

Chapter 8

Security

A note on the use of these PowerPoint slides:

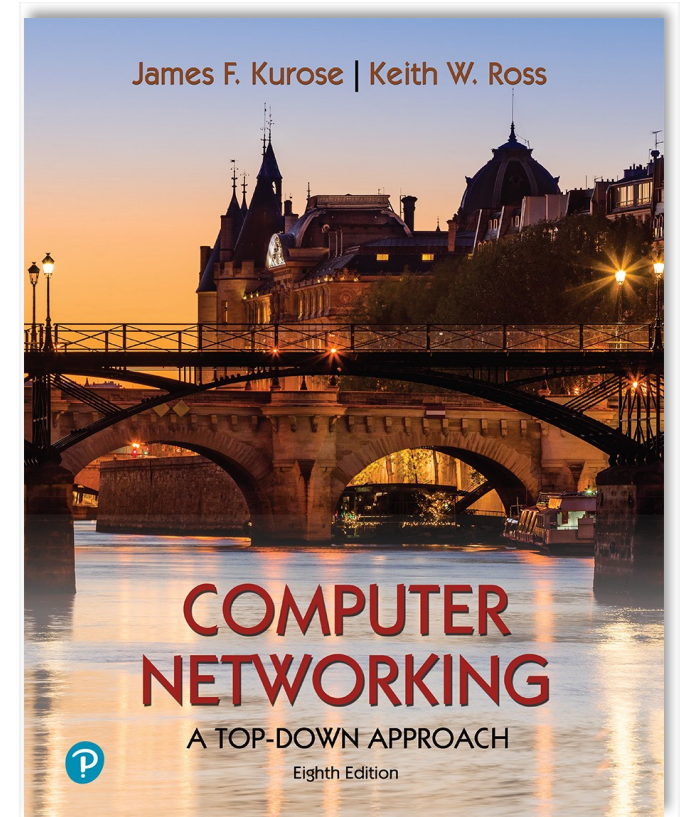
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, 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 1996-2020
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

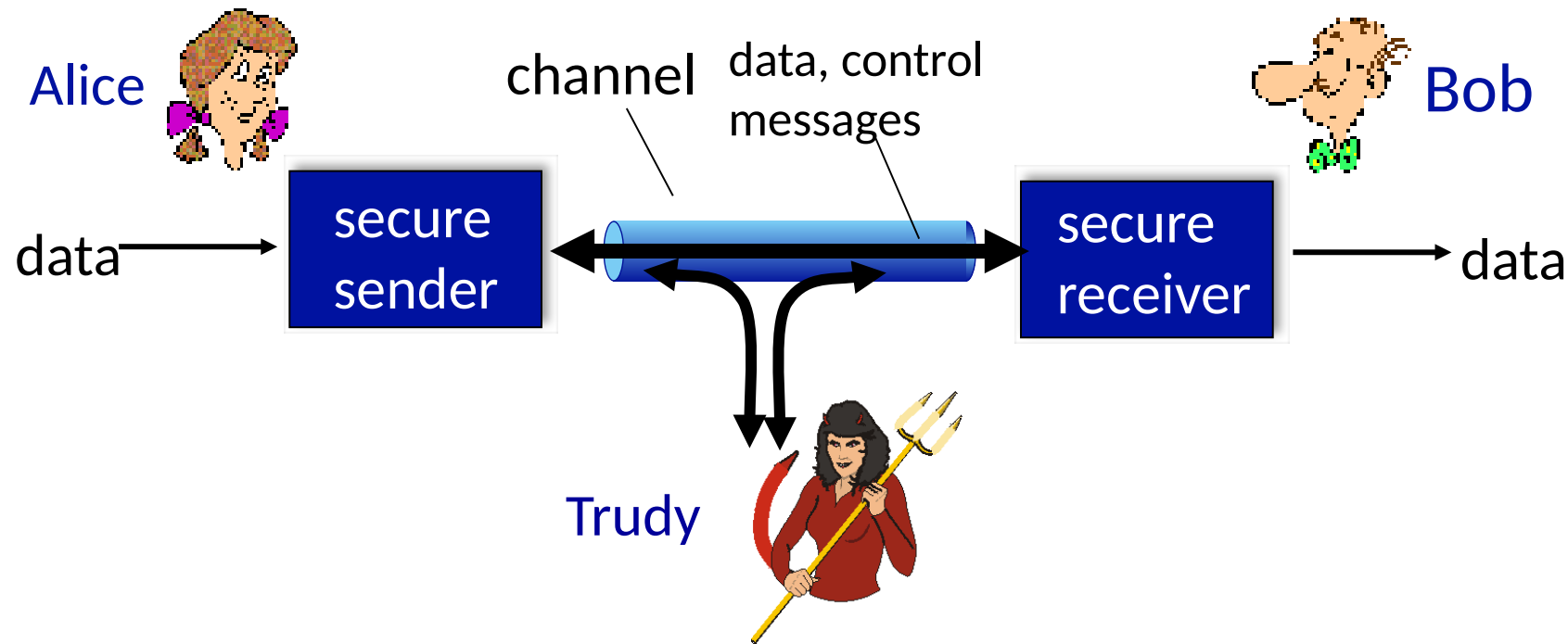
authentication: sender, receiver want to confirm identity of each other

