

Chapter 8

Security

A note on the use of these PowerPoint slides:

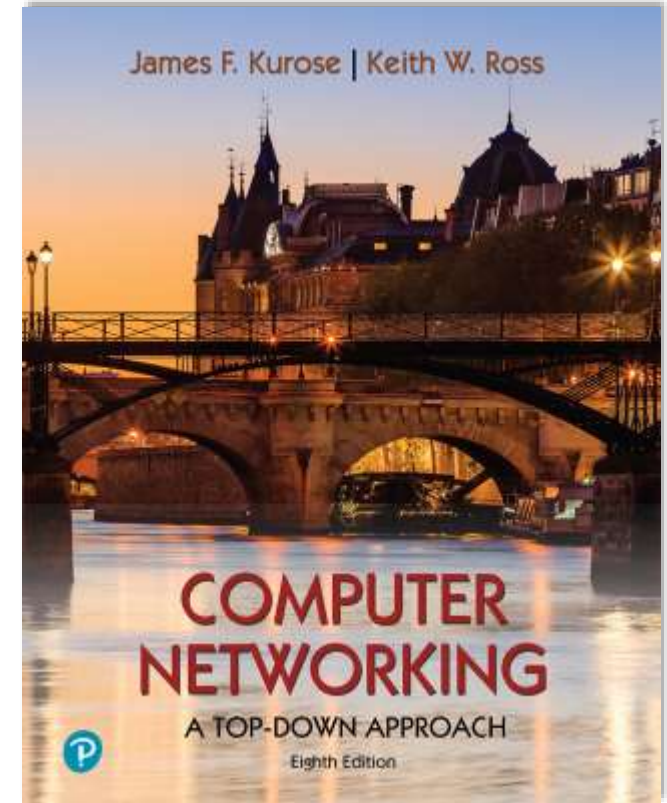
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2023
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking: A
Top-Down Approach*

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Security: overview

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 outline

- What is network security?
- Principles of cryptography
- Message integrity, authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



What is network security?

confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

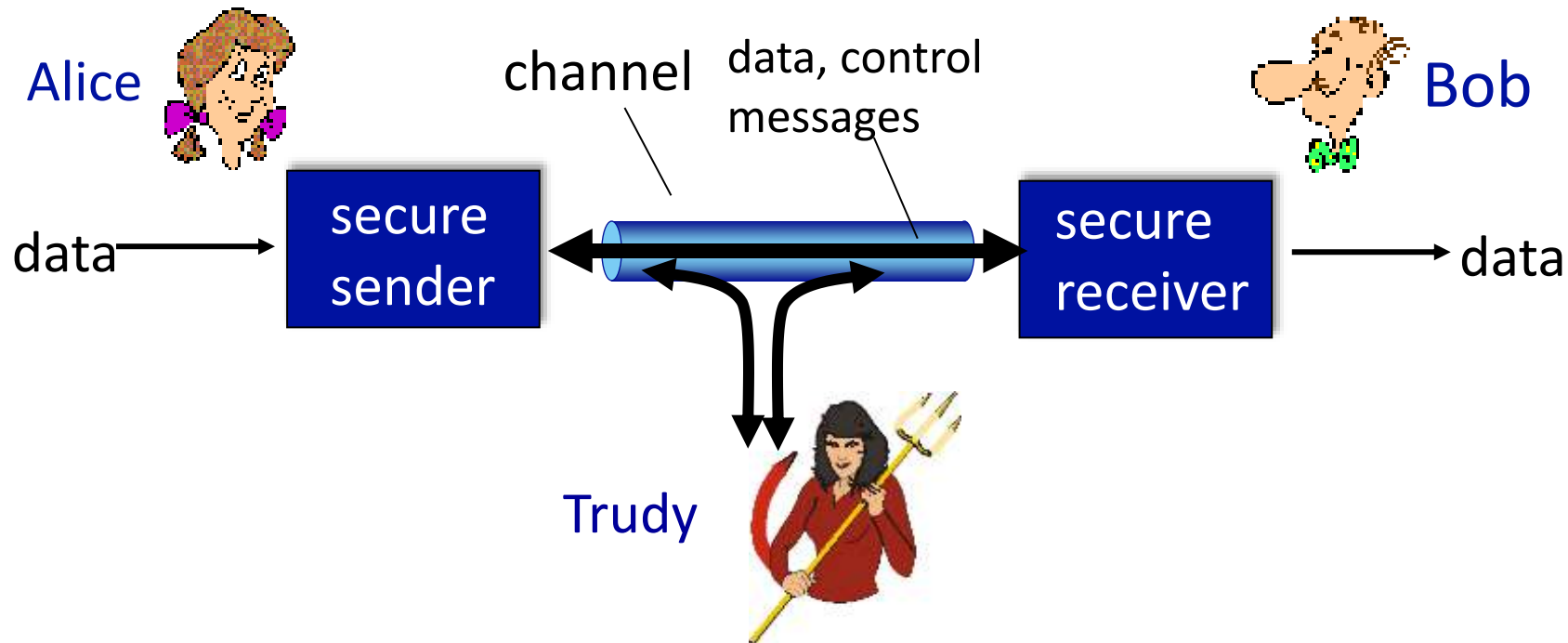
authentication: sender, receiver want to confirm identity of each other

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection; recall checksums

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 want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



Friends and enemies: Alice, Bob, Trudy

Who might Bob and Alice be?

- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- TCP handshake
- DNS servers
- BGP routers exchanging routing table updates

There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

A: A lot!

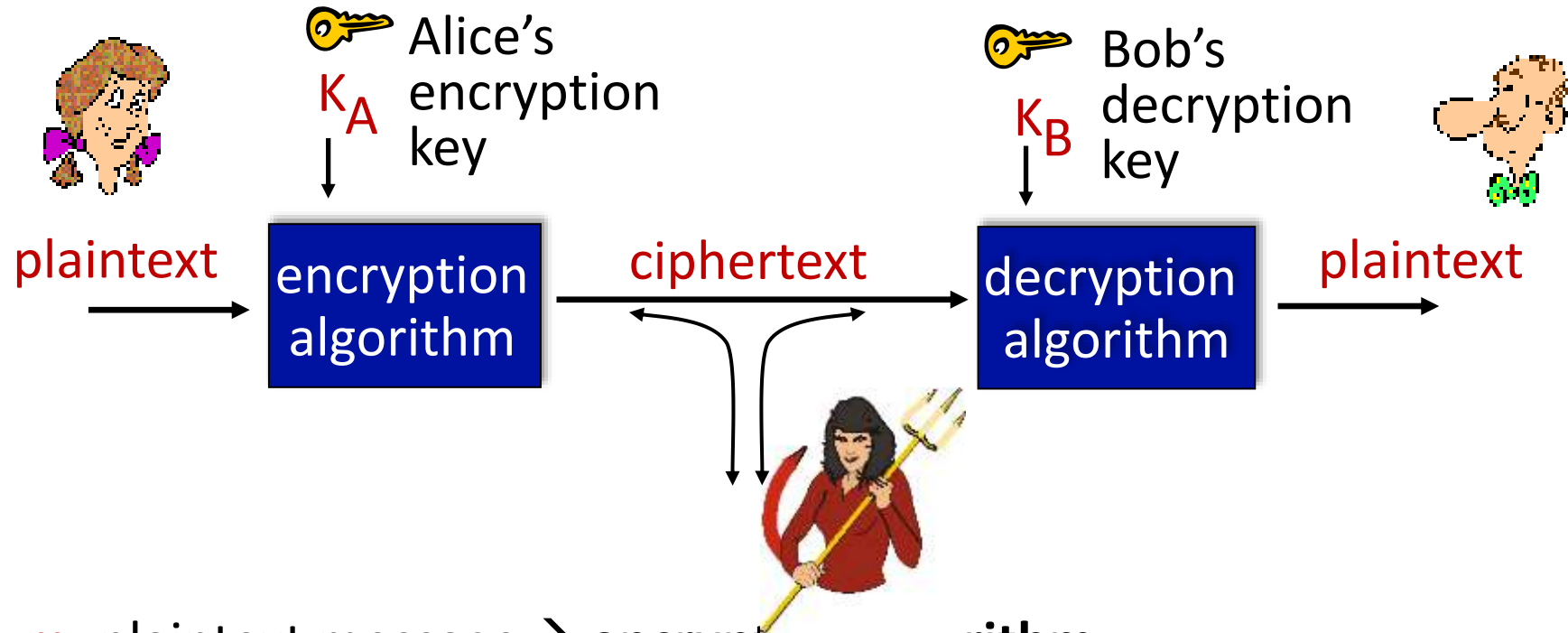
- **eavesdrop**: intercept messages
- actively **insert** messages into connection
- **impersonation**: can fake (spoof) source address in packet (or any field in packet)
- **hijacking**: “take over” ongoing connection by removing sender or receiver, inserting himself in place
- **denial of service**: prevent service from being used by others (e.g., by overloading resources)

Chapter 8 outline

- What is network security?
- **Principles of cryptography**
- Message integrity, authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



The language of cryptography

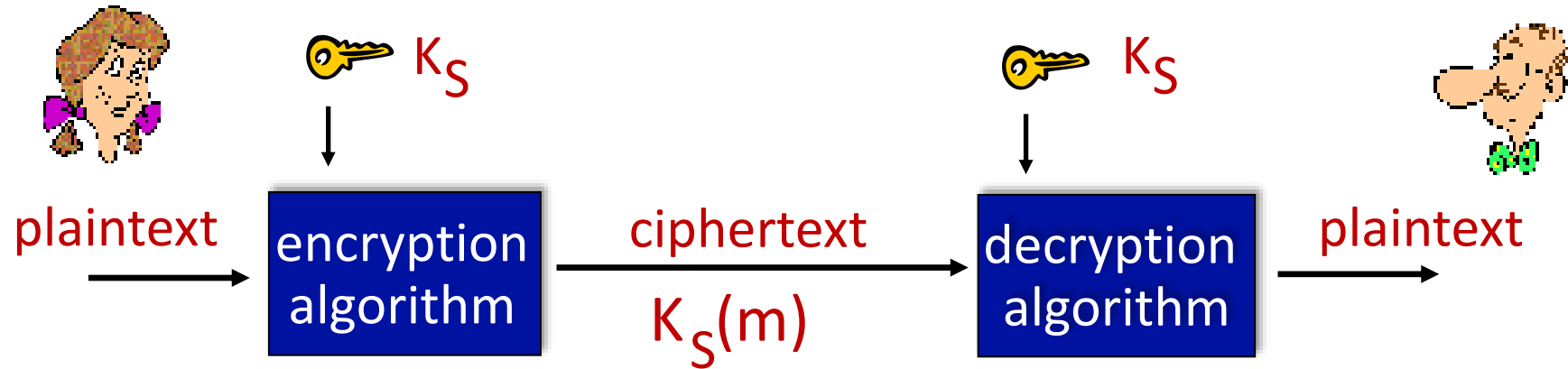


m : plaintext message \rightarrow encryption algorithm

$K_A(m)$: **ciphertext** (encrypted message), encrypted with key K_A

$m = K_B(K_A(m))$

Symmetric key cryptography



symmetric key cryptography: Bob and Alice share same (symmetric) key

- In public key systems, a pair of keys is used
- Concerns with public keys?

Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

plaintext:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
		↓																							↓	
ciphertext:	m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

e.g.: Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc

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

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

Password Complexity – Brute Force Effectiveness

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15 bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100 tn years	7qd years



**TIME IT TAKES
A HACKER TO
BRUTE FORCE
YOUR
PASSWORD**

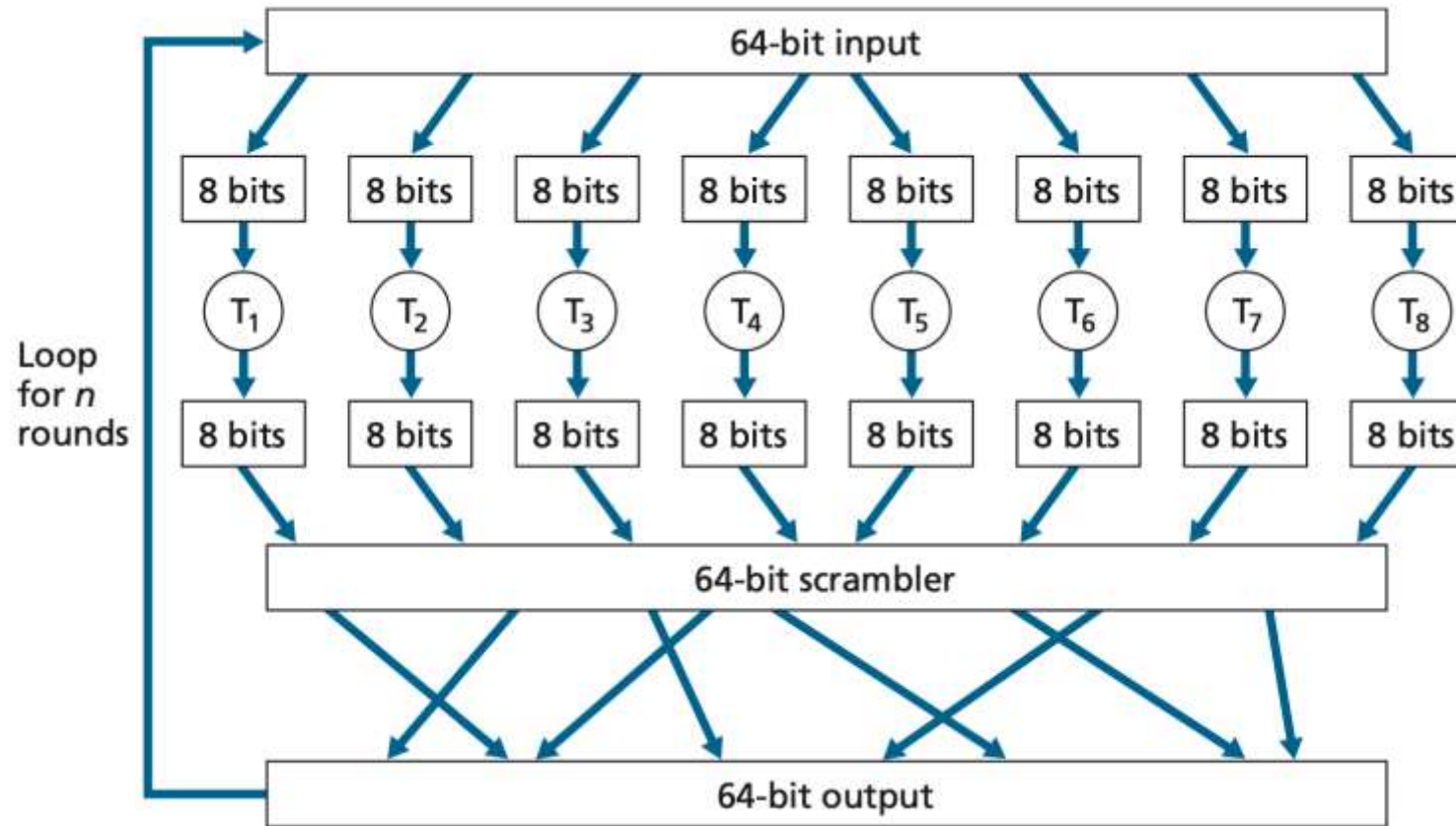


-Data sourced from [HowSecureIsMyPassword.net](https://howsecureismypassword.net)

Block Ciphers – Symmetric Key Encryption

- Block ciphers users in many secure Internet protocols, including PGP (for secure email), TLS (to secure TCP connections) and IPsec (securing network –layer transport).
- The message to be encrypted is processed in blocks of k bits. For example, if $k = 64$, then the message is broken into 64-bit blocks, and each block is encrypted independently
 - Typically use functions that simulate randomly permuted tables

Block Cipher



Symmetric key crypto: DES

DES: Data Encryption Standard

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

AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128-bit blocks
- 128, 192, or 256 bits keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

symmetric key crypto:

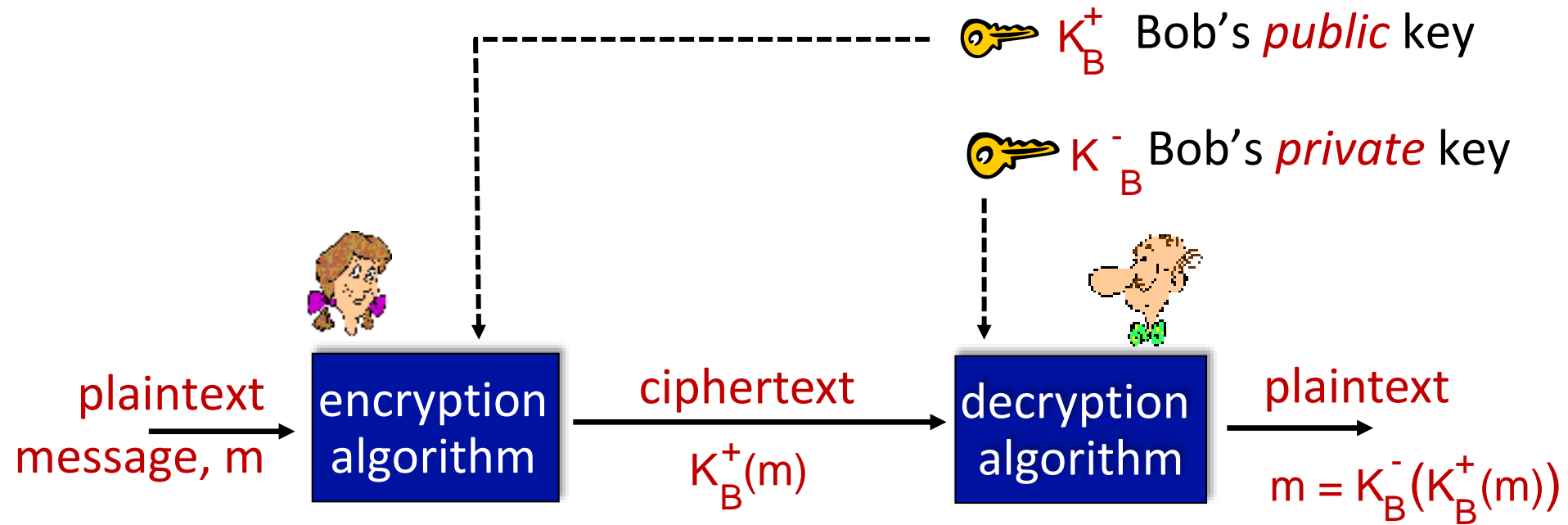
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never “met”)?

public key crypto

- *radically* different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver
- Also! Adds authentication and digital signatures



Public Key Cryptography



Wow - public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- Concerns with public and private key?

Public key encryption algorithms

requirements:

① need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

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

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

RSA: Rivest, Shamir, Adelson algorithm

RSA: Creating public/private key pair

1. choose two large prime numbers p, q . (the larger the better, product to be 1024 bits)
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) .
 $\underbrace{(n, e)}_{K_B^+}$ $\underbrace{(n, d)}_{K_B^-}$

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!

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

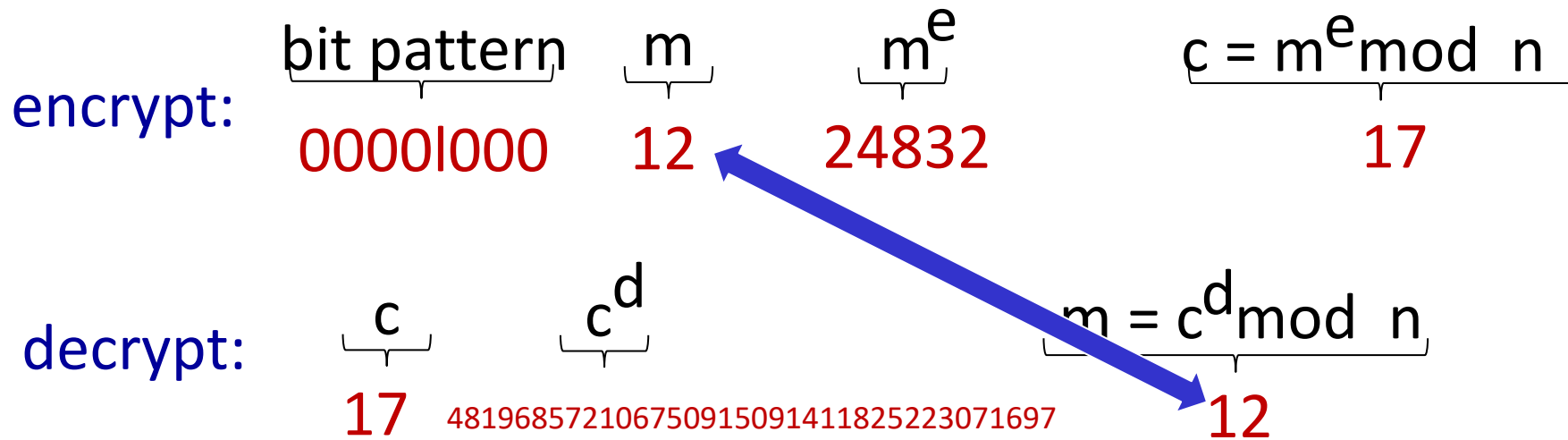
RSA example:

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.



RSA Example

Plaintext message: Love:

Plaintext Letter	m : numeric representation	m^e	Ciphertext $c = m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

Table 8.2 ♦ Alice's RSA encryption, $e = 5$, $n = 35$

RSA Example

Ciphertext c	c^d	$m = c^d \bmod n$	Plaintext Letter
17	4819685721067509150915091411825223071697	12	l
15	127834039403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	10000000000000000000000000000000	5	e

Table 8.3 ♦ Bob's RSA decryption, $d = 29$, $n = 35$

RSA in practice: session keys

- Used with symmetric key encryption in conjunction with asymmetric (RSA)
 - Encrypt the data with the symmetric key cipher(AES) – **session key**
 - Encrypt the session key using public key of the recipient
 - Recipient decrypts using the private key, get the symmetric key, and decrypts the data using the same algorithm.

