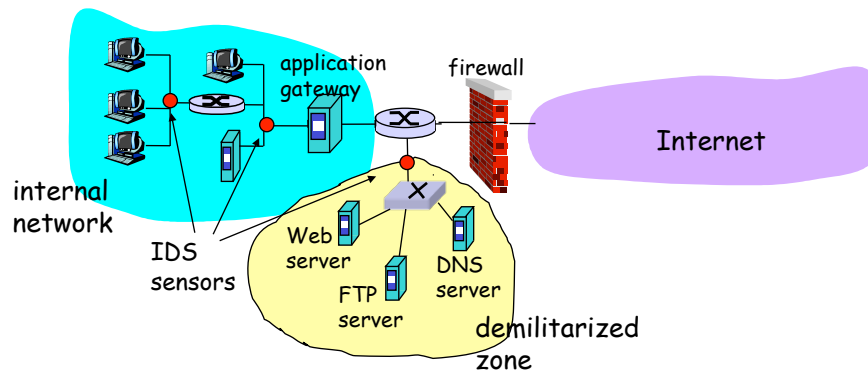
## Intrusion detection systems

- 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

- multiple IDSs: different types of checking at different locations



## 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)
- IP sec
- 802.11

### Operational Security: firewalls and IDS