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

access and availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

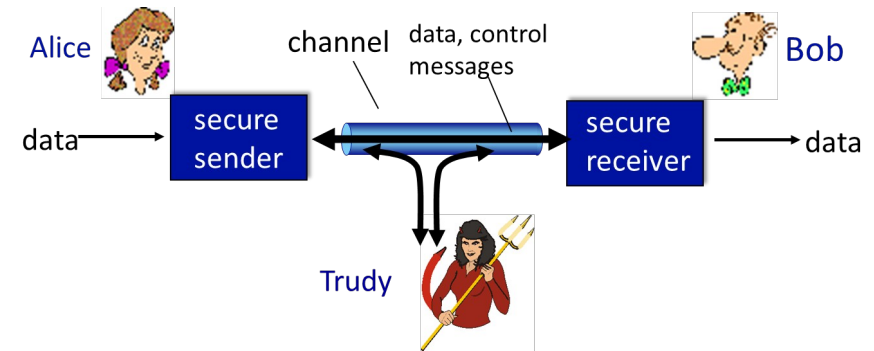
- well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



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
- DNS servers
- BGP routers exchanging routing table updates
- other examples ...

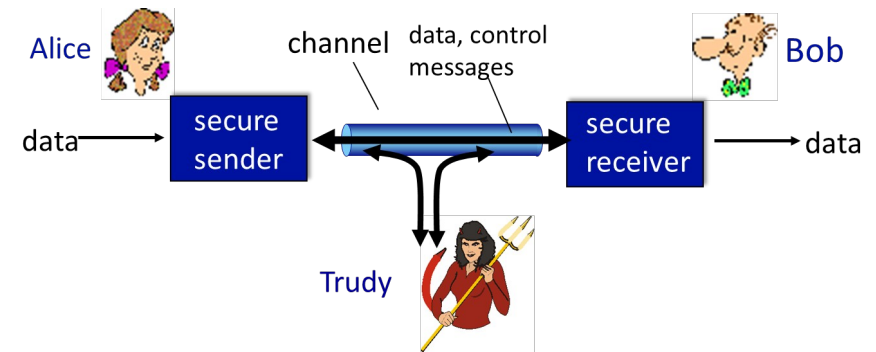


There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

A: A lot! (recall section 1.6)

- **eavesdrop**: intercept messages
- actively **insert** messages into connection
- **impersonation**: fake/spoof source address (or any other 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)