Diffie-Hellman Algorithm

- Public key encryption algorithm
- Not as versatile as RSA; it can't be used to encrypt messages of arbitrary length
- Can be used to establish symmetric session key, which in turn used to encrypt messages

Chapter 8 outline

- What is network security?
- Principles of cryptography
- **Authentication**, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



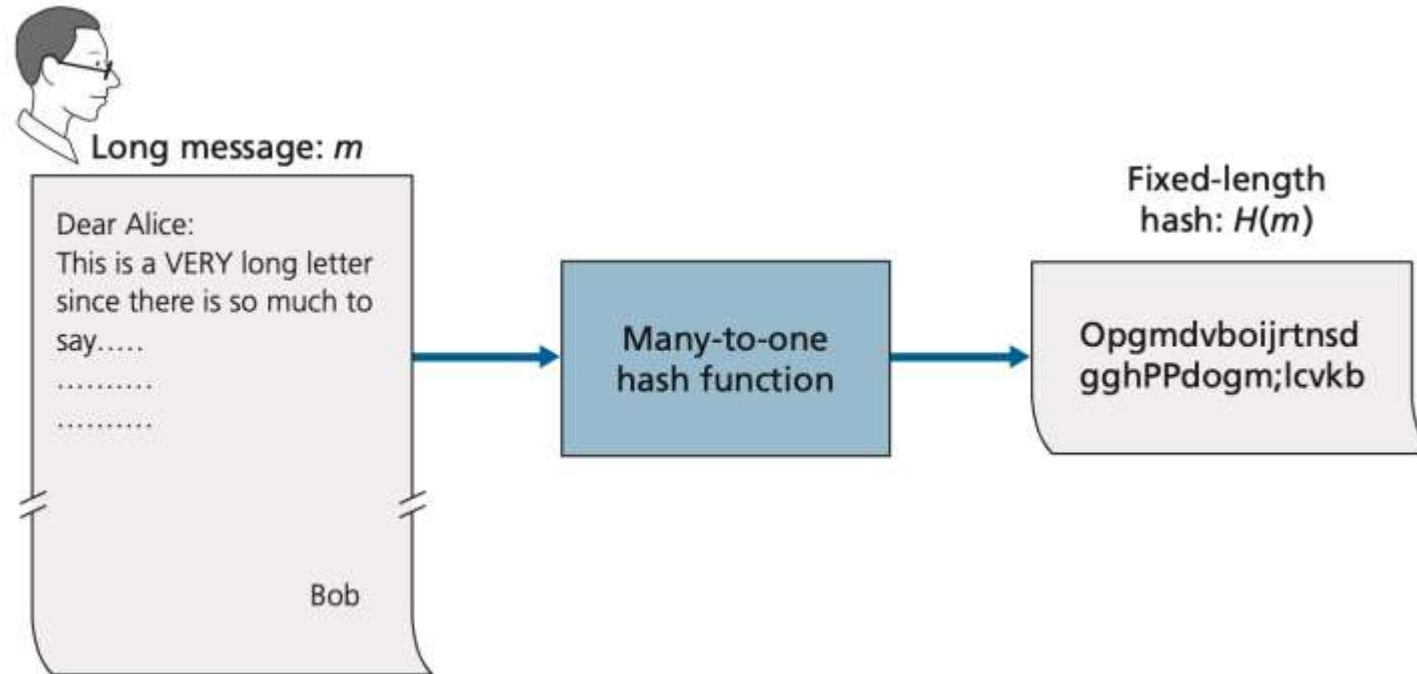
Message Integrity & Digital Signatures

- Previously we addressed confidentiality (encryption)
- Message integrity or authentication:
 - Digital Signatures
 - End-point authentication

Hash Functions

Input m , computes fixed-size string $H(m)$ known as **Hash**

- **Recall Internet checksum, recall CRC meets the criteria only on the surface!**
- Properties: infeasible to find any two different messages x and y such that $H(x) = H(y)$



Hash Functions

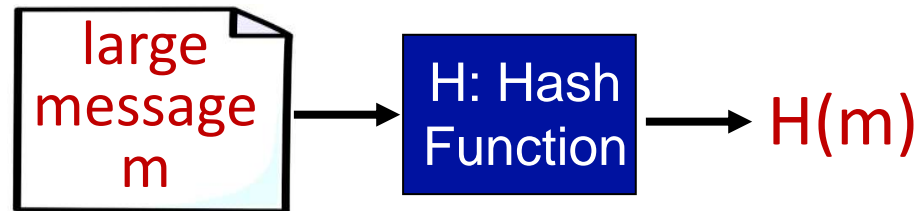
- Intruder can't substitute one message for another message that is protected by the hash function
- $M, H(m)$ can't be equal to $Y, H(m)$.
- **Simple checksum algorithm, such as internet checksum, and CRC will violate this property**
 - We need more powerful hash functions for security purposes

Message digests

computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy- to-compute digital “fingerprint”

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



Hash function properties:


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

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of message
- is many-to-one

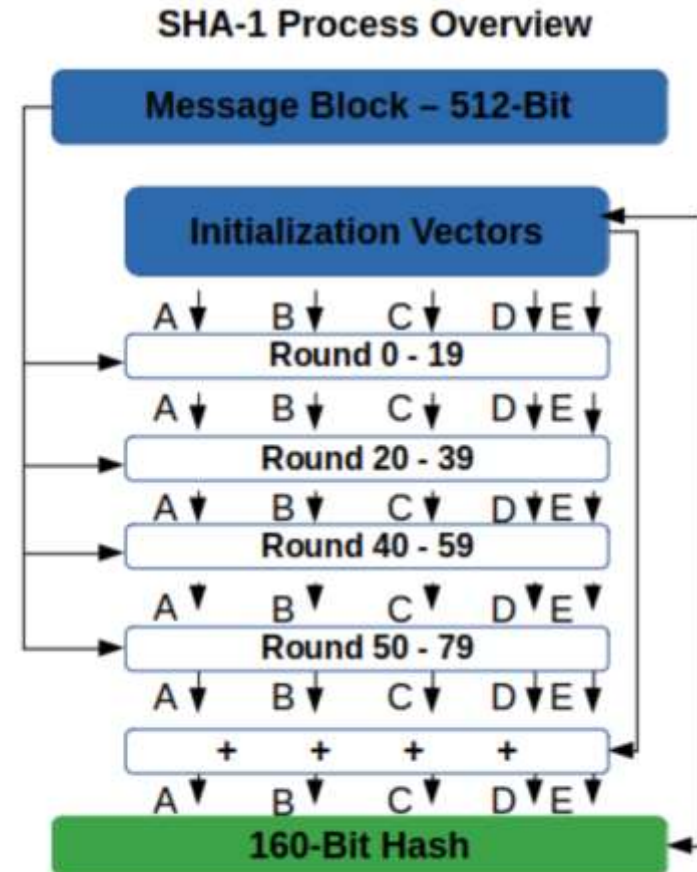
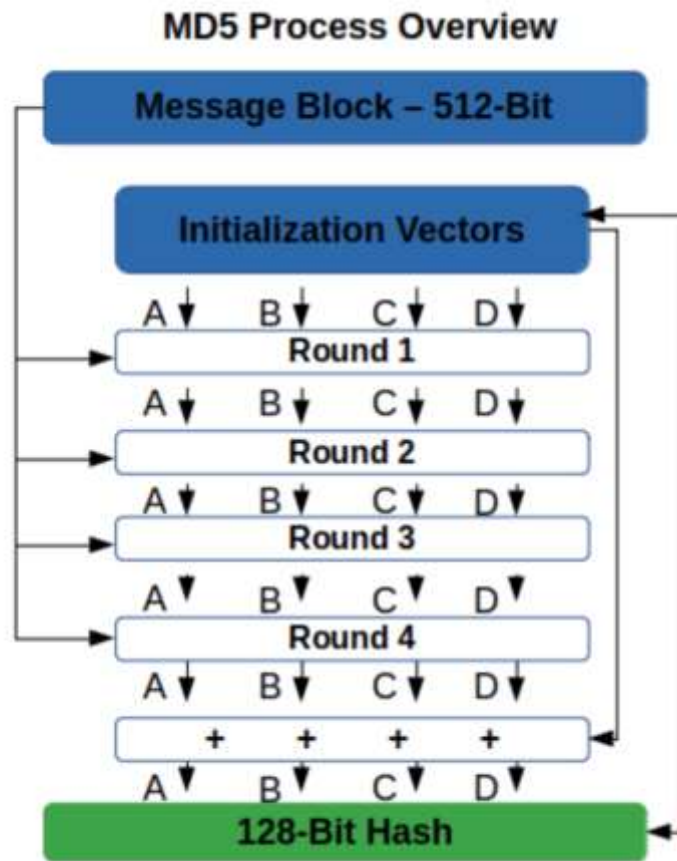
but given message with given hash value, it is easy to find another message with same hash value:

<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
<hr/>			<hr/>	
B2 C1 D2 AC		 <i>different messages</i> <i>but identical checksums!</i>	B2 C1 D2 AC	

Hash Functions

- MD5, Ron Rivest, in use today.
 - 128-bit hash in a four-step process consisting of a padding step (adding “1” with multiple zeroes to “boost” the length of the message).
 - Append step (appending 64-bit representation of the message length before padding)
 - Accumulator
 - Message’s 16-word blocks are processed (mangled) in 4 rounds
- SHA-1, similar to MD4 (predecessor of MD5).
 - SHA-1, US federal standard
 - Produces 160-bit message digest

MD5, SHA-1



Message Authentication Code

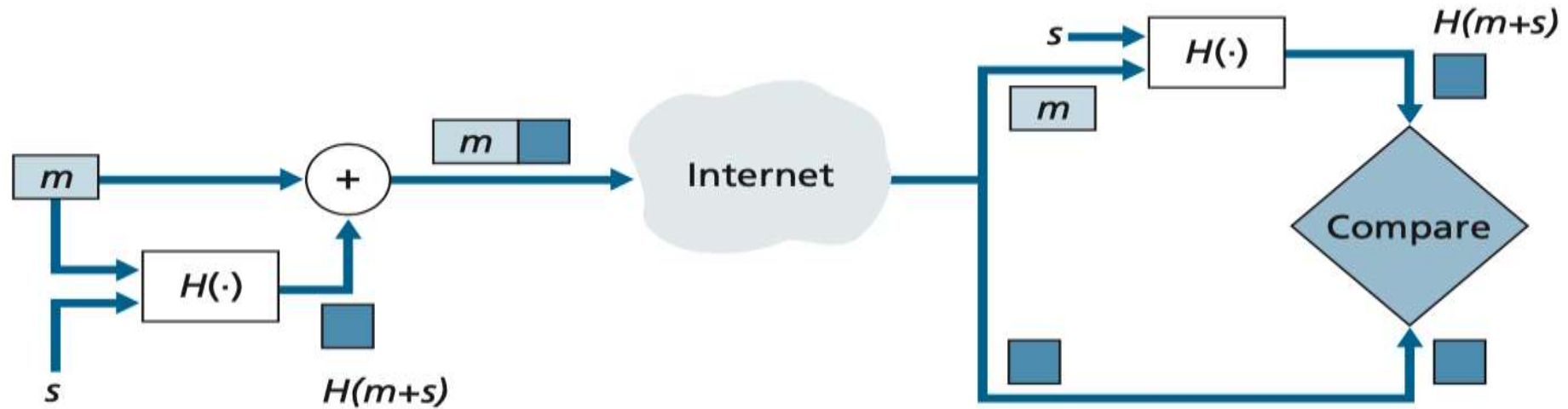
- Alice creates message m and calculates the hash $H(m)$ (for example, with SHA-2).
- Alice then appends $H(m)$ to the message m , creating an extended message
- $(m, H(m))$, and sends the extended message to Bob.
- Bob receives an extended message (m, h) and calculates $H(m)$. If $H(m) = h$,
- Bob concludes that everything is fine.

Problem with that?

Authentication Key

- To perform message integrity, in addition to hash functions, Alice and Bob will need a shared secret, S .
- String of bits – **authentication key**
 - Using that shared secret, message integrity can be performed as follows:
 1. Alice creates message m , concatenates s with m to create $m + s$, and calculates the hash $H(m + s)$ (for example, with SHA-2). $H(m + s)$ is called the **message authentication code (MAC)**.
 2. Alice then appends the MAC to the message m , creating an extended message $(m, H(m + s))$, and sends the extended message to Bob.
 3. Bob receives an extended message (m, h) and knowing s , calculates the MAC $H(m + s)$. If $H(m + s) = h$, Bob concludes that everything is fine.

Authentication Key in Action



Key:

m = Message
 s = Shared secret

How do we distribute the shared authentication key to the communicating entities?

Chapter 8 outline

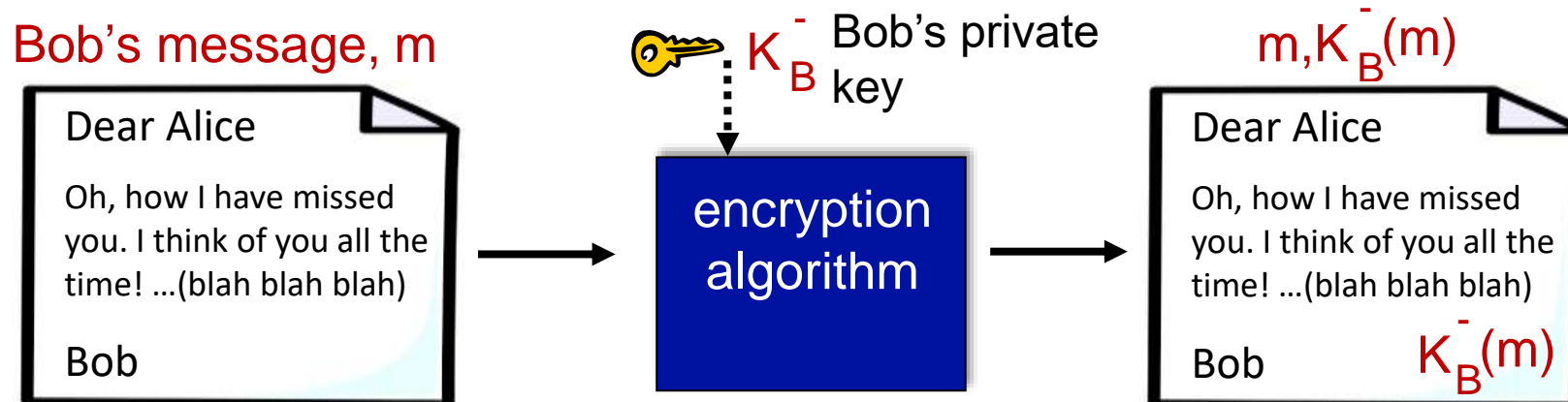
- What is network security?
- Principles of cryptography
- Authentication, **message integrity**
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



Digital signatures

cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document: he is document owner/creator.
- *verifiable, nonforgeable*: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
- **simple digital signature for message m :**
 - Bob signs m by encrypting with his private key K_B , creating “signed” message, $K_B^-(m)$



Digital signatures

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

Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

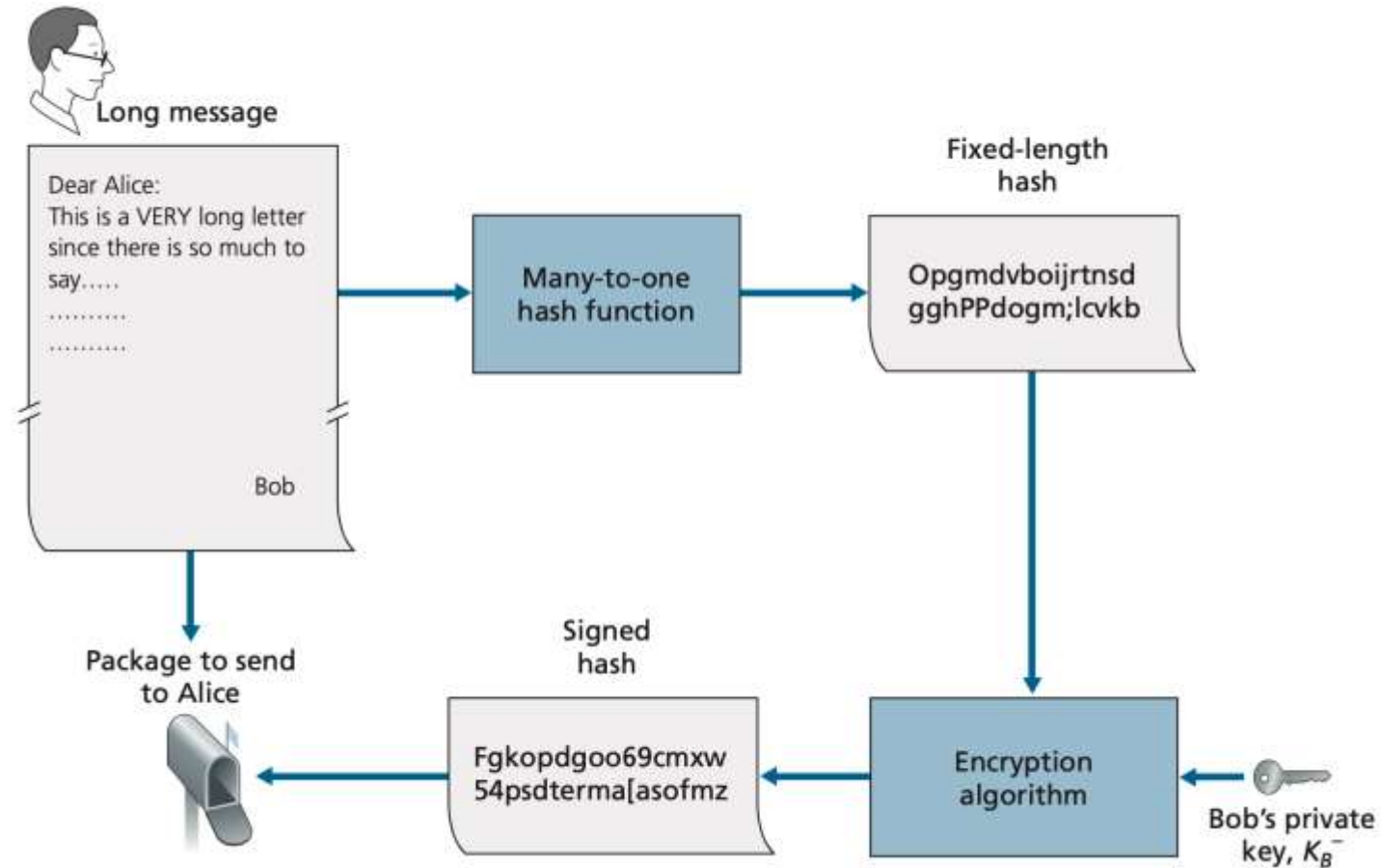
- ✓ Alice can take m , and signature $K_B(m)$ to court and prove that Bob signed m

One concern with signing data by encryption and decryption (mathematical procedures) is cost in terms of computation.

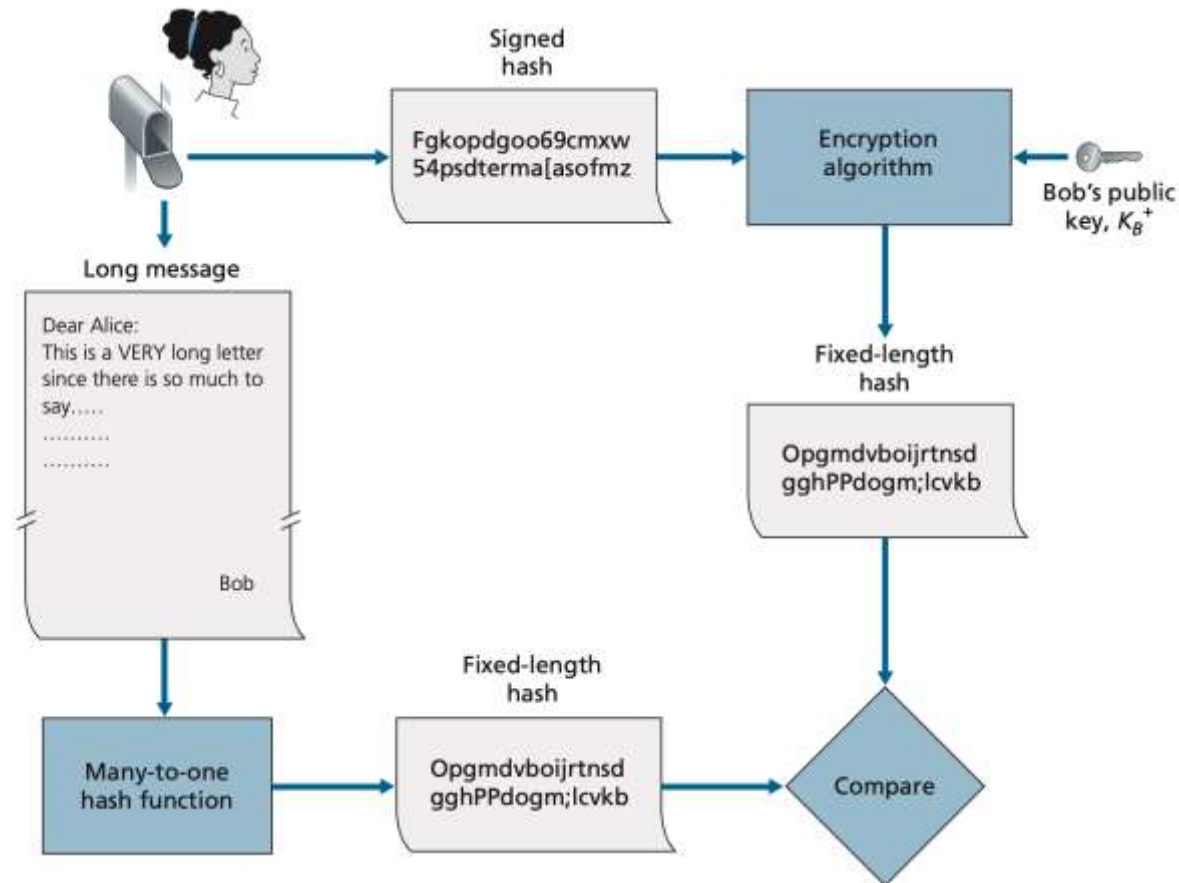
Hash Functions in Digital Signatures

- Signing data via complete encryption/decryption is extensive and inefficient.
- Hash functions over digital signatures
 - Computing fixed-length “fingerprint”
- **Using Hash function, sender signs the hash of the message in opposed to message itself**
 - Hash output is much smaller in nature than the original message, hence, much easier to compute

Hash in Digital Signatures in Action - Sender



Hash in Digital Signatures in Action - Receiver



Hash in Digital Signatures – Step by Step

Sender:

1. Sender puts his original long message through a hash function
2. Sender then digitally signs the resulting hash with the private key.
3. The original message (in cleartext) along with the digitally signed message digest (henceforth referred to as the digital signature) is then sent to Alice.