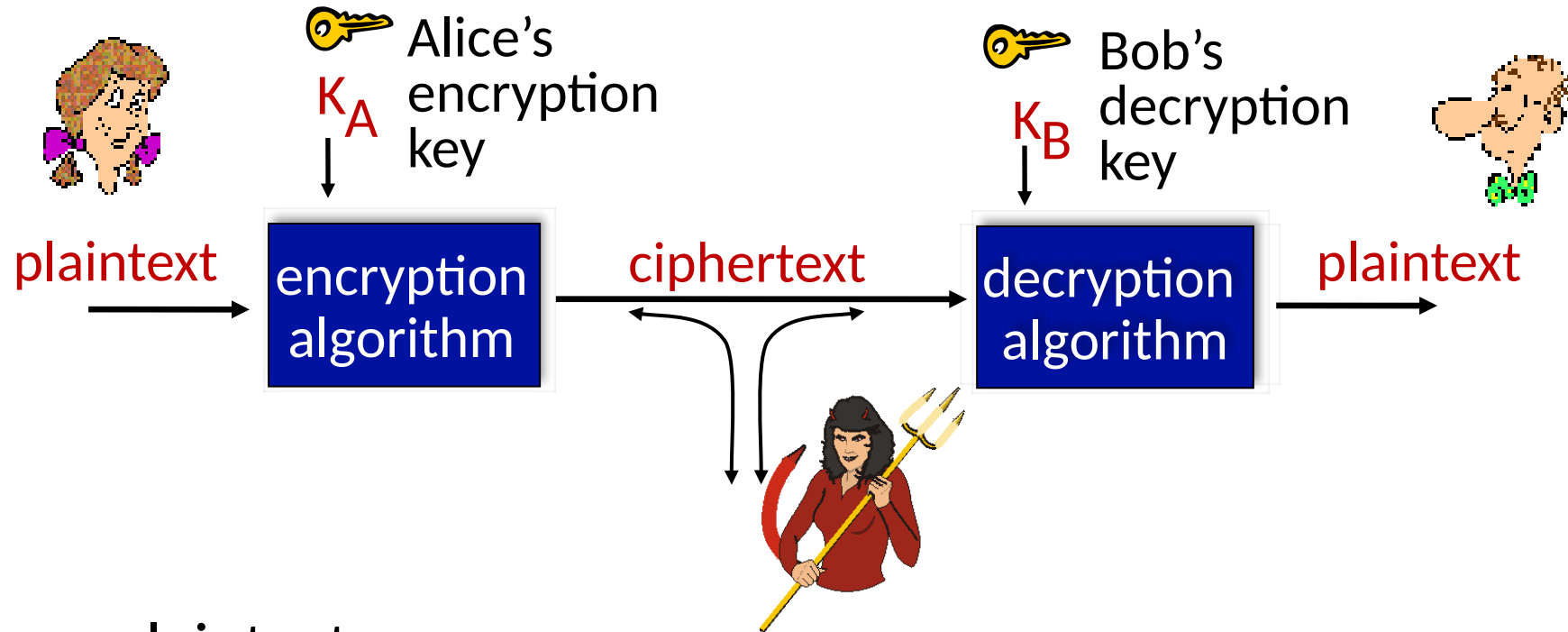


Chapter 8 outline

- 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

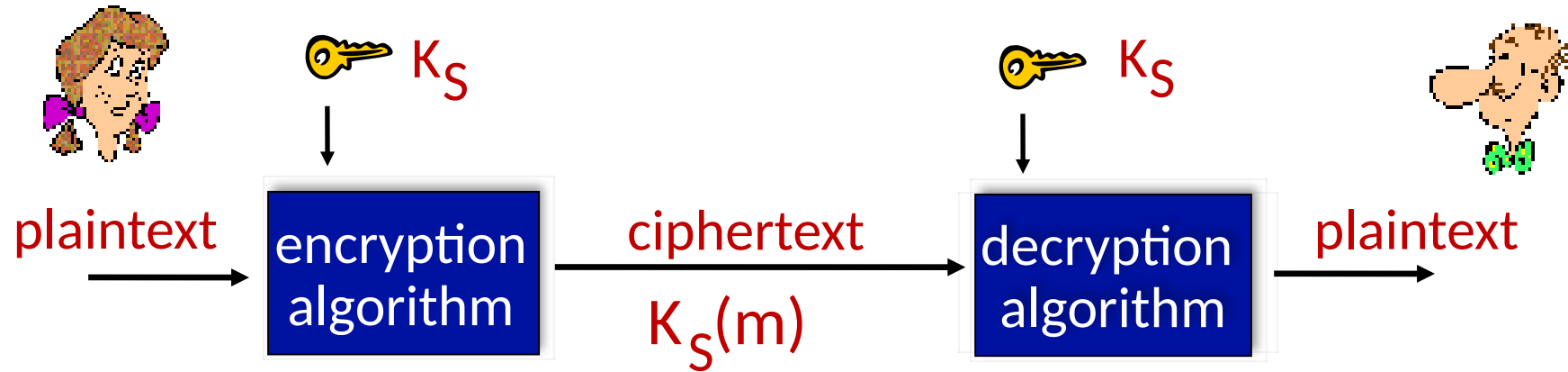


m : plaintext message

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

$m = K_B(K_A(m))$

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K

- e.g., key is knowing substitution pattern in substitution cipher

Q: how do Bob and Alice agree on key value?

Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

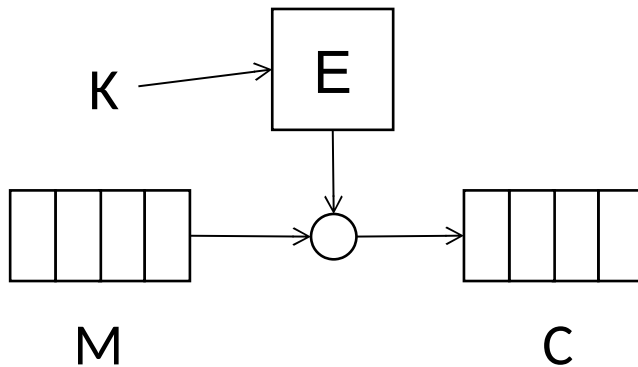
plaintext:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
		↓																							↓	
ciphertext:	m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

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

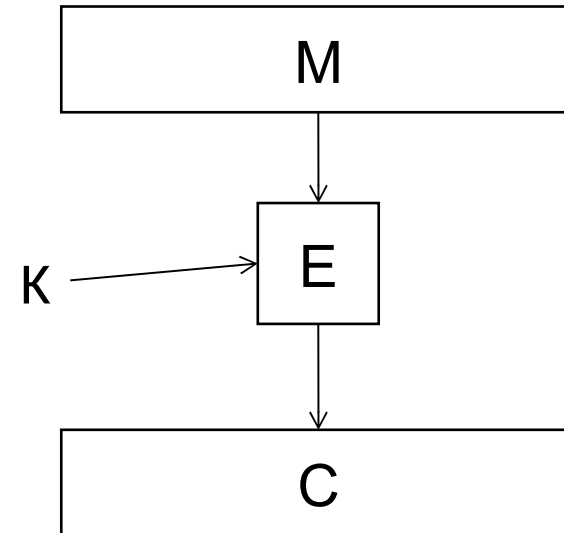
🔑 *Encryption key*: mapping from set of 26 letters
to set of 26 letters