Receiver:

1. Receiver applies the sender's public key to the message to obtain a hash result.
2. Receiver also applies the hash function to the cleartext message to obtain a second hash result.
3. If the two hashes match, then Alice can be sure about the integrity and author of the message

Digital Signatures vs Message Authentication Code(MAC)

- Both use document or a message for input
- To create a MAC out of the message, we append an authentication key to the message, and then take the hash of the result
- **Note that neither public key nor symmetric key encryption is involved in creating the MAC**
 - Used in OSPF
 - TLS and IPSec
- To create a digital signature, we first take the hash of the message and then encrypt the message with our private key
- Thus, a digital signature is a “heavier” technique, since it requires an underlying Public Key Infrastructure (PKI) with certification authorities (discussed next)

Public Key Certification

- Public key certification – certification that public key belongs to a specific entity
 - Used in IPSec and TLS

Need for certified public keys

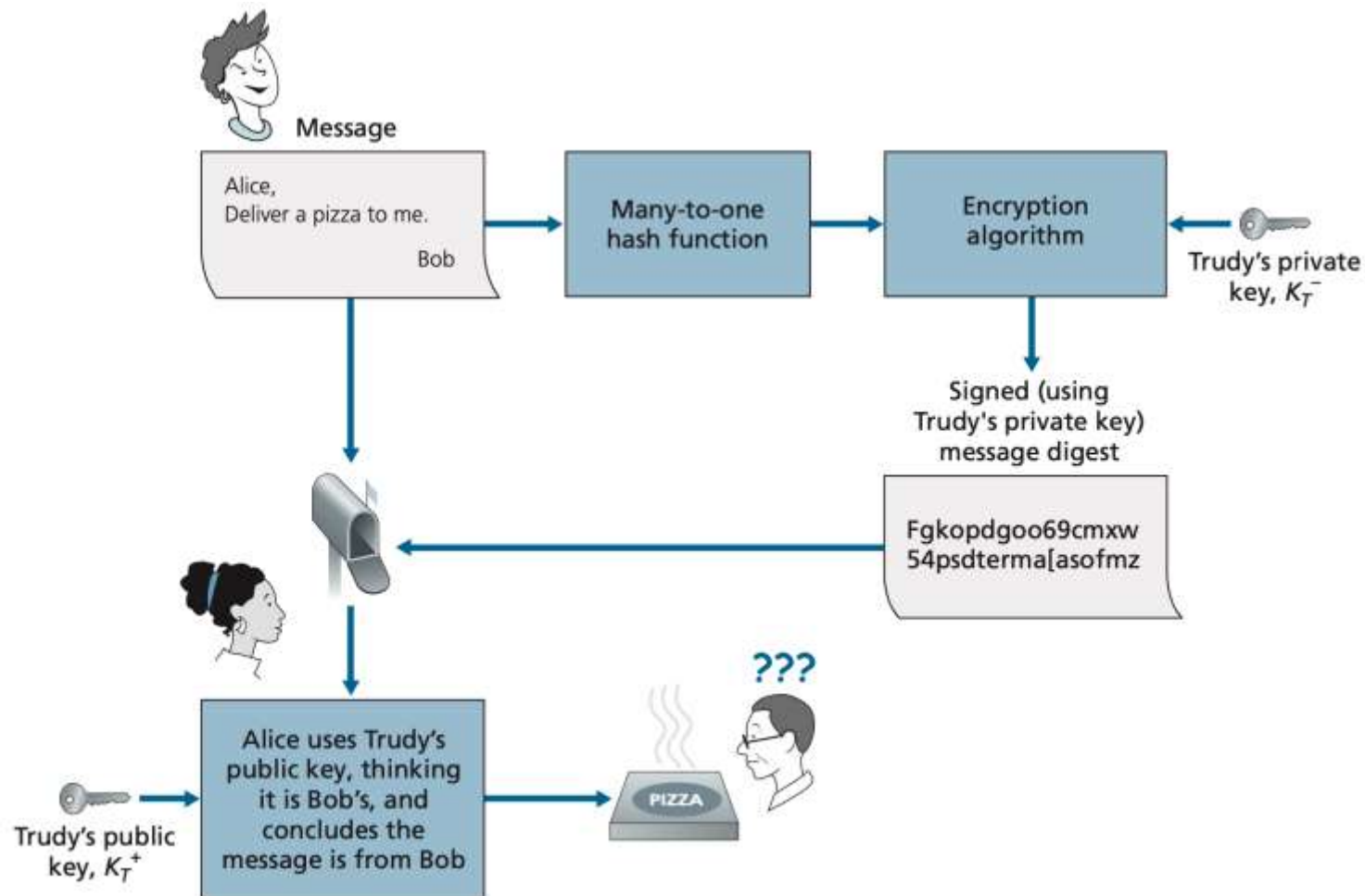
- Alice is in pizza delivery business and accepts pizza delivery over the Internet
- Bob sends plaintext message with his home address, pizza type he wants
- Bob also includes a digital signature to prove to Alice that he is the true source of the message
- To verify the signature, Alice obtains Bob's public key (perhaps from a public key server or from the e-mail message) and checks the digital signature



Need for certified public keys

- Trudy comes along; does the prank
- Trudy sends an email to Alice, in which she says she is Bob, and provides all the information about Bob, **but includes her own public key**
- Alice assumes it is Bob's public key
- Trudy attaches digital signature which was created with her own private key
- Alice applies Trudy's public key (thinking it is Bob's) to digital signature and concludes that Bob was indeed the author

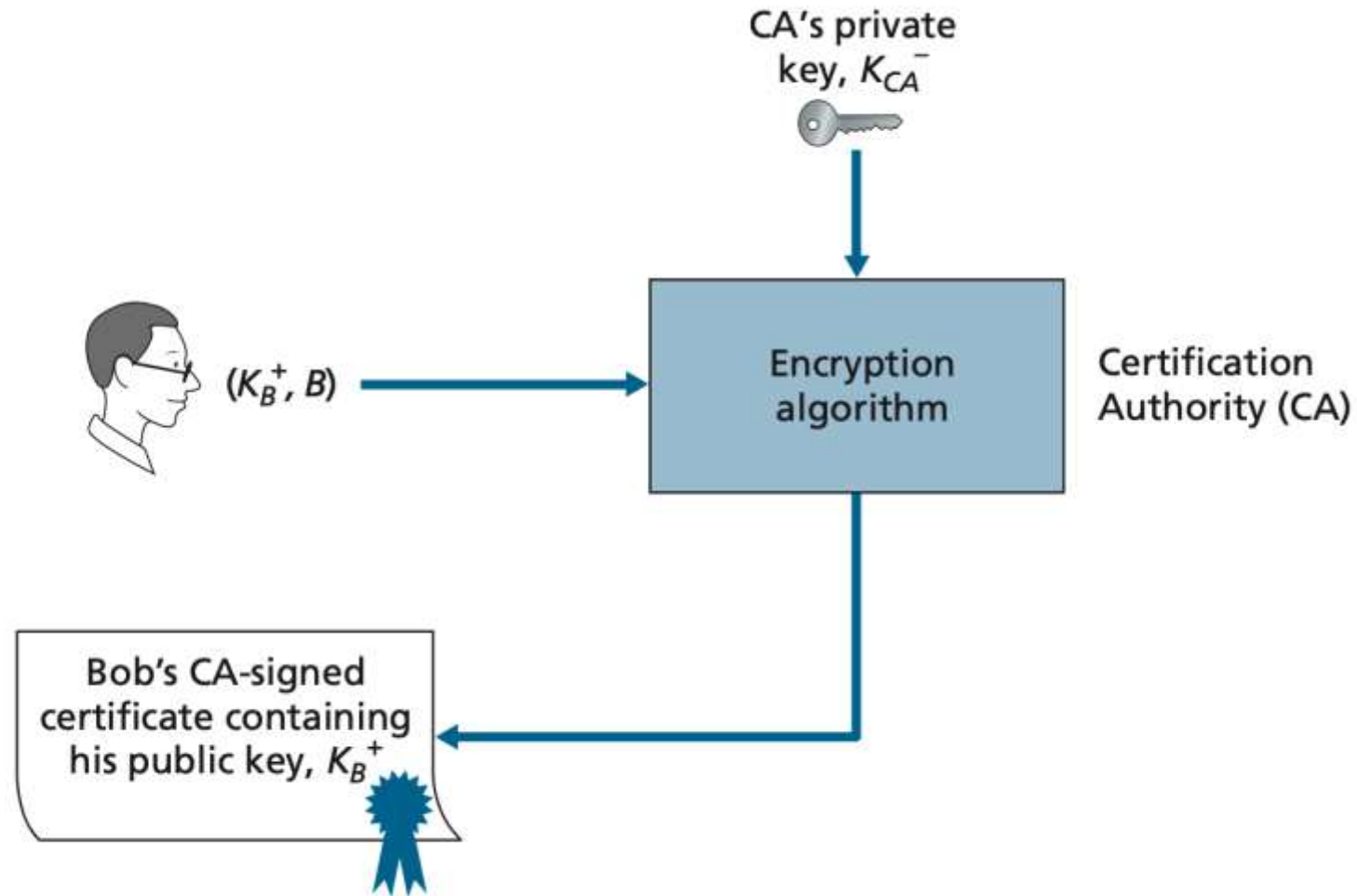




Public key Certification Authorities (CA)

- **certification authority (CA):** binds public key to particular entity
- Validates identities and issues certificates
- A CA verifies that an entity (a person, a router, and so on) is who it says it is. There are no mandated procedures for how certification is done. When dealing with a CA, one must trust the CA to have performed a suitably rigorous identity verification.
- Once the CA verifies the identity of the entity, the CA creates a certificate that binds the public key of the entity to the identity. The certificate contains the public key and globally unique identifying information about the owner of the public key (for example, a human name or an IP address). The certificate is digitally signed by the CA

Public key Certification Authorities (CA)



Standards for CA

- ITU X.509 [ITU 2005a] specifies an authentication service as well as a specific syntax for certificates
- RFC 1422] describes CA- based key management for use with secure Internet e-mail. It is compatible with X.509 but goes beyond X.509 by establishing procedures and conventions for a key management architecture

Field Name	Description
Version	Version number of X.509 specification
Serial number	CA-issued unique identifier for a certificate
Signature	Specifies the algorithm used by CA to sign this certificate
Issuer name	Identity of CA issuing this certificate, in distinguished name (DN) [RFC 4514] format
Validity period	Start and end of period of validity for certificate
Subject name	Identity of entity whose public key is associated with this certificate, in DN format
Subject public key	The subject's public key as well indication of the public key algorithm (and algorithm parameters) to be used with this key

Chapter 8 outline

- What is network security?
- Principles of cryptography
- **Authentication**, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



Authentication

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

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

Non-Human authentication – routers; OSPF →

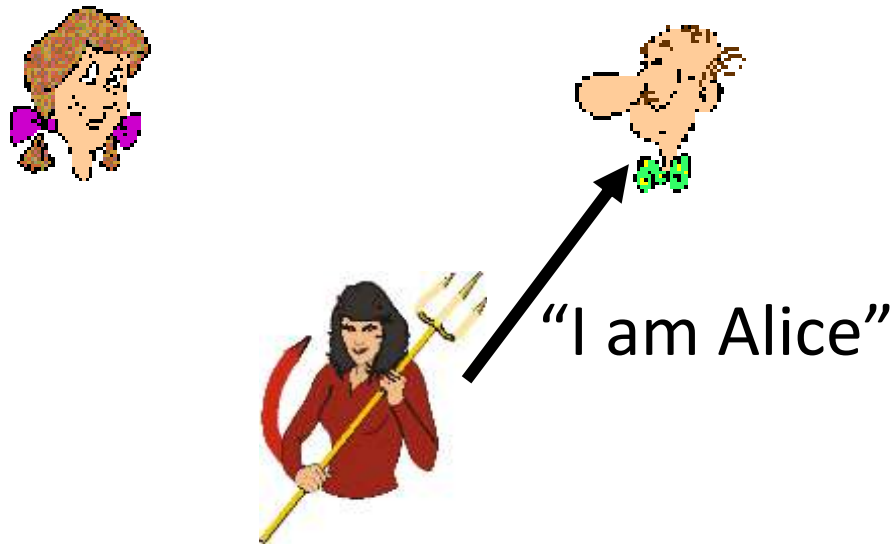
Authentication Protocol



Authentication

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

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



*in a network, Bob
can not “see”
Alice, so Trudy
simply declares
herself to be Alice*



Authentication: another try

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

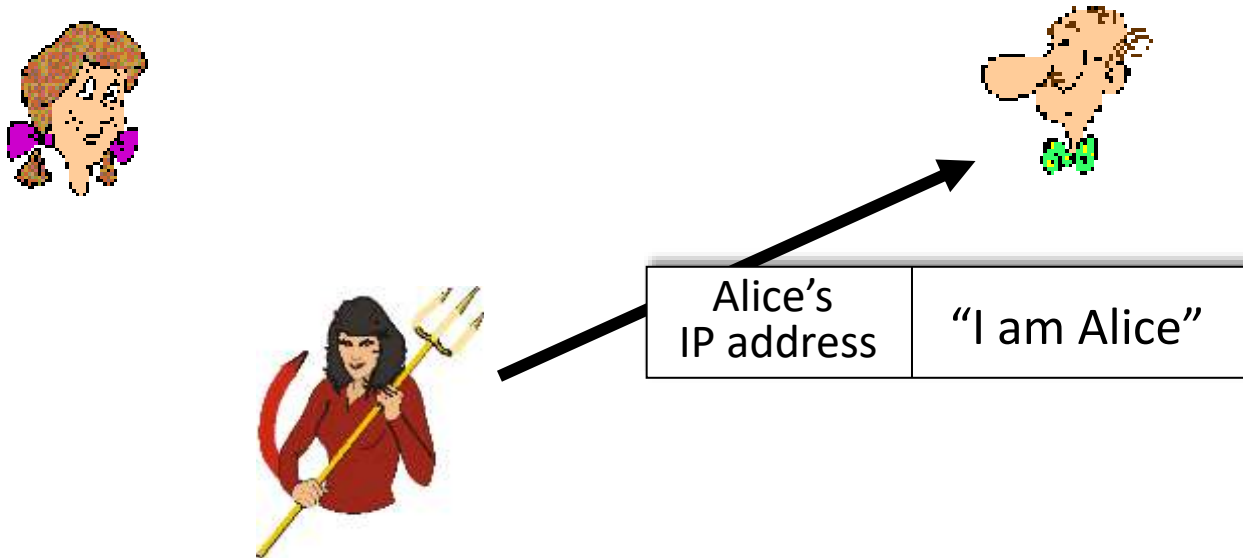
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Authentication: another try

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

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

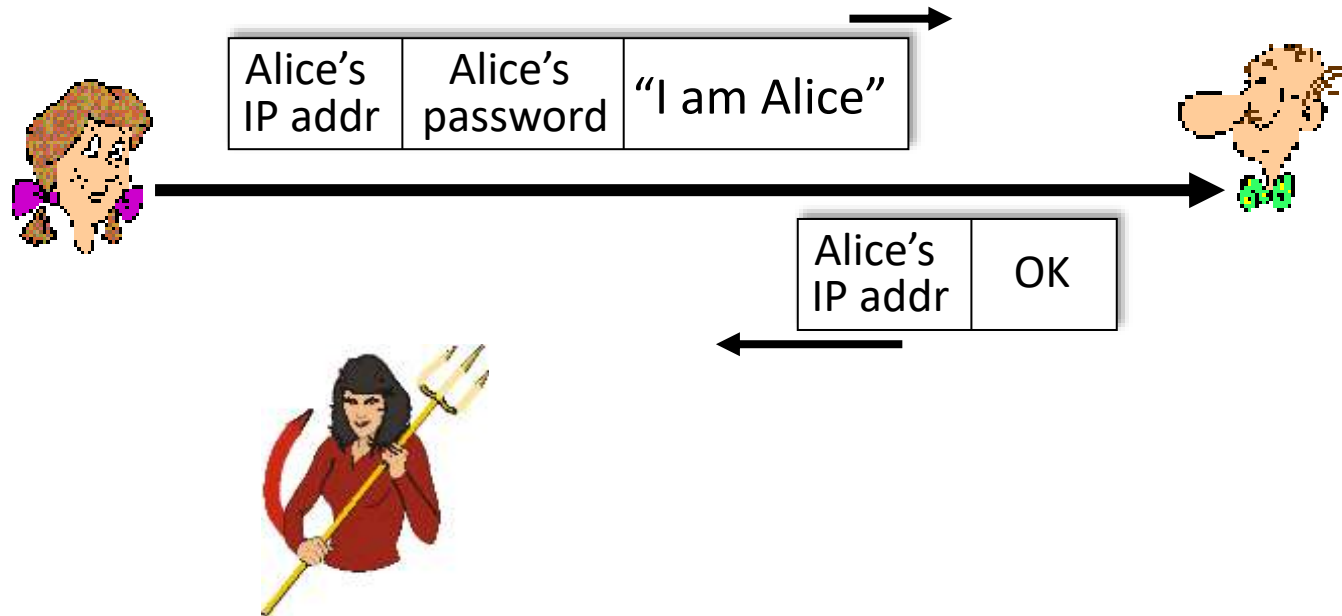


*Trudy can create
a packet “spoofing”
Alice’s address*

Authentication: a third try

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

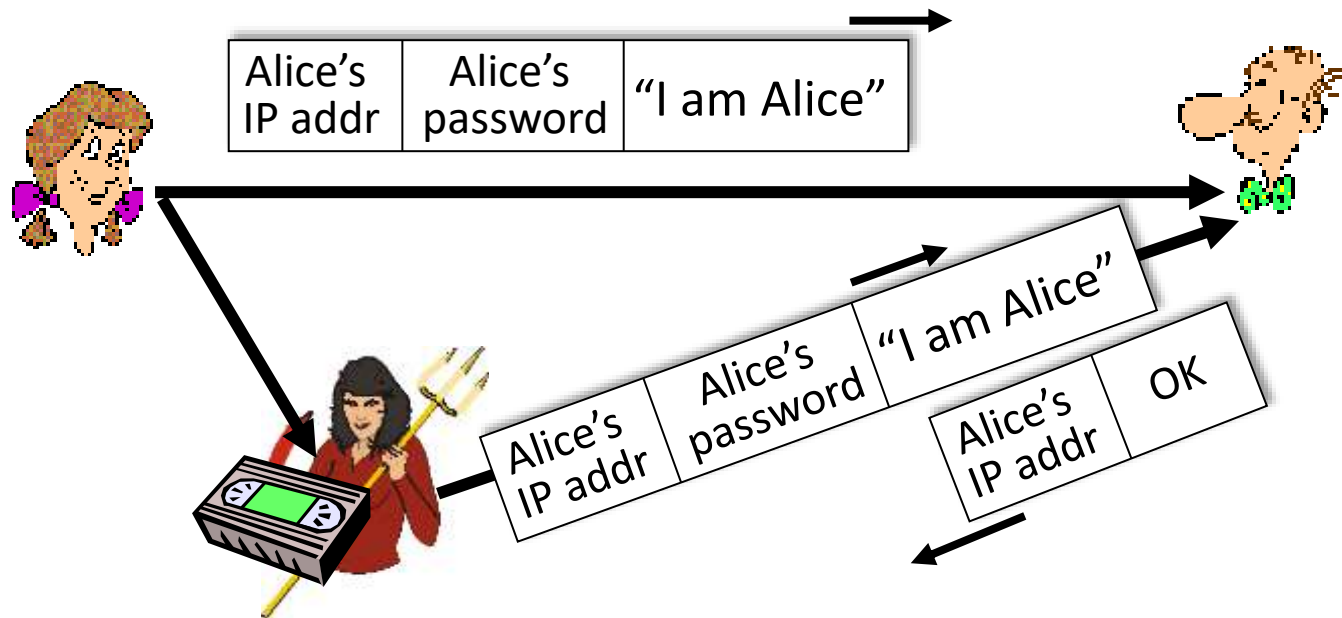
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



Authentication: a third try

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

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

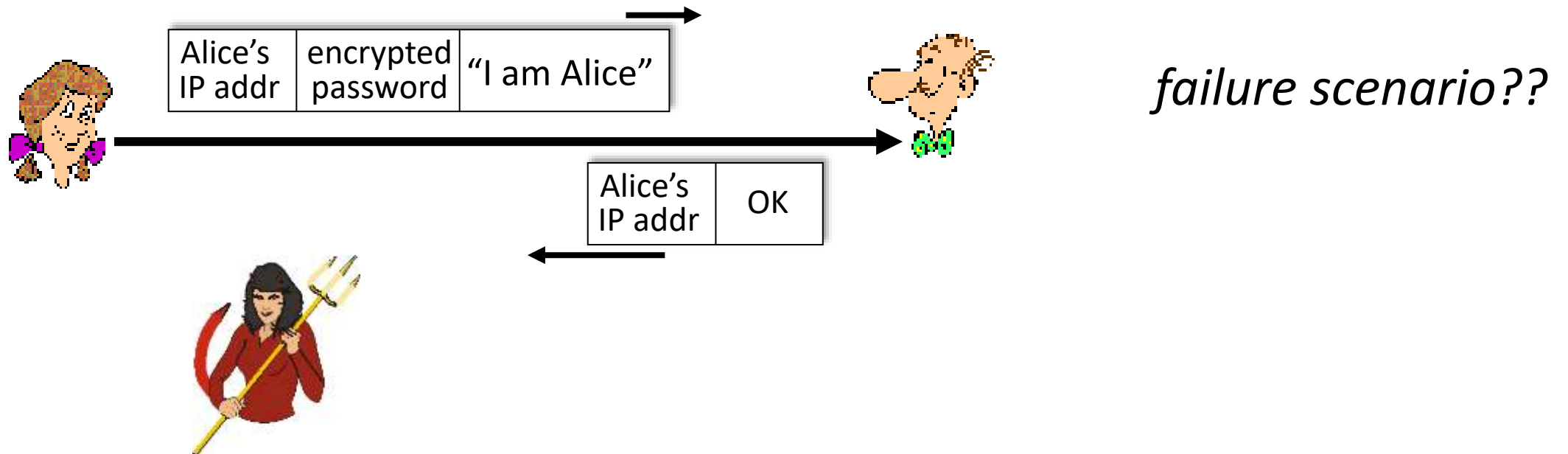


*playback attack:
Trudy records
Alice's packet
and later
plays it back to Bob*

Authentication: a modified third try

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

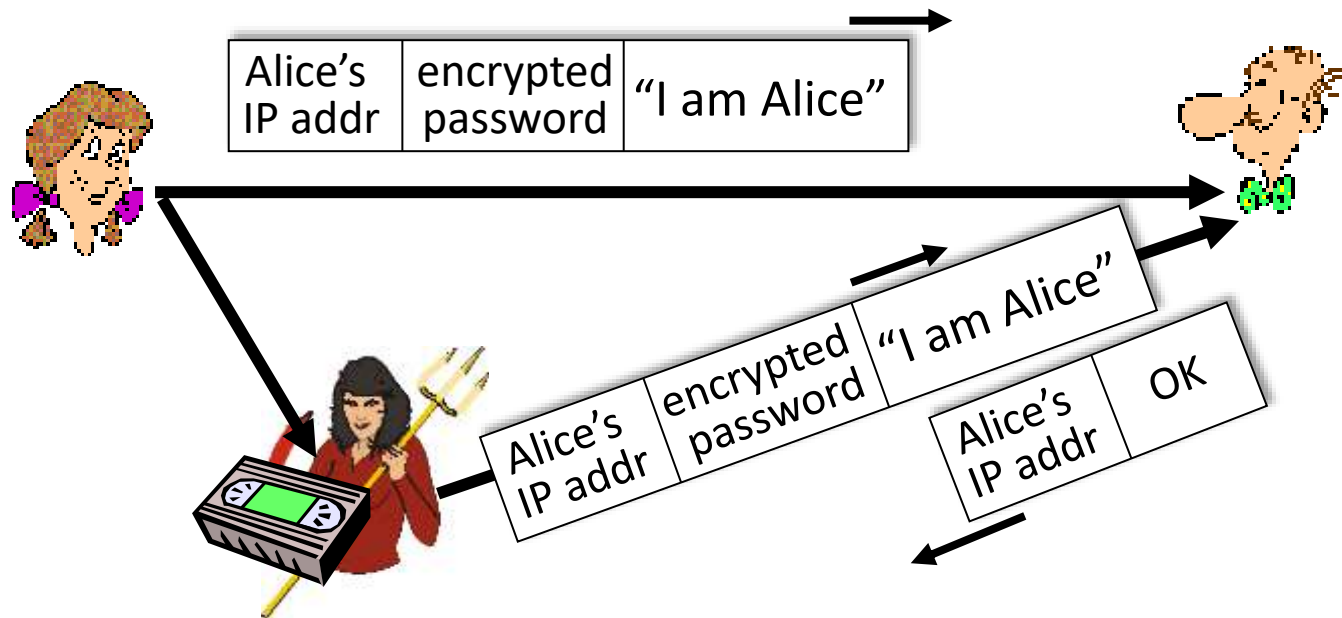
Protocol ap3.0: Alice says “I am Alice” and sends her encrypted secret password to “prove” it.



Authentication: a modified third try

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

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



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

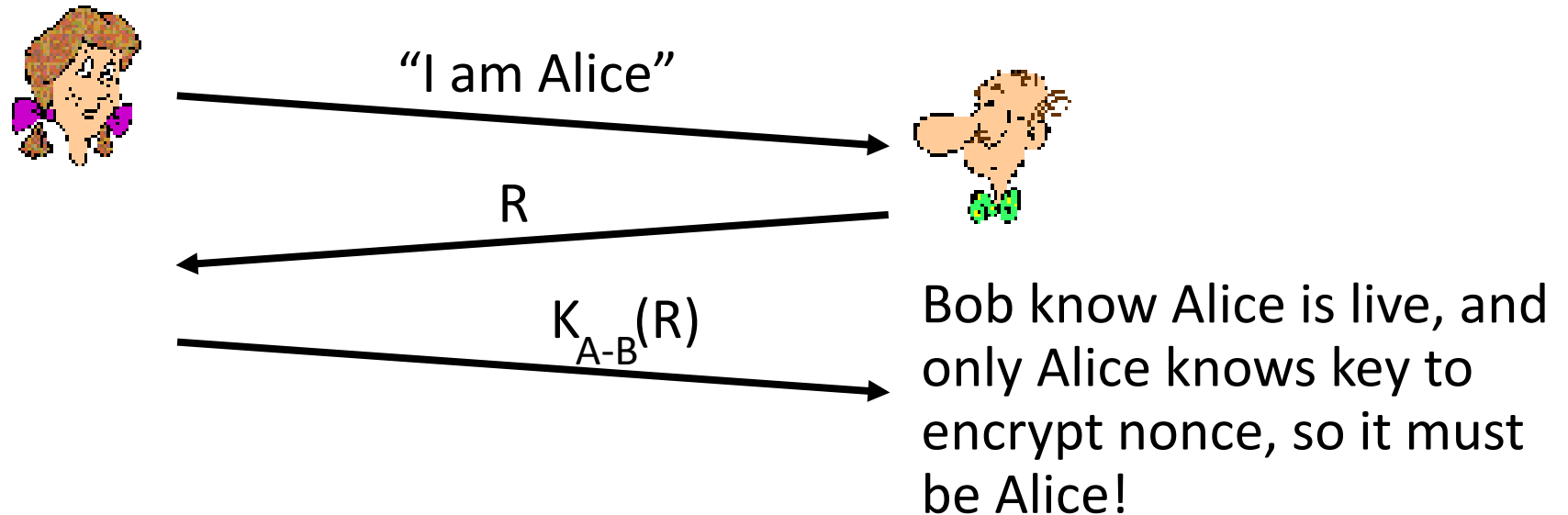
Authentication: a fourth try

Goal: avoid playback attack

nonce: number (R) used only **once-in-a-lifetime**; analogous to seq number in TCP

protocol ap4.0: to prove Alice “live”, Bob sends Alice nonce, R

- Alice must return R, encrypted with shared secret key



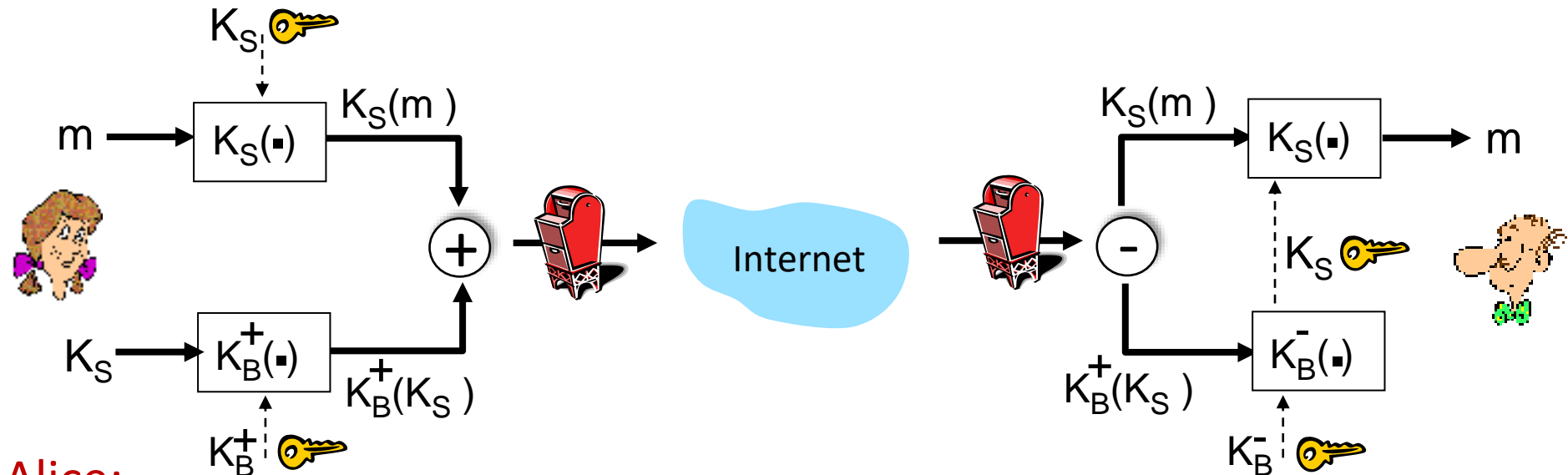
Chapter 8 outline

- What is network security?
- Principles of cryptography
- Authentication, message integrity
- **Securing e-mail**
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



Secure e-mail: confidentiality

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

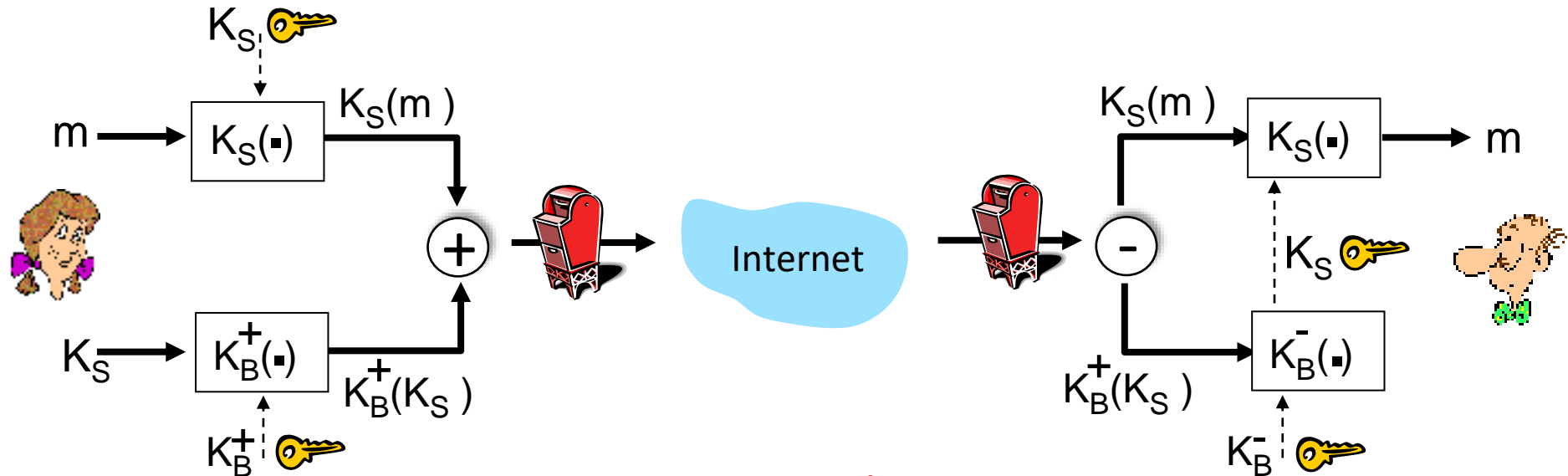


Alice:

- generates random *symmetric* key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- concatenates the encrypted message and the encrypted symmetric key to form a "package,"
- sends both $K_S(m)$ and $K_B^+(K_S)$ to Bob

Secure e-mail: confidentiality (more)

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

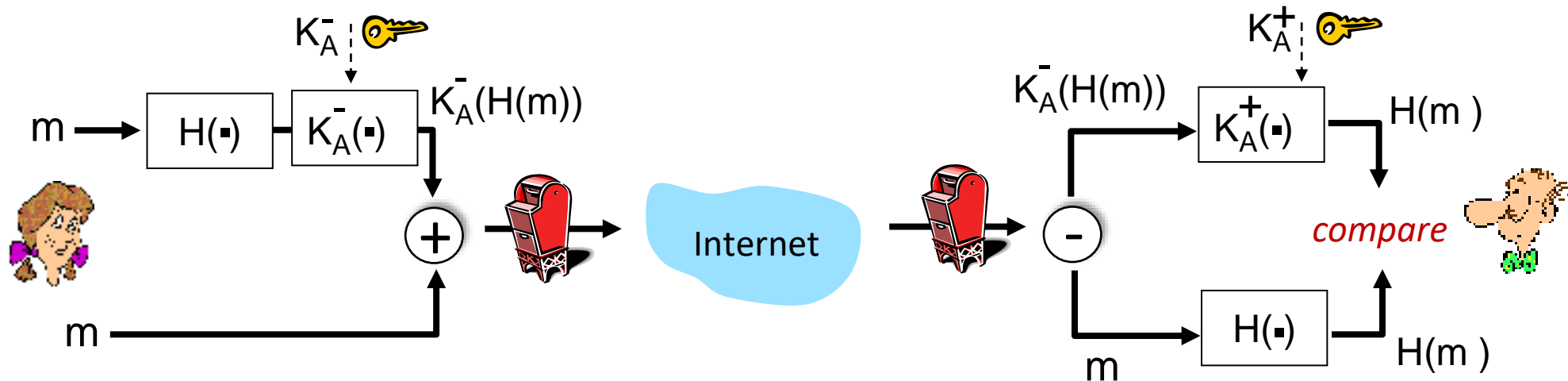


Bob:

- uses his private key to to obtain symmetric key, K_S
- uses K_S to decrypt $K_S(m)$ to decrypt the message m

Secure e-mail: integrity, authentication

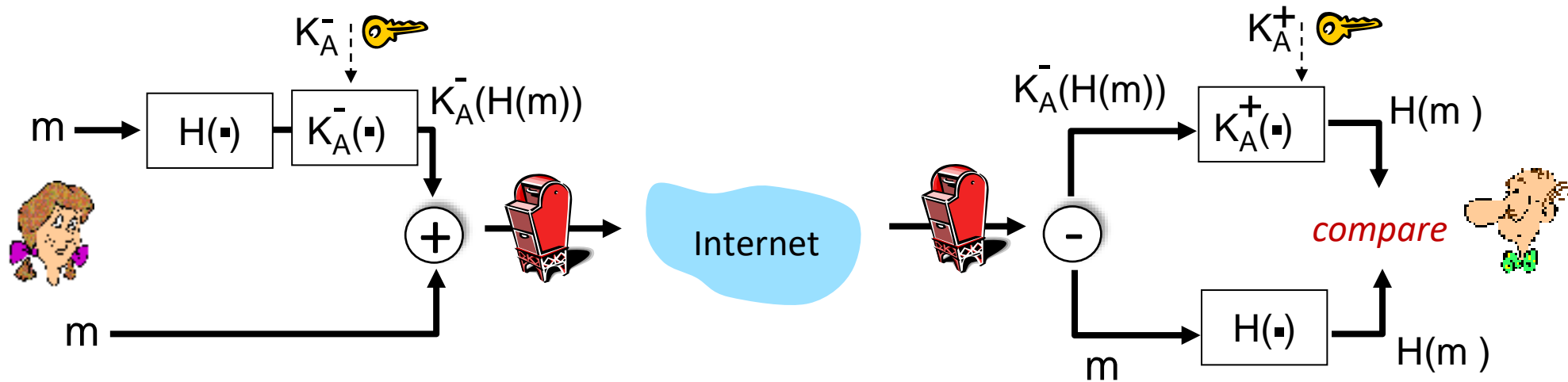
Alice wants to send m to Bob, with *message integrity, authentication*
Digital Signatures & Message Digests:



- Alice applies hash function H , to message M to obtain message digest
- Alice signs the result of the hash function with her private key, to create digital signature
- Concatenates the original (unencrypted) message with the signature to create package
- Then sends the package to bob's email address.

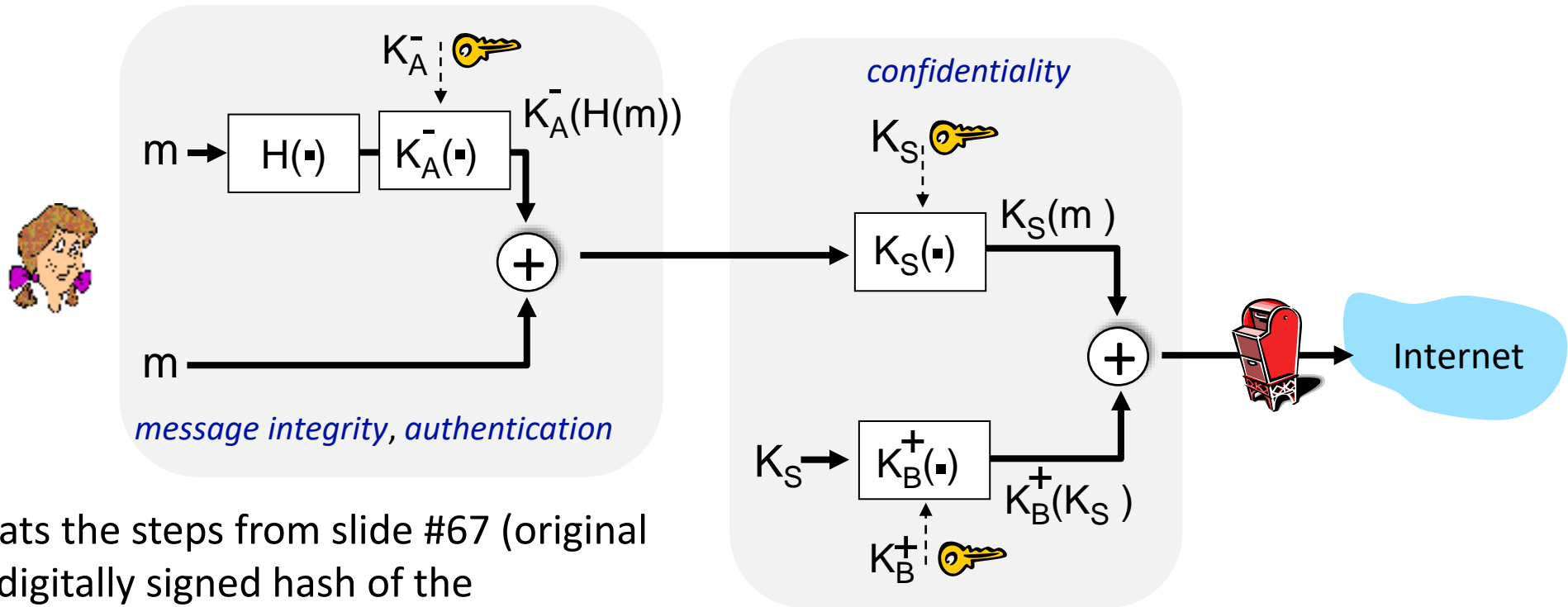
Secure e-mail: integrity, authentication

Alice wants to send m to Bob, with *message integrity, authentication*
Digital Signatures & Message Digests:



- Bob applies Alice's public key to the signed message digest
- Compares the result of this operation with his own hash, H , of the message.
- If results are the same \rightarrow message came from Alice and is unaltered.

Secure e-mail: confidentiality integrity, authentication



- Alice repeats the steps from slide #67 (original message, digitally signed hash of the message)
- Then she treats that package as a message in itself and sends through the steps in Slide #66 (using the combination of public key and symmetric key)

PGP

- Written by Phil Zimmermann in 1991, **Pretty Good Privacy** (PGP) is an example of an e-mail encryption scheme.
- The design is very similar to what we just covered.
- When PGP is installed the software creates a public key pair for the user
 - The public key can be posted on the user's Web site or placed in a public key. Server
 - The private key is protected by the use of a password.
 - Password is entered every time a user is accessing the private key.
 - PGP gives the user the option of digitally signing and encrypting the message

Chapter 8 outline

- What is network security?
- Principles of cryptography
- Authentication, message integrity
- Securing e-mail
- **Securing TCP connections: TLS**
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



Transport-Layer security (TLS)

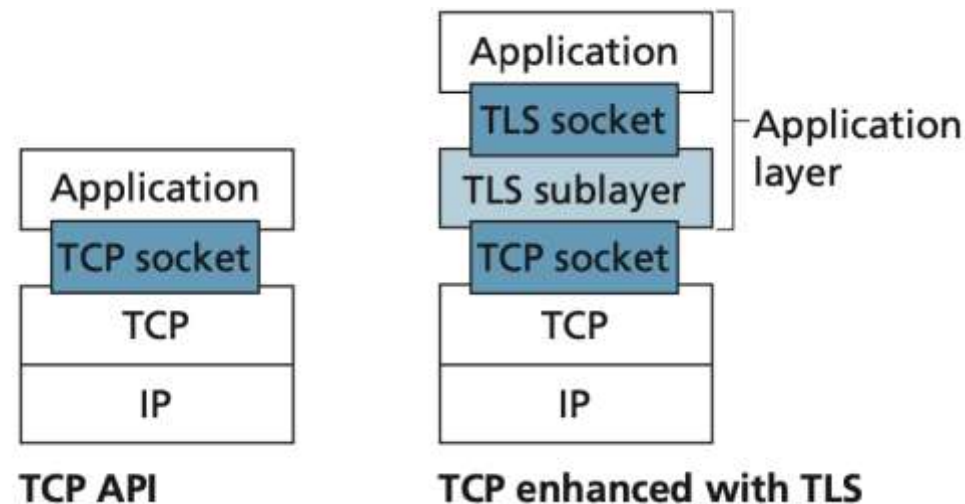
- widely deployed security protocol above the transport layer
 - supported by almost all browsers, web servers: https (port 443)
 - Earlier version is SSL (secure socket layer)
 - provides:
 - **confidentiality**: via *symmetric encryption*
 - **integrity**: via *cryptographic hashing*
 - **authentication**: via *public key cryptography*
- } *all techniques we have studied!*
- Used by Gmail and Internet e-commerce sites, and many others.
 - history:
 - early research, implementation: secure network programming, secure sockets
 - secure socket layer (SSL) deprecated [2015]
 - TLS 1.3: RFC 8846 [2018]

TLS Overview

- Someone is purchasing an item from the web site (store)
- Enters all the address and credit card information, shipping address, etc.
- Customer expects to receive an item ordered, and correct amount of money charged for the item in the credit card statement
- If no Security measures taken:
 - If no confidentiality (encryption) is used, an intruder could intercept customer's order and obtain the payment card information. The intruder could then make purchases at customer's expense.
 - If no data integrity is used, an intruder could modify customer's order, having them purchase ten times more, or anything else
 - if no server authentication is used, a server could display "Alice Incorporated's" famous logo when in actuality the site maintained by Trudy, who is masquerading as Alice Incorporated. After receiving Bob's order, Trudy could take Bob's money and run.

TLS Overview

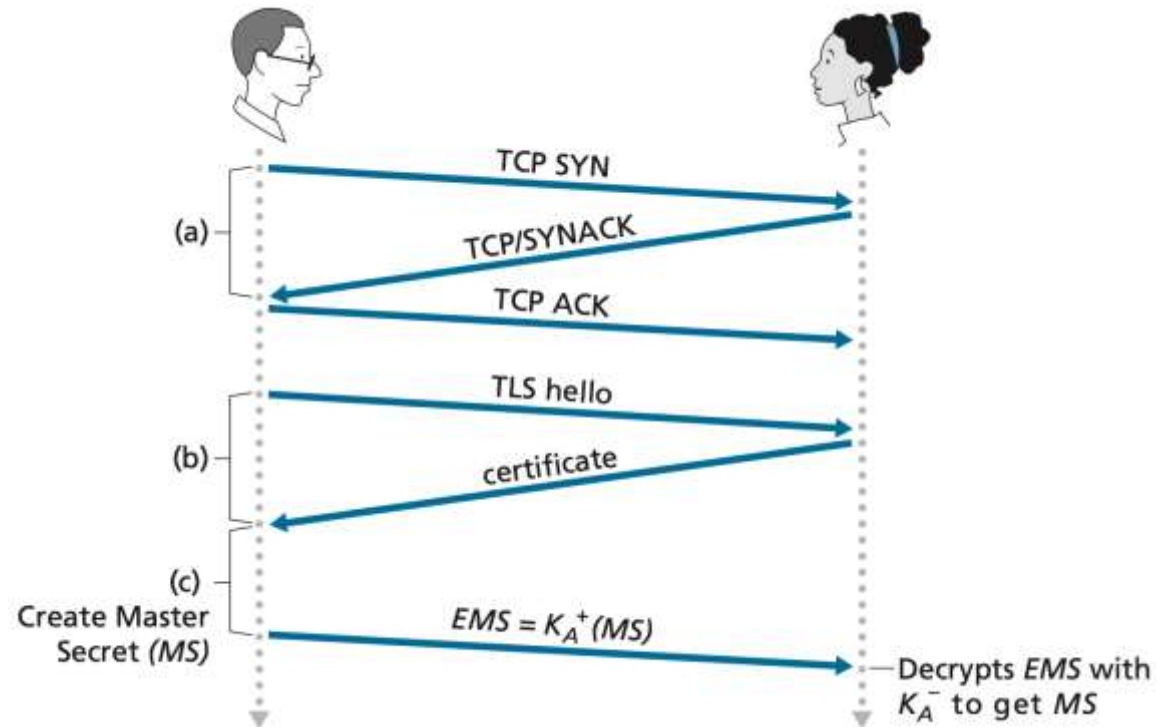
- TLS addressing those issues by enhancing TCP with **Confidentiality, data Integrity, Server Authentication and Client Authentication**
- **TLS is often used to provide security to transactions that take place over HTTP**
- **Because TLS secures TCP, it can be employed by an application that runs over TCP.**
- TLS provide API with sockets which is analogous to TCP's API
- When an application wants to employ TLS. The application includes TLS classes/libraries.
- **TLS technically resides in the application layer, From the developer's perspective it is a transport protocol that provides TCP's services enhanced with security services.**



Transport-layer security: what's needed?

- let's *build* almost- TLS protocol, *a-tls*, to see what's needed!
- we've seen the “pieces” already previously:
 - **handshake**: Alice, Bob use their certificates, private keys to authenticate each other, exchange or create shared secret
 - **key derivation**: Alice, Bob use shared secret to derive set of keys
 - **data transfer**: stream data transfer: data as a series of records
 - not just one-time transactions
 - **connection closure**: special messages to securely close connection

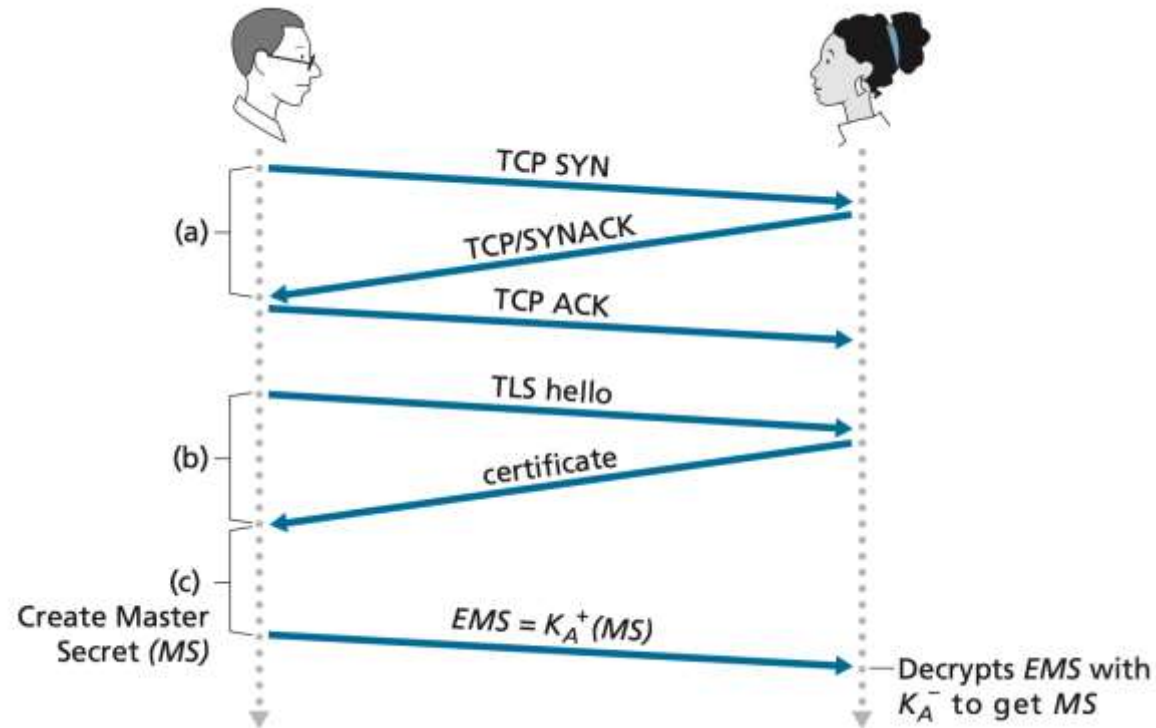
a-tls: initial handshake



Steps required:

- (a) establish a TCP connection with Alice,
- (b) verify that Alice is really Alice
- (c) send Alice a master secret key, which will be used by both Alice and Bob to generate all the symmetric keys they need for the TLS session

a-tls: initial handshake



- Because the certificate has been certified by a CA (certificate Authority), Bob knows for sure that the public key in the certificate belongs to Alice.
- Bob then generates a Master Secret (MS) (which will only be used for this TLS session), encrypts the MS with Alice's public key to create the Encrypted Master Secret (EMS), and sends the EMS to Alice.
- Alice decrypts the EMS with her private key to get the MS. After this phase, both Bob and Alice (and no one else) know the master secret for this TLS session

A-tls: key derivation

- In principle, the MS (Master Secret), now shared by Bob and Alice, could be used as the symmetric session key for all subsequent encryption and data integrity checking.
 - Different cryptographic keys suggested, and also to use different keys for encryption and integrity checking. **Thus, both Alice and Bob use the MS to generate four keys:**
- EB = session encryption key for data sent from Bob to Alice
- MB = session HMAC key for data sent from Bob to Alice, where HMAC [RFC 2104] is a standardized hashed message authentication code (MAC)
- EA = session encryption key for data sent from Alice to Bob
- MA = session HMAC key for data sent from Alice to Bob
- At the end of the key derivation phase, both Alice and Bob have all four keys. **The two encryption keys will be used to encrypt data; the two HMAC keys will be used to verify the integrity of the data.**

A-tls: data transfer

- Keys are generated, ready for data transfer
- Since TCP is a byte-stream protocol, a natural approach would be for TLS to encrypt application data on the fly and then pass the encrypted data on the fly to TCP
 - Integrity check placeholder?
- To address this issue, TLS breaks the data stream into records, appends an HMAC to each record for integrity checking, and then encrypts the record + HMAC.

t-tls: encrypting data (more)

- possible attacks on data stream?
 - *re-ordering*: man-in middle intercepts TCP segments and reorders (manipulating sequence #s in unencrypted TCP header)
 - *replay*
- solutions:
 - use TLS sequence numbers (data, TLS-seq-# incorporated into MAC - message authentication code)
 - use nonce

TLS Record



EB = session encryption key for data sent from Bob to Alice

TLS Actual Step by Step

- <https://www.youtube.com/watch?v=j9QmMEWmcfo>

TLS Actual Step by Step

- The client sends a list of cryptographic algorithms it supports, along with a client nonce.
- From the list, the server chooses a symmetric algorithm (for example, AES) and a public key algorithm (for example, RSA with a specific key length), and HMAC algorithm (MD5 or SHA-1) along with the HMAC keys. It sends back to the client its choices, as well as a certificate and a server nonce.
- The client verifies the certificate, extracts the server's public key, generates a Pre-Master Secret (PMS), encrypts the PMS with the server's public key, and sends the encrypted PMS to the server.
- Using the same key derivation function (as specified by the TLS standard), the client and server independently compute the Master Secret (MS) from the PMS and nonces. The MS is then sliced up to generate the two encryption and two HMAC keys. Furthermore, when the chosen symmetric cipher employs CBC (such as 3DES or AES), then two Initialization Vectors (IVs)—one for each side of the connection—are also obtained from the MS. Henceforth, all messages sent between client and server are encrypted and authenticated (with the HMAC).
- The client sends the HMAC of all the handshake messages.
- The server sends the HMAC of all the handshake messages.

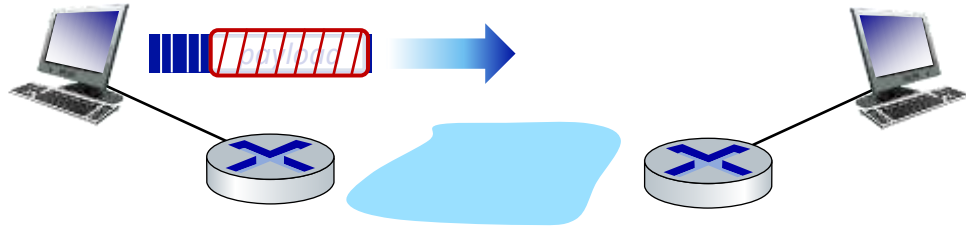
Chapter 8 outline

- What is network security?
- Principles of cryptography
- Authentication, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- **Network layer security: IPsec**
- Security in wireless and mobile networks
- Operational security: firewalls and IDS



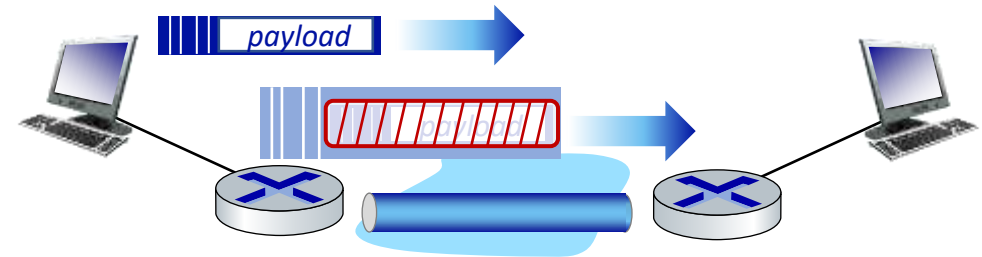
IP Sec

- provides datagram-level encryption, authentication, integrity
 - for both user traffic and control traffic (e.g., BGP, DNS messages)
- two “modes”:



transport mode:

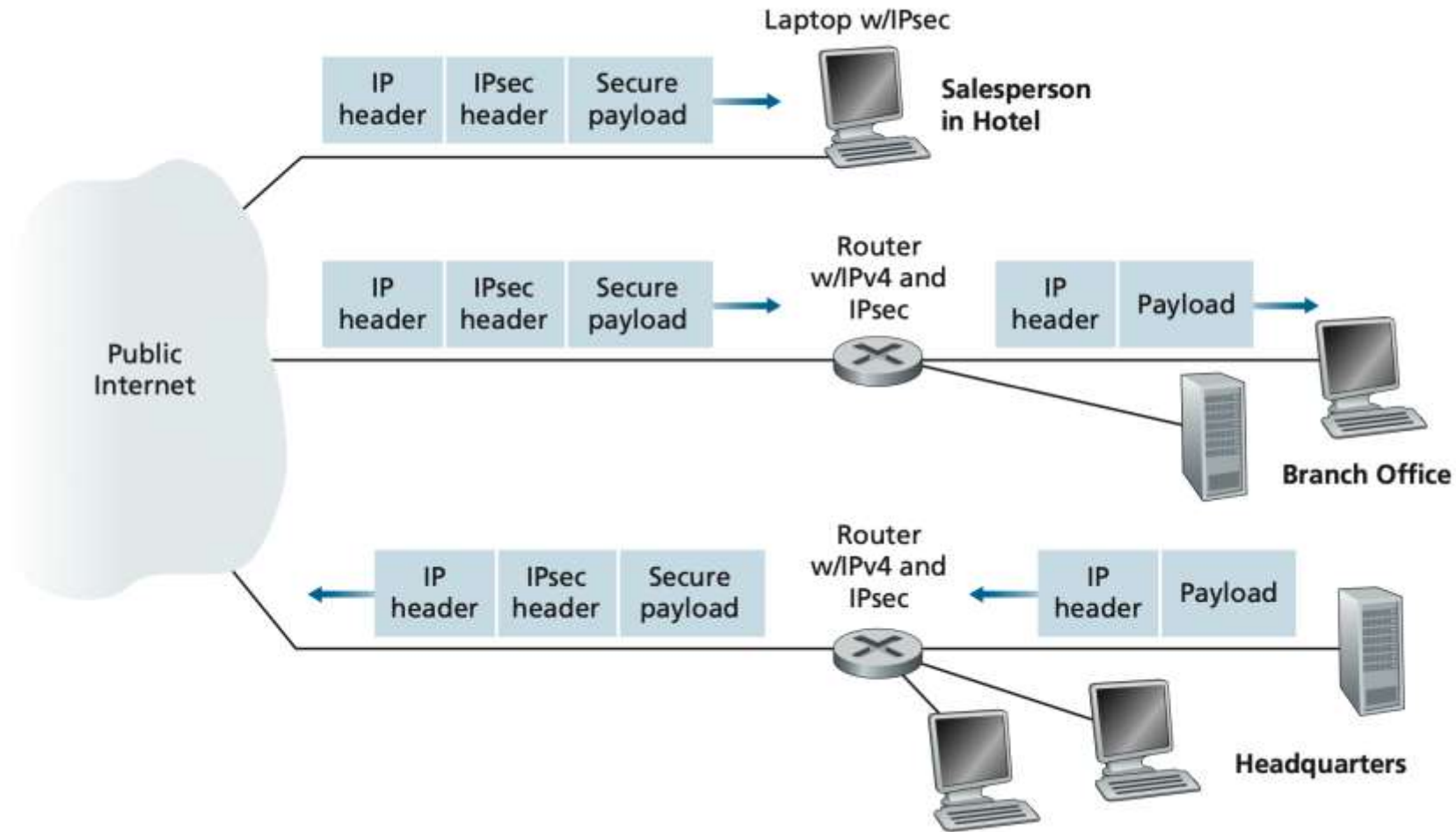
- *only* datagram *payload* is encrypted, authenticated



tunnel mode:

- entire datagram is encrypted, authenticated
- encrypted datagram encapsulated in new datagram with new IP header, tunneled to destination

VPN

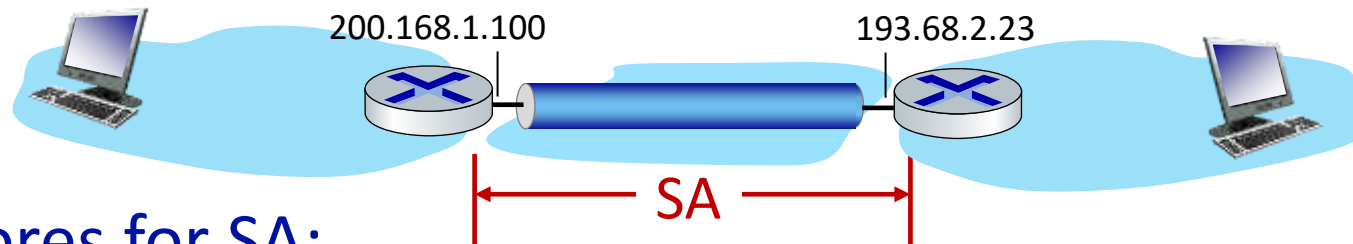


Two IPsec protocols

- Authentication Header (AH) protocol [RFC 4302]
 - provides source authentication & data integrity but ***not confidentiality***
- Encapsulation Security Protocol (ESP) [RFC 4303]
 - **provides source authentication, data integrity, *and confidentiality***
 - more widely used than AH
 - Because confidentiality is often critical for VPNs and other IPsec applications, the ESP protocol is much more widely used than the AH protocol

Security associations (SAs)

- before sending data, **security association (SA)** established from sending to receiving entity (directional)
- ending, receiving entities maintain *state information* about SA
 - recall: TCP endpoints also maintain state info
 - IP is connectionless; **IPsec is connection-oriented!**

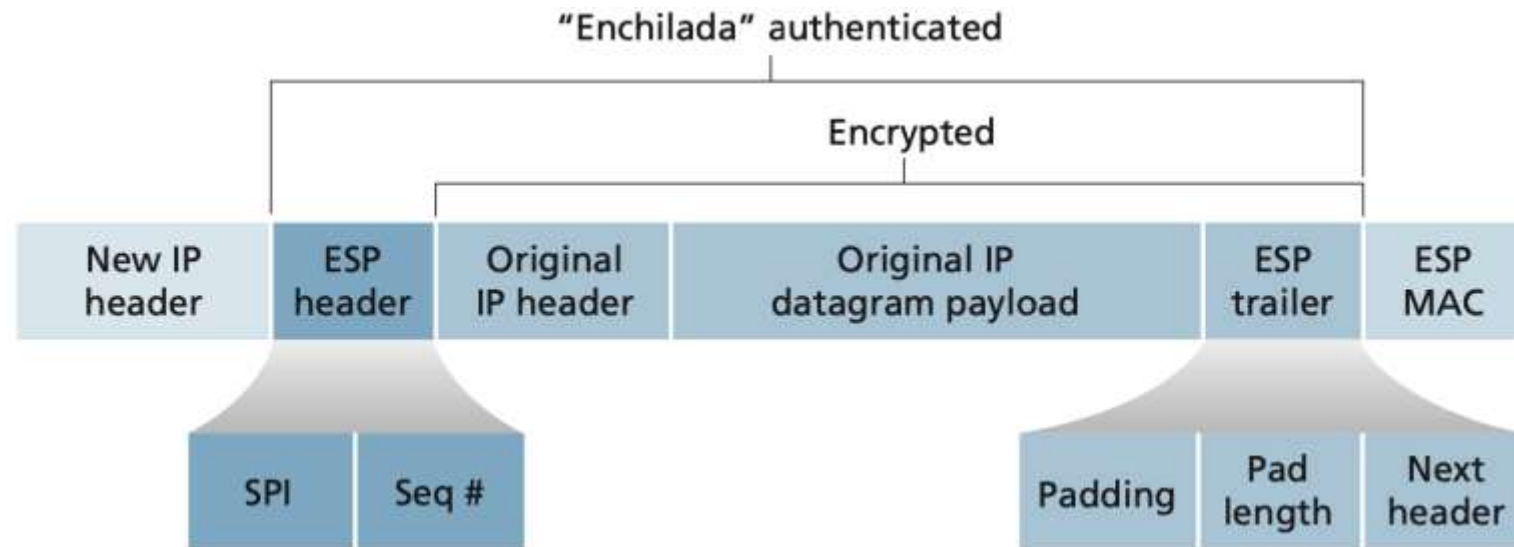


R1 stores for SA:

- 32-bit identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used
- encryption key
- type of integrity check used
- authentication key

IPSec

- IPSec has two different packet forms: tunnel mode and transport mode
 - Tunnel mode being more appropriate for VPNs, more widely deployed than the transport mode.



IP Sec

- ESP trailer consists of three fields: padding; pad length; and next header. Recall that block ciphers require the message to be encrypted to be an integer multiple of the block length.
- Padding (consisting of meaningless bytes) is used so that when added to the original datagram (along with the pad length and next header fields), the resulting “message” is an integer number of blocks.
- The pad-length field indicates to the receiving entity how much padding was inserted (and thus needs to be removed).
- The next header identifies the type (e.g., UDP) of data contained in the payload-data field. The payload data (typically the original IP datagram) and the ESP trailer are concatenated and then encrypted.

IP Sec

- ESP header, which is sent in the clear and consists of two fields: the SPI and the sequence number field.
- The SPI indicates to the receiving entity the SA to which the datagram belongs; the receiving entity can then index its SAD with the SPI to determine the appropriate authentication/decryption algorithms and keys. The sequence number field is used to defend against replay attacks.

IPSec

- Appends to the back of the original IPv4 datagram (which includes the original header fields!) an “ESP trailer” field
- Encrypts the result using the algorithm and key specified by the SA
- Appends to the front of this encrypted quantity a field called “ESP header”; the
- resulting package is called the “enchilada”
- Creates an authentication MAC over the whole enchilada using the algorithm and key specified in the SA
- Appends the MAC to the back of the enchilada forming the payload
- Finally, creates a brand new IP header with all the classic IPv4 header fields
- (together normally 20 bytes long), which it appends before the payload

IP Sec

- The sending entity also appends an authentication MAC.
- The sending entity calculates a MAC over the whole enchilada (consisting of the ESP header, the original IP datagram, and the ESP trailer—with the datagram and trailer being encrypted).
- Recall that to calculate a MAC, the sender appends a secret MAC key to the enchilada and then calculates a fixed-length hash of the result.

IPSec

- Q: But what about the source and destination IP addresses that are in the new IP header, that is, in the left-most header of the IPsec datagram?
- They are set to the source and destination router interfaces at the two ends of the tunnels, namely, 200.168.1.100 and 193.68.2.23. Also, the protocol number in this new IPv4 header field is not set to that of TCP, UDP, or SMTP, but instead to 50, designating that this is an IPsec datagram using the ESP protocol.
- https://www.youtube.com/watch?v=Mf2IDIU_Yvg

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
 - doesn't keep track of *all* received packets; instead uses a window

IPsec security databases

Security Policy Database (SPD)

- policy: for given datagram, sender needs to know if it should use IP sec
- policy stored in **security policy database (SPD)**
- needs to know which SA to use
 - may use: source and destination IP address; protocol number

SAD: “how” to do it

Security Assoc. Database (SAD)

- endpoint holds SA state in **security association database (SAD)**
- 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, processing datagram accordingly.

SPD: “what” to do

Summary: IPsec services



Trudy sits somewhere between R1, R2. she doesn't know the keys

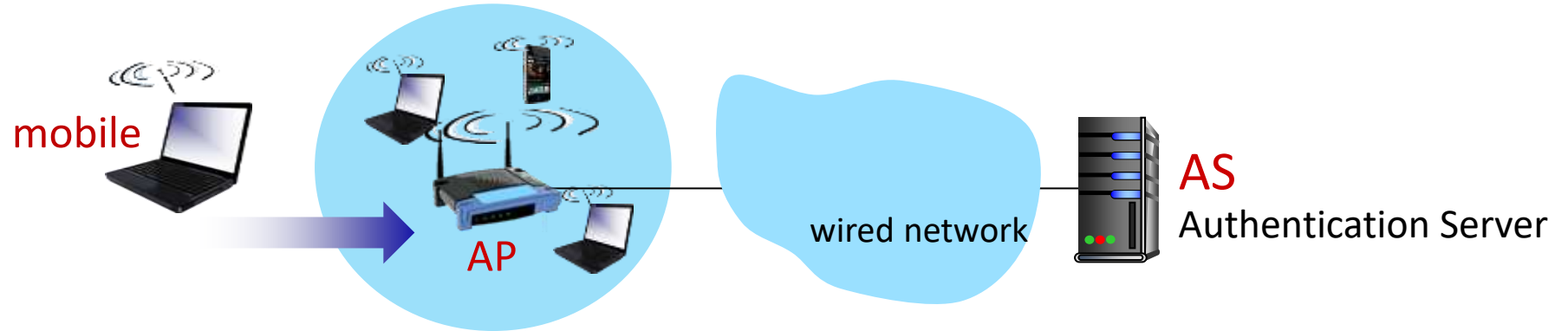
- will Trudy be able to see original contents of datagram? How about source, dest IP address, transport protocol, application port?
- flip bits without detection?
- masquerade as R1 using R1's IP address?
- replay a datagram?

Chapter 8 outline

- What is network security?
- Principles of cryptography
- Authentication, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- **Security in wireless and mobile networks**
 - 802.11 (WiFi)
 - 4G/5G
- Operational security: firewalls and IDS



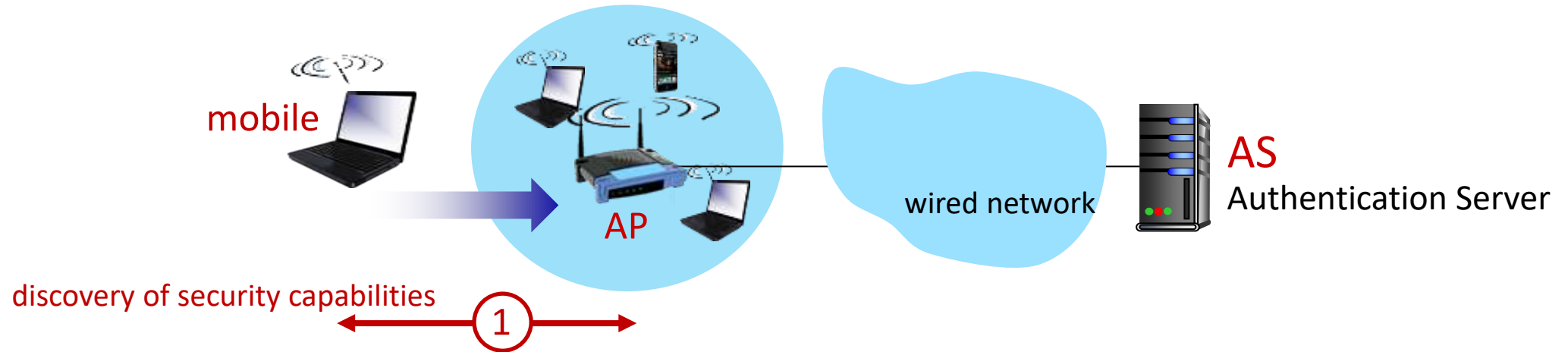
802.11: authentication, encryption



Arriving mobile must:

- associate with access point: (establish) communication over wireless link
- authenticate to network

802.11: authentication, encryption

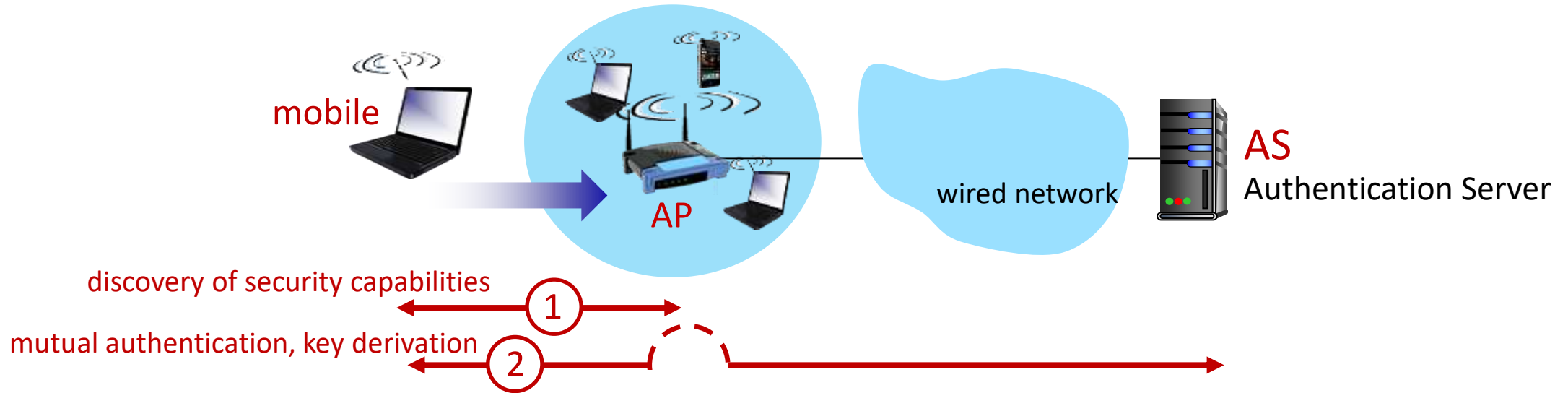


① discovery of security capabilities:

- AP advertises its presence, forms of authentication and encryption provided
- device requests specific forms authentication, encryption desired

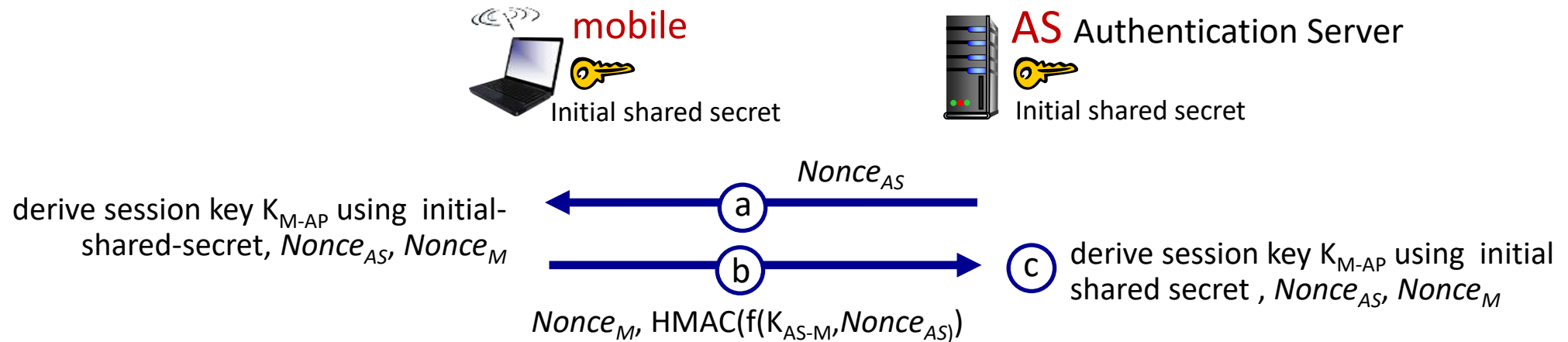
although device, AP already exchanging messages, device not yet authenticated, does not have encryption keys

802.11: authentication, encryption



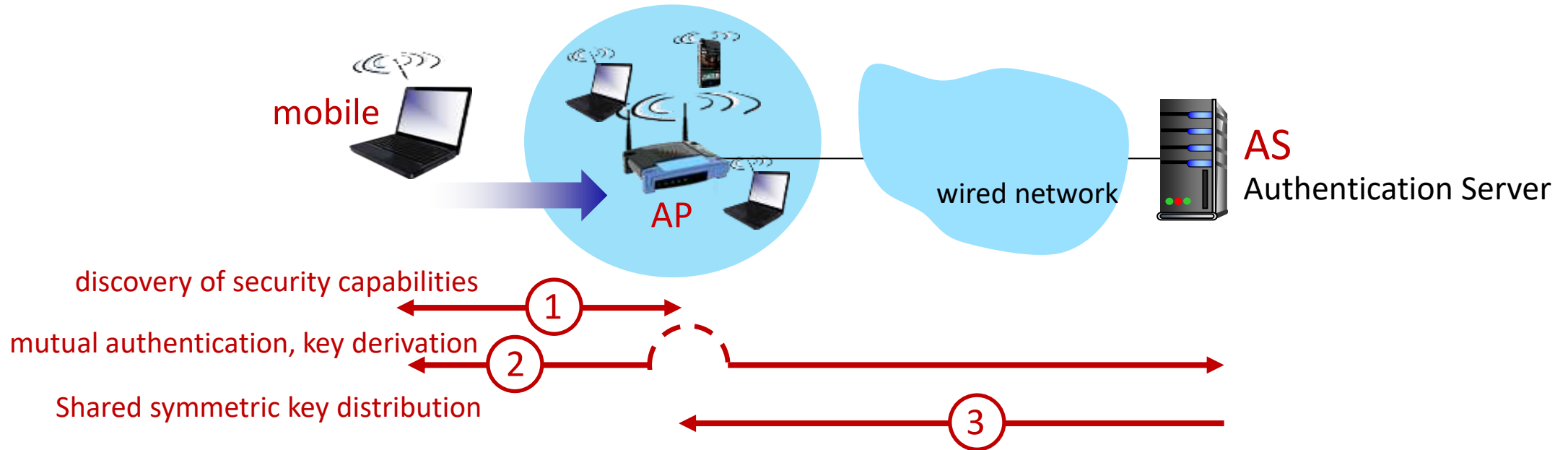
- ② mutual authentication and shared symmetric key derivation:
- AS, mobile already have shared common secret (e.g., password)
 - AS, mobile use shared secret, nonces (prevent relay attacks), cryptographic hashing (ensure message integrity) to authenticating each other
 - AS, mobile derive symmetric session key

802.11: WPA3 handshake



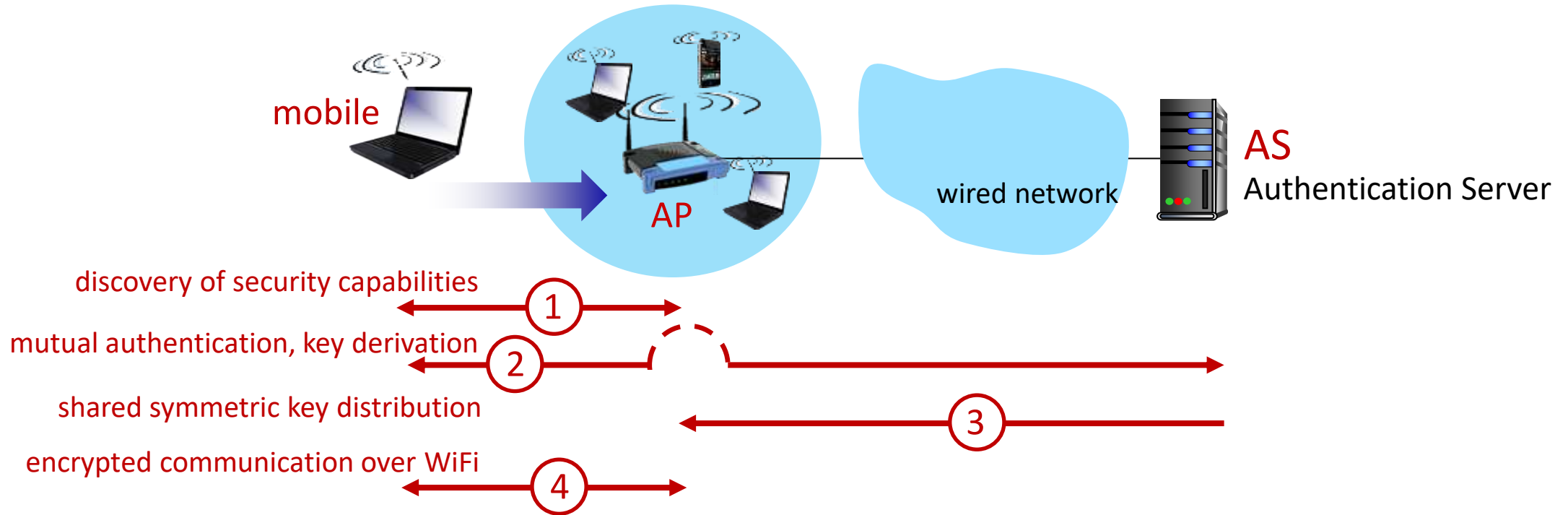
- ① AS generates $Nonce_{AS}$, sends to mobile
- ② mobile receives $Nonce_{AS}$
 - generates $Nonce_M$
 - generates symmetric shared session key K_{M-AP} using $Nonce_{AS}$, $Nonce_M$, and initial shared secret
 - sends $Nonce_M$ and HMAC-signed value using $Nonce_{AS}$ and initial shared secret
- ③ AS derives symmetric shared session key K_{M-AP}

802.11: authentication, encryption



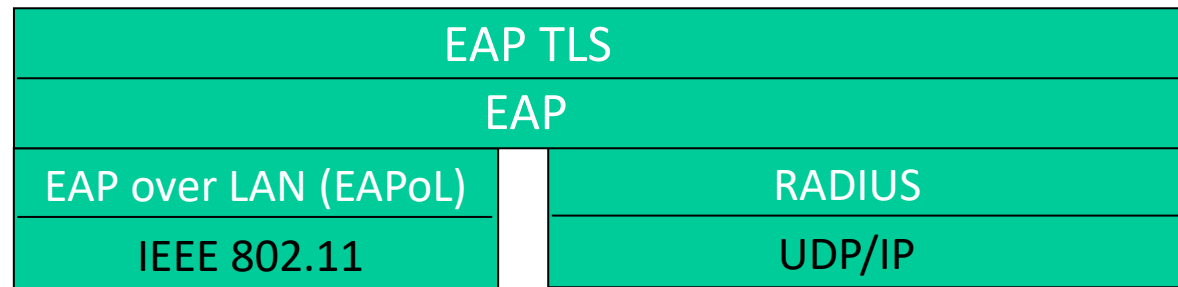
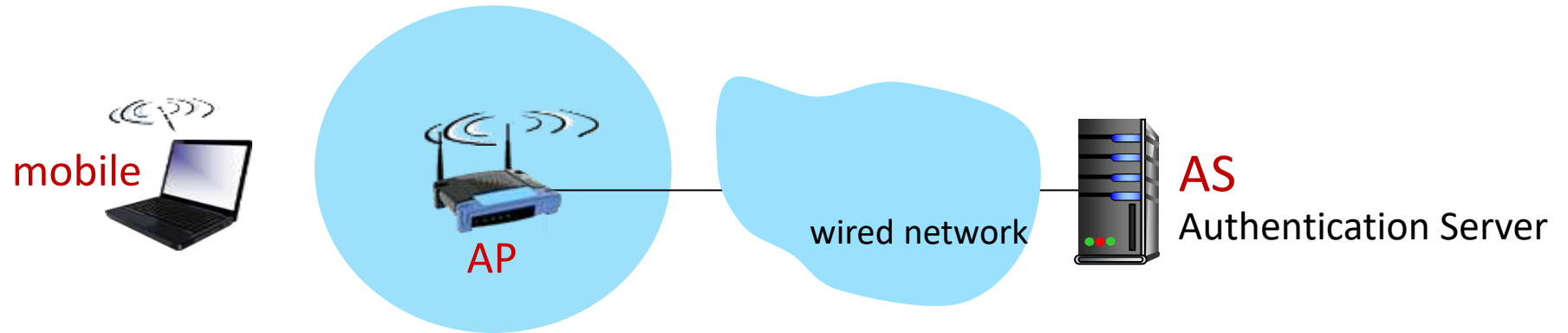
- ③ shared symmetric session key distribution (e.g., for AES encryption)
- same key derived at mobile, AS
 - AS informs AP of the shared symmetric session

802.11: authentication, encryption



- ④ encrypted communication between mobile and remote host via AP
- same key derived at mobile, AS
 - AS informs AP of the shared symmetric session

802.11: authentication, encryption



- Extensible Authentication Protocol (EAP) [RFC 3748] defines end-to-end request/response protocol between mobile device, AS

Chapter 8 outline

- What is network security?
- Principles of cryptography
- Authentication, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- **Security in wireless and mobile networks**
 - 802.11 (WiFi)
 - 4G/5G
- Operational security: firewalls and IDS



Authentication, encryption in 4G LTE



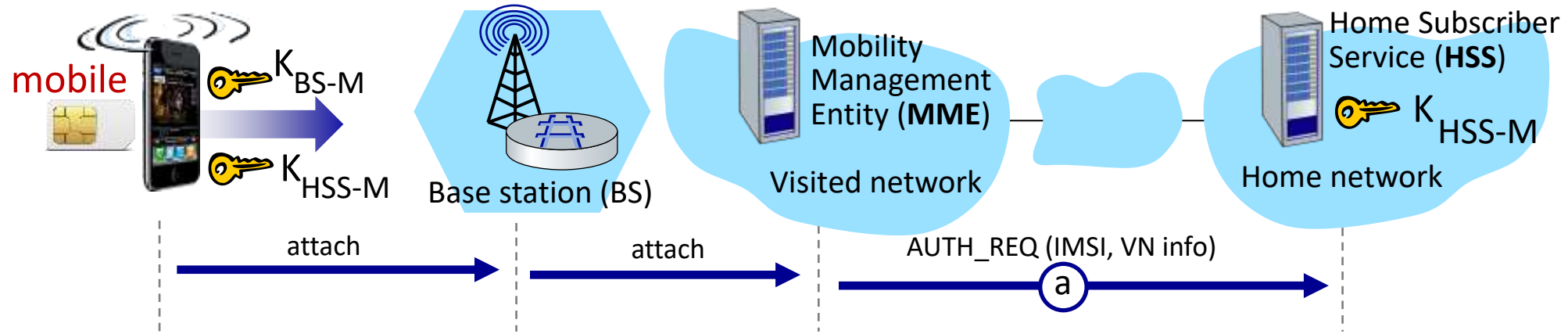
- arriving mobile must:
 - associate with BS: (establish) communication over 4G wireless link
 - authenticate itself to network, and authenticate network
- notable differences from WiFi
 - mobile's SIMcard provides global identity, contains shared keys
 - services in visited network depend on (paid) service subscription in home network

Authentication, encryption in 4G LTE



- mobile, BS use derived session key K_{BS-M} to encrypt communications over 4G link
- MME in visited network + HSS in home network, together play role of WiFi AS
 - ultimate authenticator is HSS
 - trust and business relationship between visited and home networks

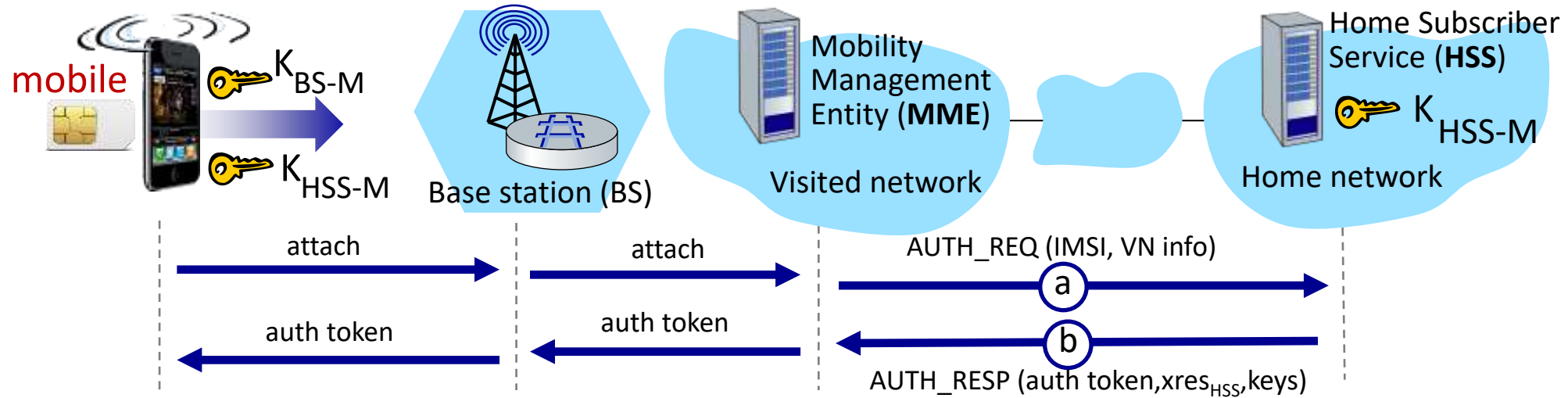
Authentication, encryption in 4G LTE



① authentication request to home network HSS

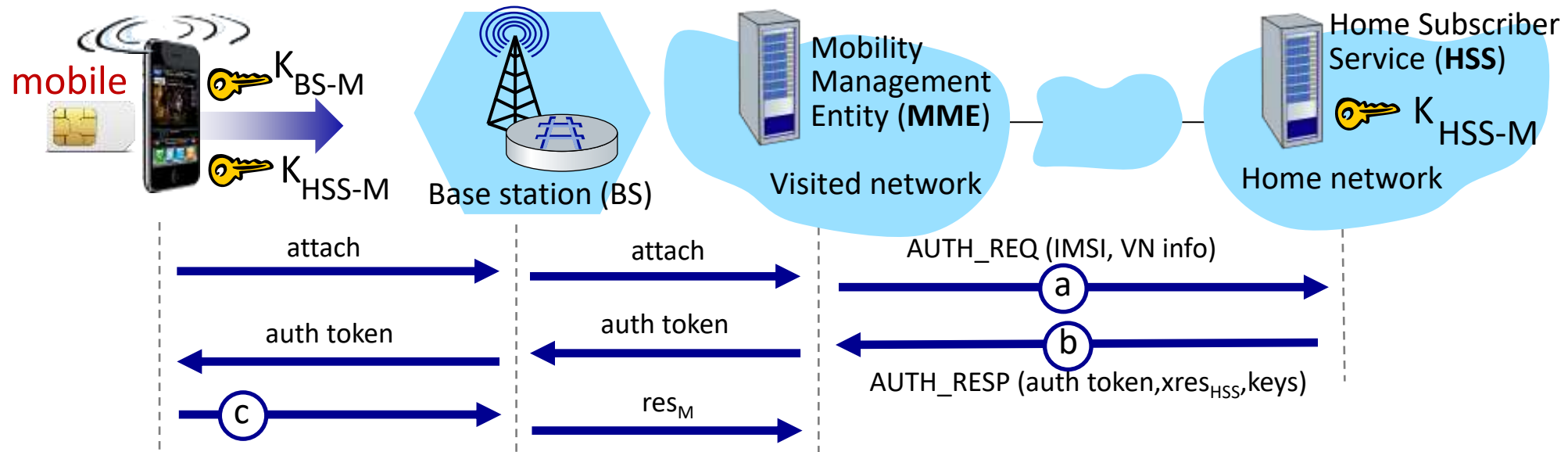
- mobile sends attach message (containing its IMSI, visited network info) relayed from BS to visited MME to home HSS
- IMSI identifies mobile's home network

Authentication, encryption in 4G LTE



- ② HSS use shared-in-advance secret key, K_{HSS-M} , to derive authentication token, *auth_token*, and expected authentication response token, $xres_{HSS}$
- *auth_token* contains info encrypted by HSS using K_{HSS-M} , allowing mobile to know that whoever computed *auth_token* knows shared-in-advance secret
 - mobile has authenticated network
 - visited HSS keeps $xres_{HSS}$ for later use

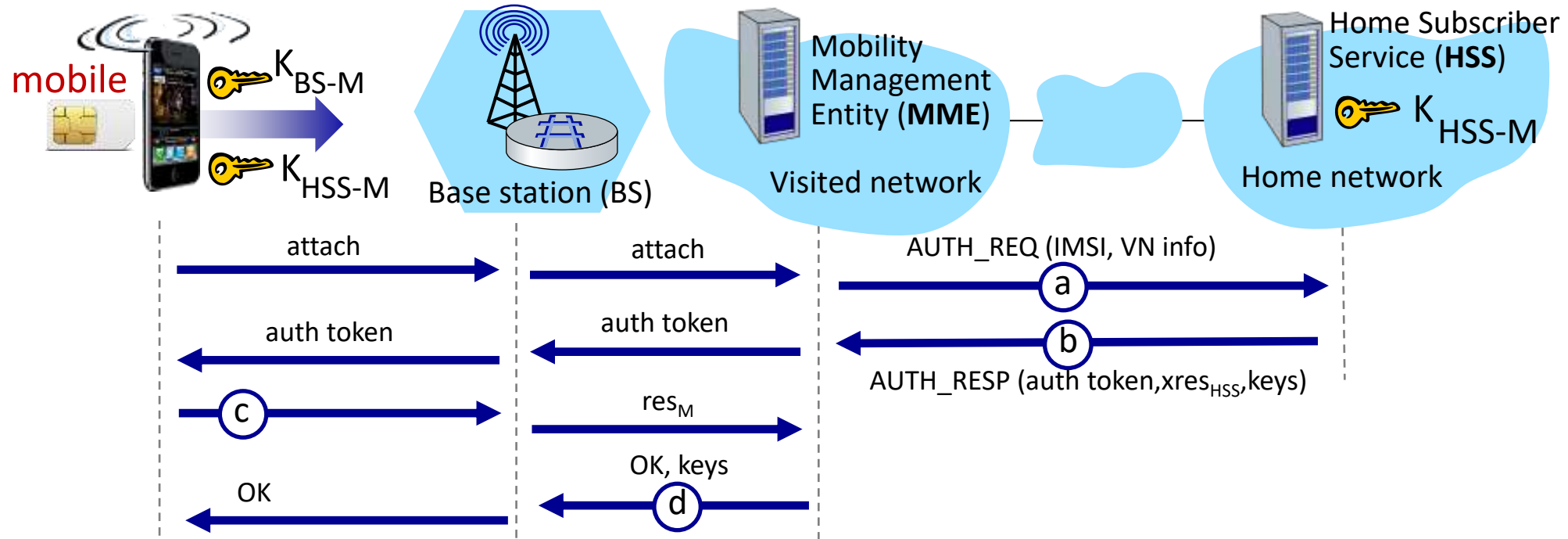
Authentication, encryption in 4G LTE



© authentication response from mobile:

- mobile computes res_M using its secret key to make same cryptographic calculation that HSS made to compute $xres_{HSS}$ and sends res_M to MME

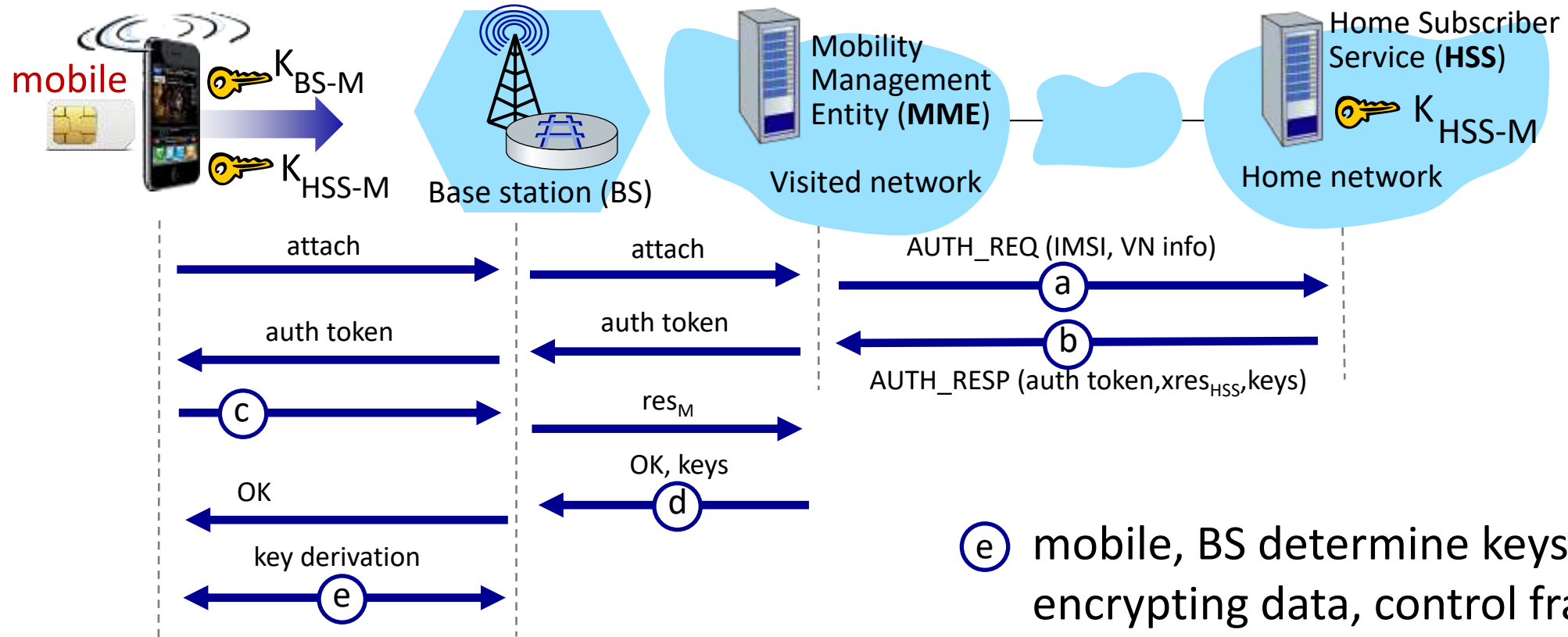
Authentication, encryption in 4G LTE



④ mobile is authenticated by network:

- MMS compares mobile-computed value of res_M with the HSS-computed value of $xres_{HSS}$. If they match, mobile is authenticated ! (why?)
- MMS informs BS that mobile is authenticated, generates keys for BS

Authentication, encryption in 4G LTE



- ⑤ mobile, BS determine keys for encrypting data, control frames over 4G wireless channel
 - AES can be used

Authentication, encryption: from 4G to 5G

- **4G:** MME in visited network makes authentication decision
- **5G:** home network provides authentication decision
 - visited MME plays “middleman” role but can still reject
- **4G:** uses shared-in-advance keys
- **5G:** keys not shared in advance for IoT
- **4G:** device IMSI transmitted in cleartext to BS
- **5G:** public key crypto used to encrypt IMSI

Chapter 8 outline

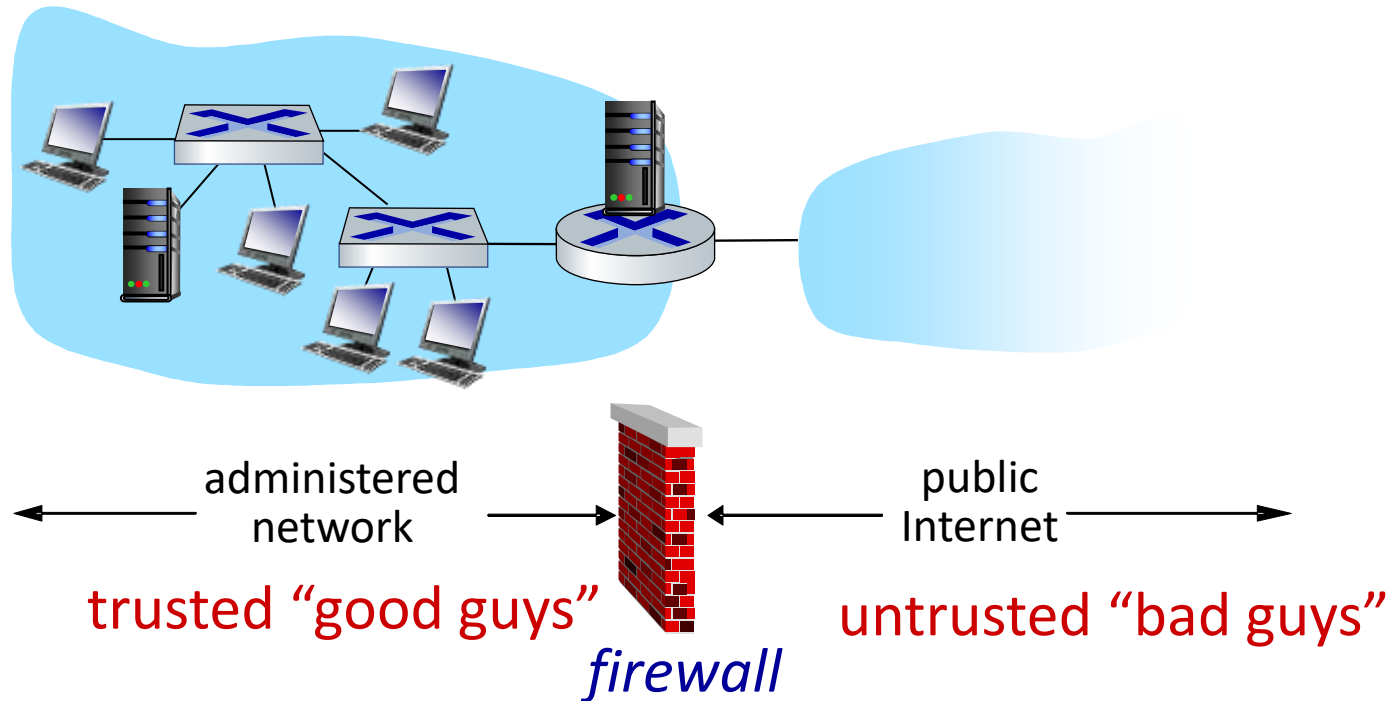
- What is network security?
- Principles of cryptography
- Authentication, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- **Operational security: firewalls and IDS**



Firewalls

firewall

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



Firewalls: why

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections

prevent illegal modification/access of internal data

- e.g., attacker replaces CIA’s homepage with something else

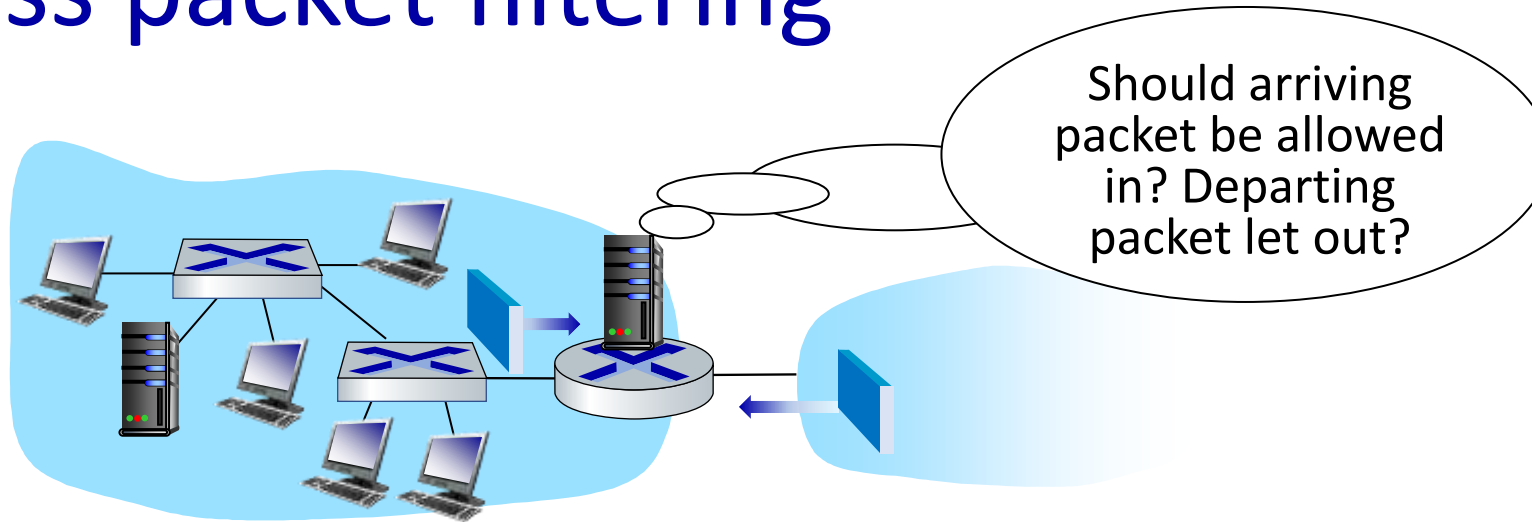
allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

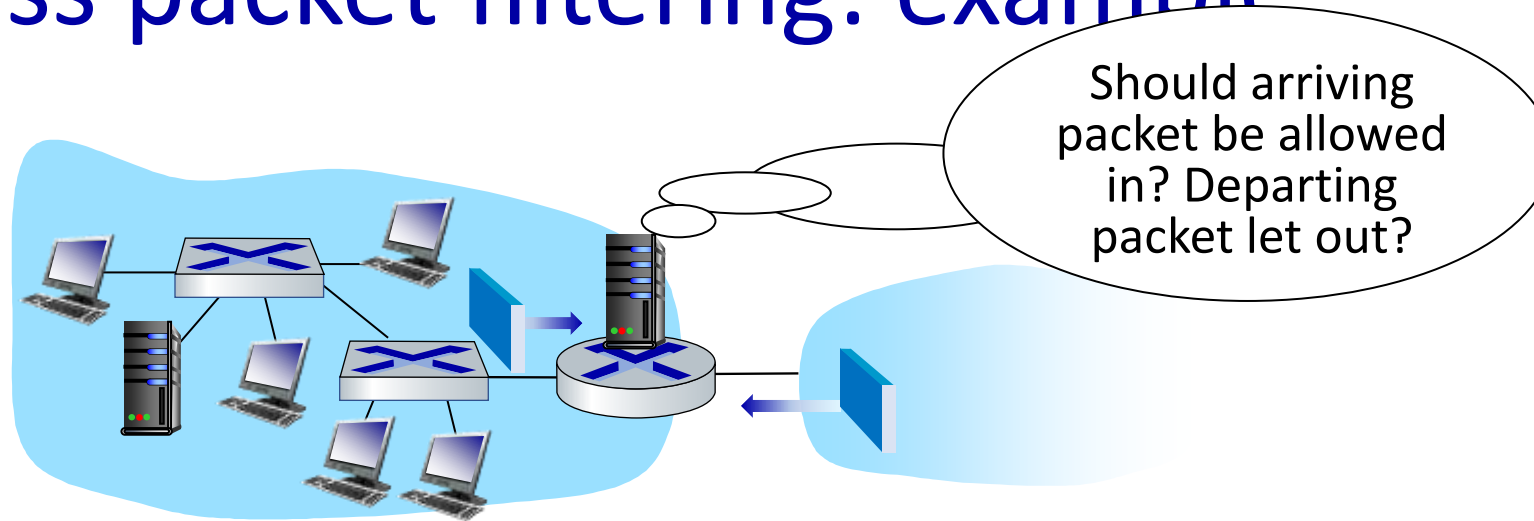
- stateless packet filters
- stateful packet filters
- application gateways

Stateless packet filtering



- internal network connected to Internet via router **firewall**
- filters **packet-by-packet**, decision to forward/drop packet based on:
 - source IP address, destination IP address
 - TCP/UDP source, destination port numbers
 - ICMP message type
 - TCP SYN, 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: looks like OpenFlow forwarding (Ch. 4)!

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

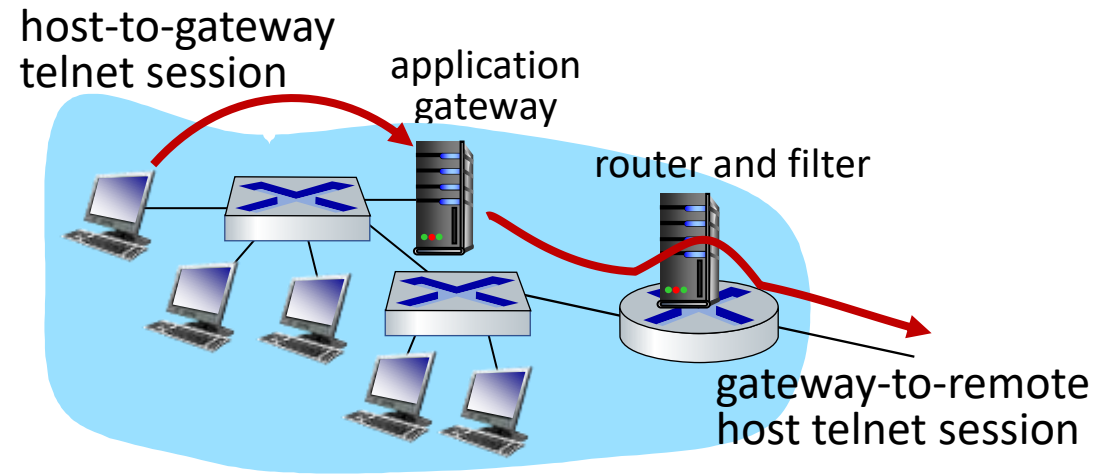
Stateful packet filtering

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 connection
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	

Application gateways

- filter packets on application data as well as on IP/TCP/UDP fields.
- *example:* allow select internal users to telnet outside



1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host
 - gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway

Limitations of firewalls, gateways

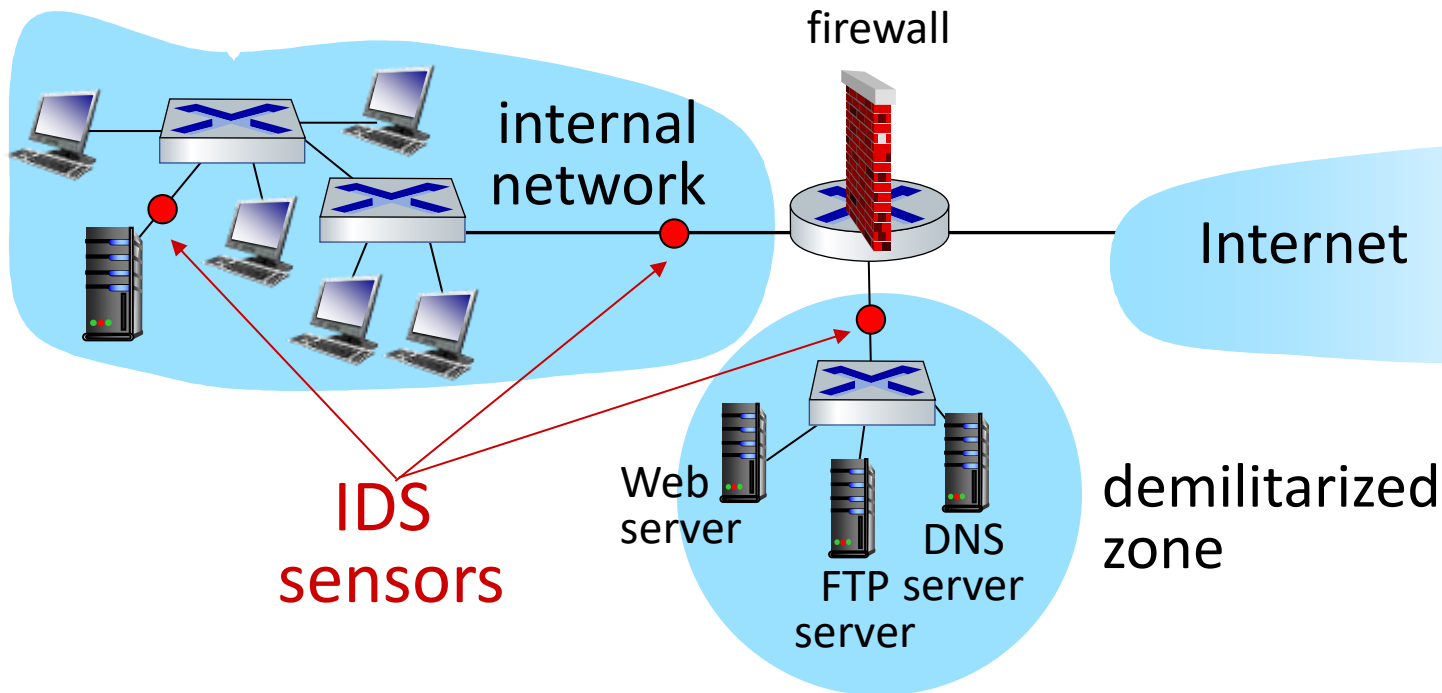
- **IP spoofing:** router can't know if data “really” comes from claimed source
- if multiple apps need special treatment, each has own app. gateway
- client software must know how to contact gateway
 - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP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

Intrusion detection systems

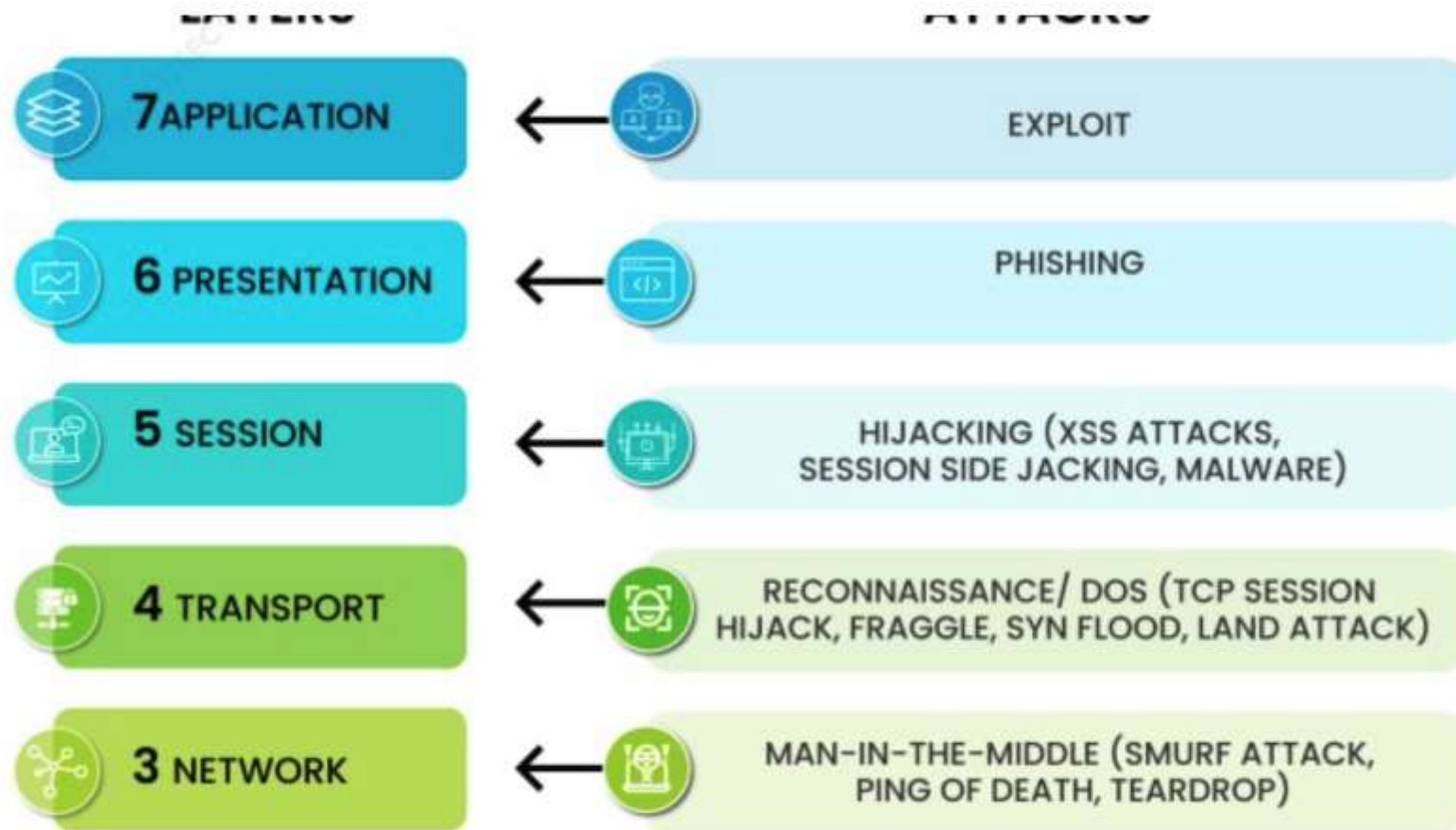
- packet filtering:
 - operates on TCP/IP headers only
 - no correlation check among sessions
- IDS: intrusion detection system
 - deep packet inspection: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
 - examine correlation among multiple packets
 - port scanning
 - network mapping
 - DoS attack

Intrusion detection systems

multiple IDSs: different types of checking at different locations



Attack on OSI layers



Attack on OSI layers

- Layer 1 - Physical:
 - Traffic eavesdropping: Intercepting and capturing data transmitted over physical media, such as cables or wireless signals.
- Layer 2 - Data Link:
 - MAC spoofing: Modifying the Media Access Control (MAC) address to impersonate another device on the network.
 - ARP spoofing: Sending forged Address Resolution Protocol (ARP) messages to associate the attacker's MAC address with the IP address of another device.
 - VLAN hopping: Exploiting misconfigurations in Virtual LAN (VLAN) implementations to gain unauthorized access to traffic in different VLANs.
 - DHCP spoofing: Impersonating a DHCP server to allocate incorrect or malicious IP addresses to network devices.
 - Rogue access points: Setting up unauthorized wireless access points to trick users into connecting to them, leading to potential data interception.

Attack on OSI layers

- Layer 3 - Network:
 - IP spoofing: Manipulating the source IP address in IP packets to make it appear as if they are originating from a trusted source.
 - Manipulating routing tables: Modifying routing tables to redirect network traffic to unauthorized destinations.
 - ICMP redirect: Sending ICMP redirect messages to modify a host's routing table and redirect traffic through an attacker-controlled system.
 - TCP/UDP flood (DDoS): Overwhelming a target system with a flood of TCP or UDP packets to consume its resources and make it unavailable.
 - SYN flood (DDoS): Exploiting the TCP three-way handshake by flooding a target system with a high volume of SYN requests, exhausting its resources.
 - Smurf attack (DDoS): Broadcasting ICMP echo requests with the victim's spoofed IP address, causing multiple hosts to respond and flood the victim with ICMP replies

Attack on OSI layers

- Layer 4 - Transport:
 - Lateral Movement: Exploiting vulnerabilities in network services to move laterally within a network and gain unauthorized access to other systems.
 - TCP/UDP port scanning: Scanning a target system's open ports to identify potential vulnerabilities or services running on specific ports.
 - DNS poisoning: Manipulating DNS (Domain Name System) responses to redirect users to malicious websites or incorrect IP addresses.
 - TCP/UDP flood (DDoS): Similar to Layer 3 attacks, overwhelming a target system with a flood of TCP or UDP packets.

Attack on OSI layers

- Layer 5 - Session:
 - Access control bypass: Exploiting weaknesses in session management or authentication mechanisms to bypass access controls and gain unauthorized access.
 - Adversary-in-the-middle attack: Intercepting and altering communication between two parties by positioning oneself as an intermediary.

Attack on OSI layers

- Layer 6 - Presentation:

- Cracking encryption: Attempting to break cryptographic algorithms or obtain encryption keys to access encrypted data.
- Injection attacks: Inserting malicious code or commands into data fields or input forms to exploit vulnerabilities in an application.
- File inclusion vulnerabilities: Exploiting insecure file inclusion mechanisms in web applications to access unauthorized files.
- Cross-site scripting (XSS): Injecting malicious scripts into web pages viewed by other users, allowing the attacker to steal information or perform unauthorized actions.
- Cross-site request forgery (CSRF): Forcing a user's browser to perform unwanted actions on a web application on behalf of the attacker without the user's knowledge.

Attack on OSI layers

- Layer 7 - Application:

- Phishing: Sending fraudulent emails or creating fake websites to deceive users into revealing sensitive information, such as passwords or credit card details.
- Password cracking: Attempting to discover passwords by using various techniques, such as dictionary attacks or brute-force methods.
- Buffer overflow: Exploiting a vulnerability in an application's input validation to overflow a buffer and execute arbitrary code.
- Format string attack: Exploiting vulnerabilities in the handling of format strings in an application to execute unauthorized code or leak sensitive information.

Network Security (summary)

basic techniques.....

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

.... used in many different security scenarios

- secure email
- secure transport (TLS)
- IP sec
- 802.11, 4G/5G

operational security: firewalls and IDS



Chapter 8 Additional Slides

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

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

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

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).

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,
$$\begin{aligned}c^d \bmod n &= (m^e \bmod n)^d \bmod n \\&= m^{ed} \bmod n \\&= m^{(ed \bmod z)} \bmod n \\&= m^1 \bmod n \\&= m\end{aligned}$$

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!

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

follows directly from modular arithmetic:

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

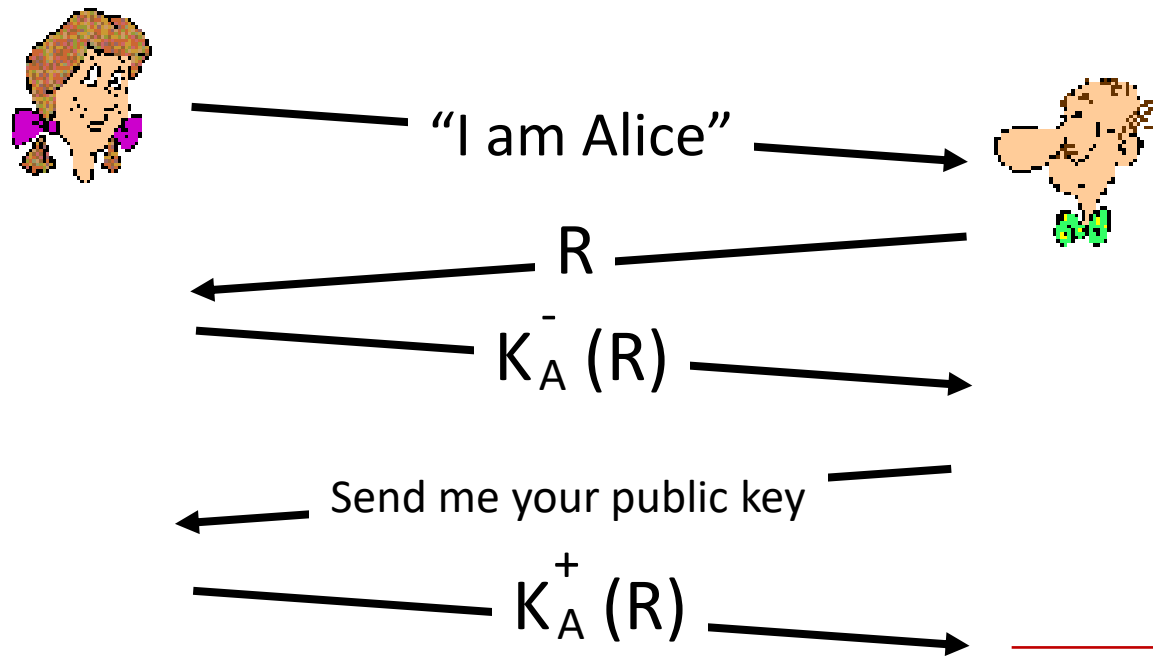
Why is RSA secure?

- 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

Authentication: ap5.0

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

ap5.0: use nonce, public key cryptography



Bob computes

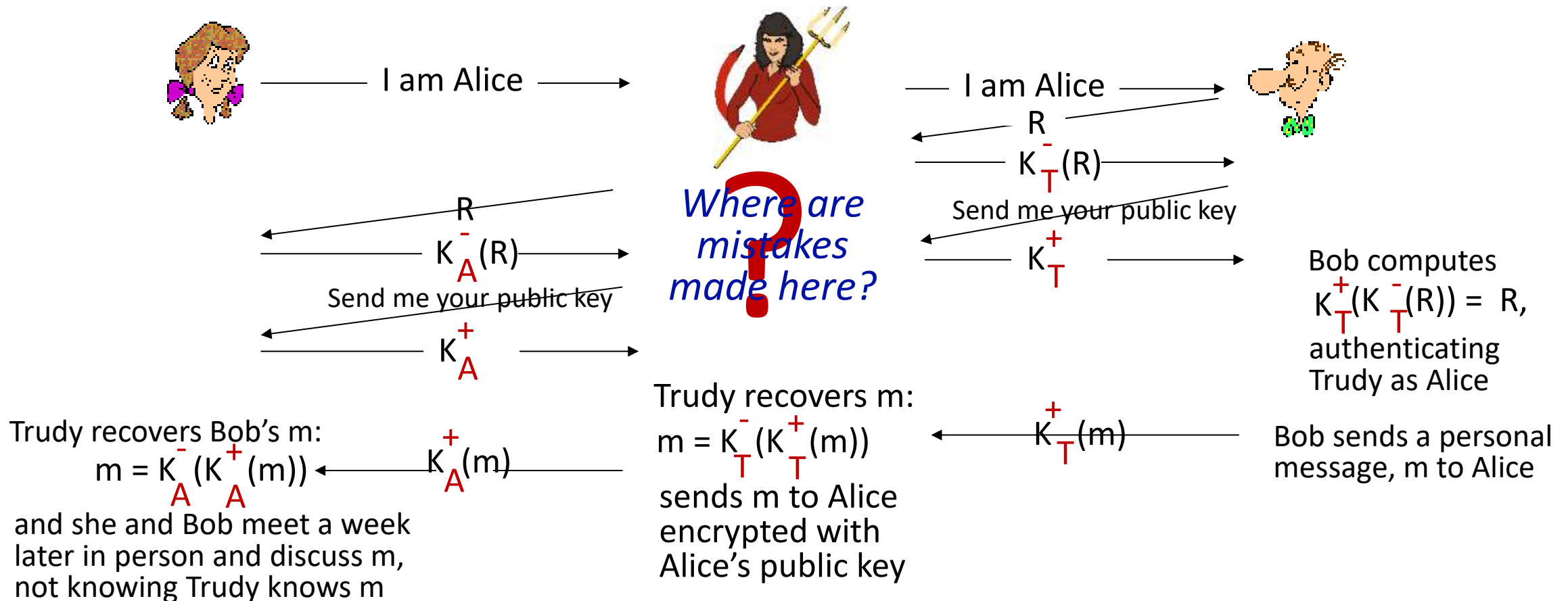
$$K_A^+ (K_A^-(R)) = R$$

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

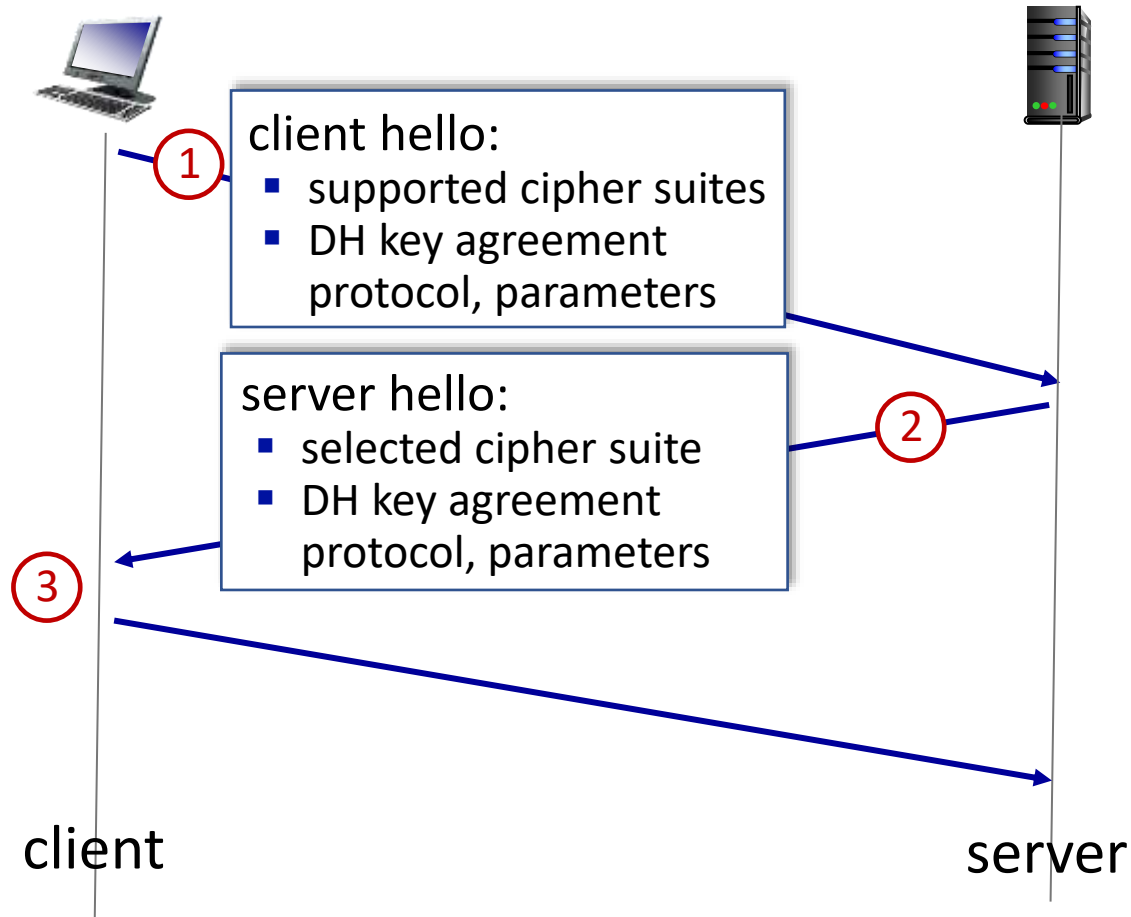
$$K_A^+ (K_A^-(R)) = R$$

Authentication: ap5.0 – there's still a flaw!

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

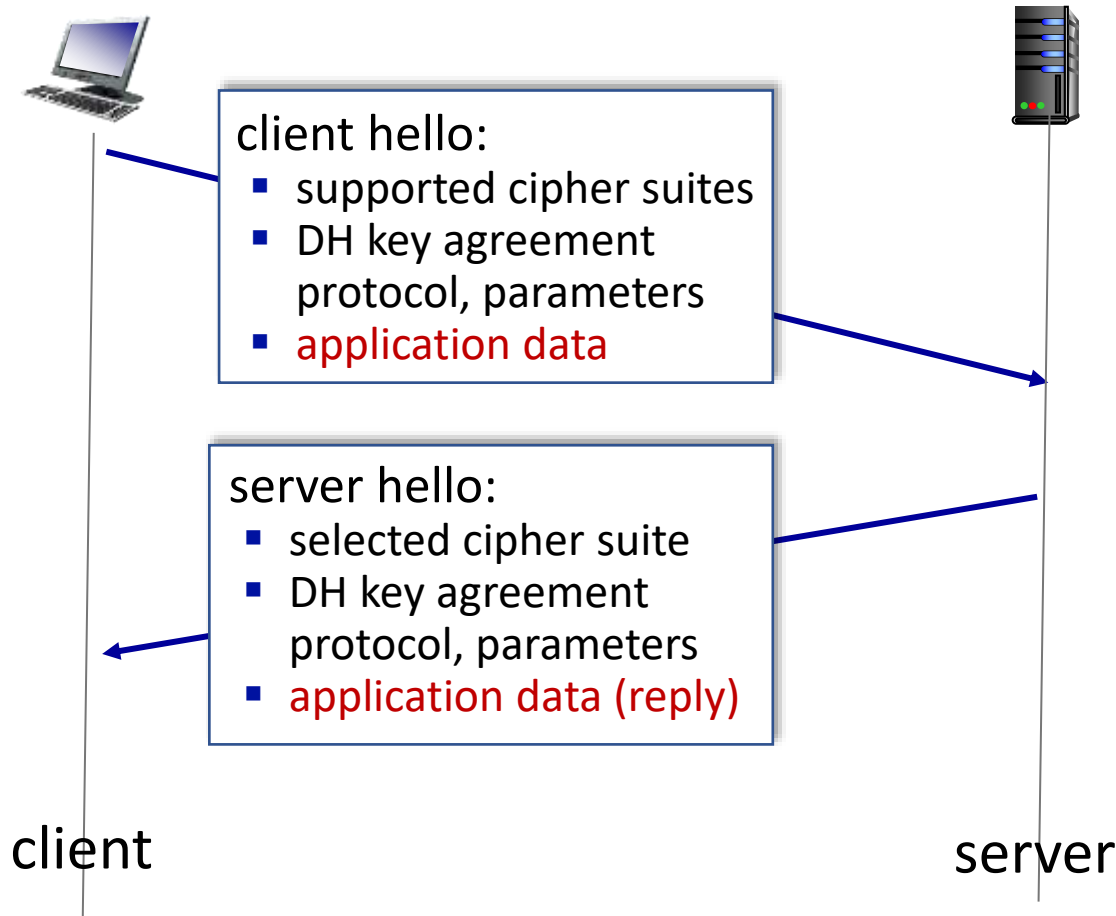


TLS 1.3 handshake: 1 RTT



- ① client TLS hello msg:
 - *guesses* key agreement protocol, parameters
 - indicates cipher suites it supports
- ② server TLS hello msg chooses
 - key agreement protocol, parameters
 - cipher suite
 - server-signed certificate
- ③ client:
 - checks server certificate
 - generates key
 - can now make application request (e.g., HTTPS GET)

TLS 1.3 handshake: 0 RTT



- initial hello message contains encrypted application data!
 - “resuming” earlier connection between client and server
 - application data encrypted using “resumption master secret” from earlier connection
- vulnerable to replay attacks!
 - maybe OK for get HTTP GET or client requests not modifying server state

IKE: Internet Key Exchange

- *previous examples:* manual establishment of 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...

- manual keying is impractical for VPN with 100s of endpoints
- instead use **IPsec IKE (Internet Key Exchange)**

IKE: PSK and PKI

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

IKE phases

- IKE has two phases
 - *phase 1*: establish bi-directional IKE SA
 - note: IKE SA different from IPsec SA
 - aka ISAKMP security association
 - *phase 2*: ISAKMP is used to securely negotiate IPsec pair of SAs
- phase 1 has two modes: aggressive mode and main mode
 - aggressive mode uses fewer messages
 - main mode provides identity protection and is more flexible