Main types of symmetric ciphers

Stream cipher

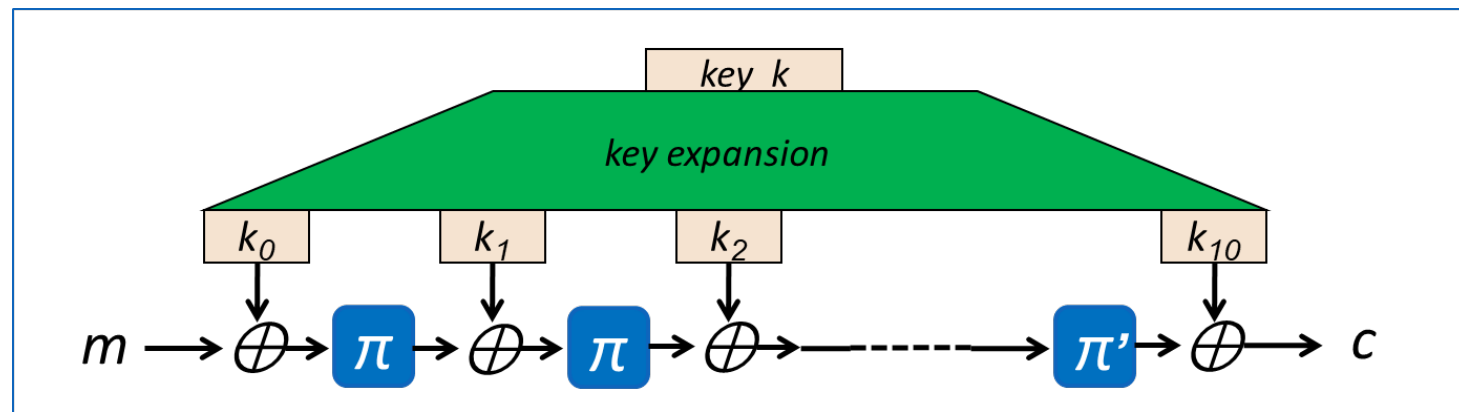


Block cipher



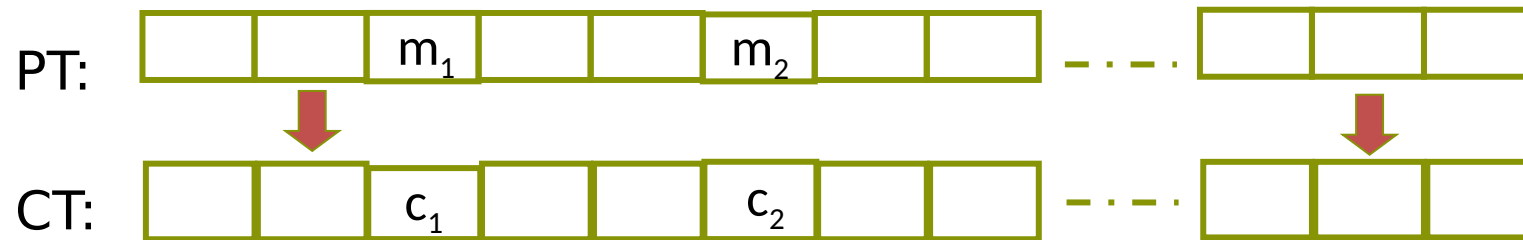
AES: Advanced Encryption Standard

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



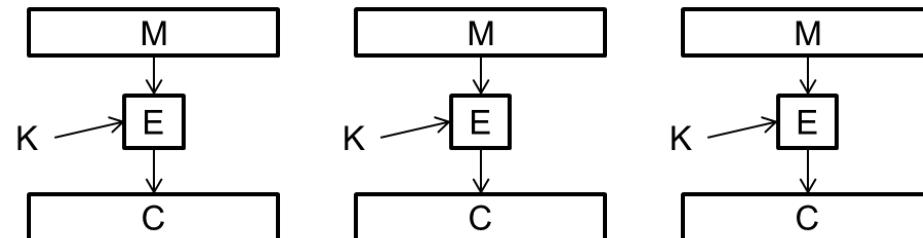
Incorrect use of block ciphers

Electronic Code Book (ECB):



Problem:

- if $m_1 = m_2$ then $c_1 = c_2$

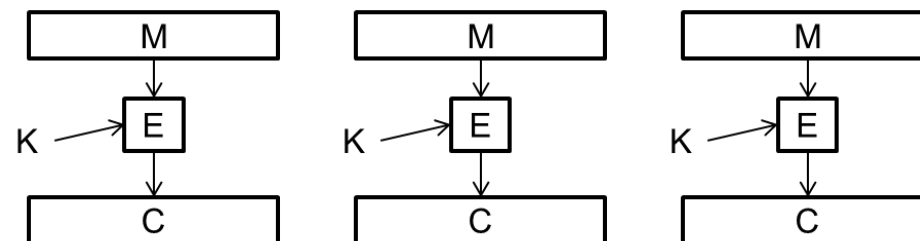
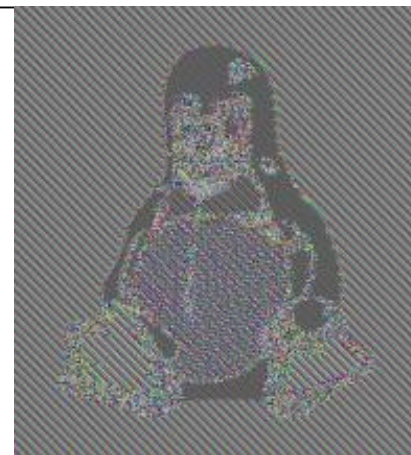
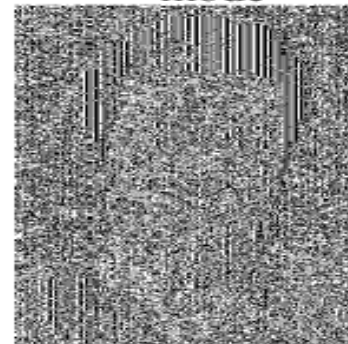


In pictures

An example plaintext

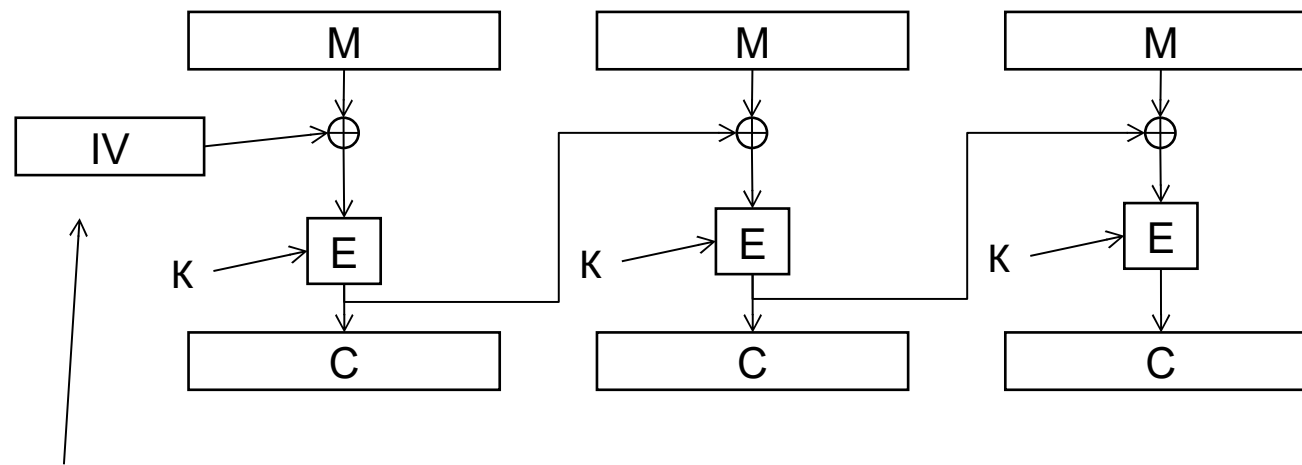


Encrypted with AES in ECB mode



Cipher Block Chaining (CBC)

Basic idea: Use previous cipher text as input to encryption of next block.



Initialization
vector (IV)



Public Key Cryptography

symmetric key crypto:

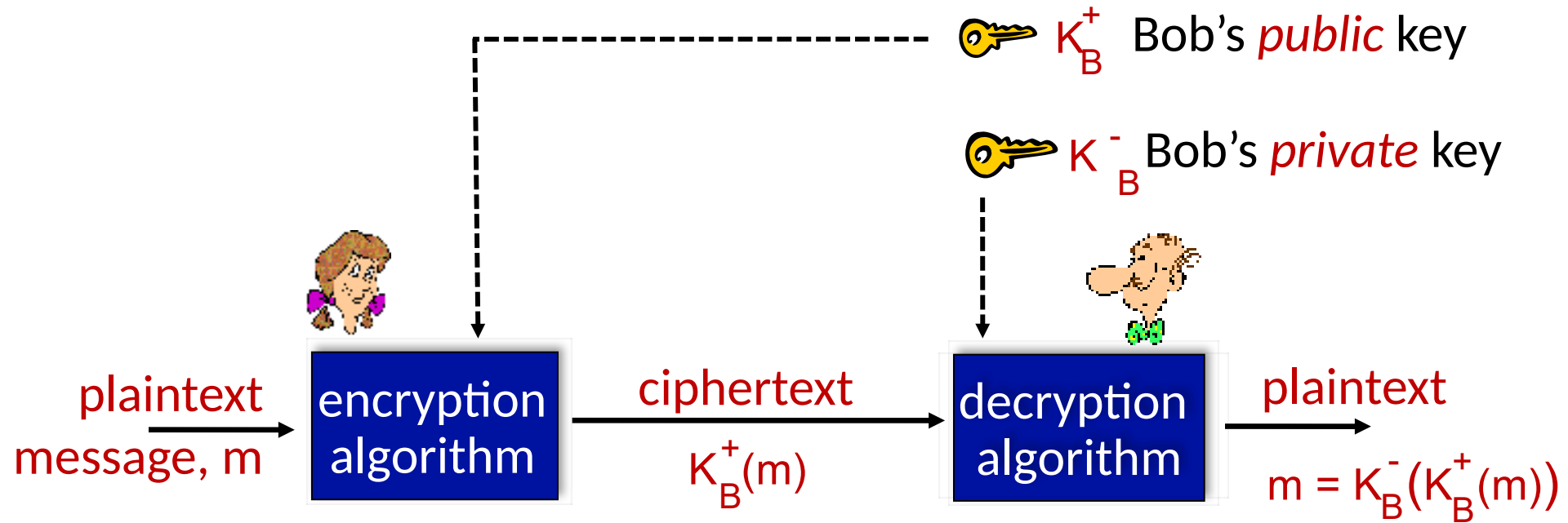
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never “met”)?

public key crypto

- *radically* different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver



Public Key Cryptography



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

- similar ideas emerged at roughly same time, independently in US and UK (classified)

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 . (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) .

$\underbrace{(n, e)}_{K_B^+}$

$\underbrace{(n, d)}_{K_B^-}$

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: 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

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_s

- Bob and Alice use RSA to exchange a symmetric session key K_s
- once both have K_s , they use symmetric key cryptography

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”



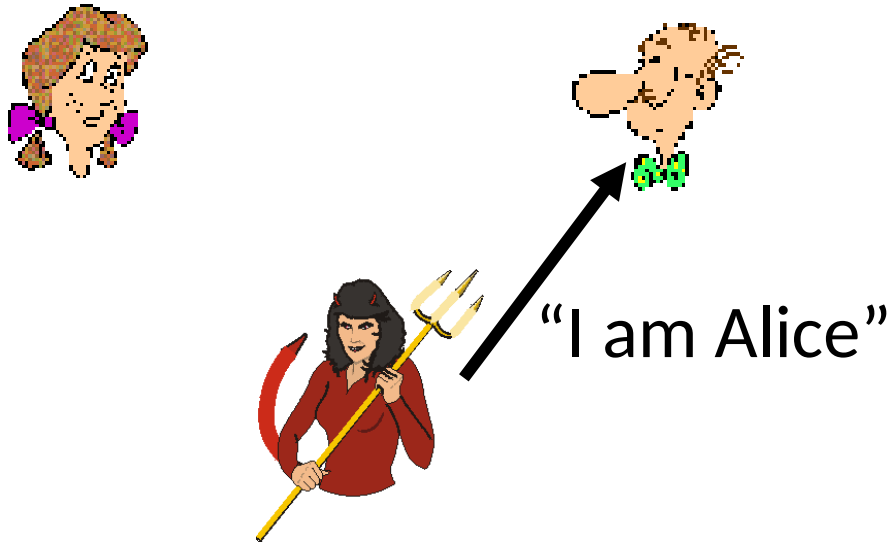
failure scenario??



Authentication

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

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

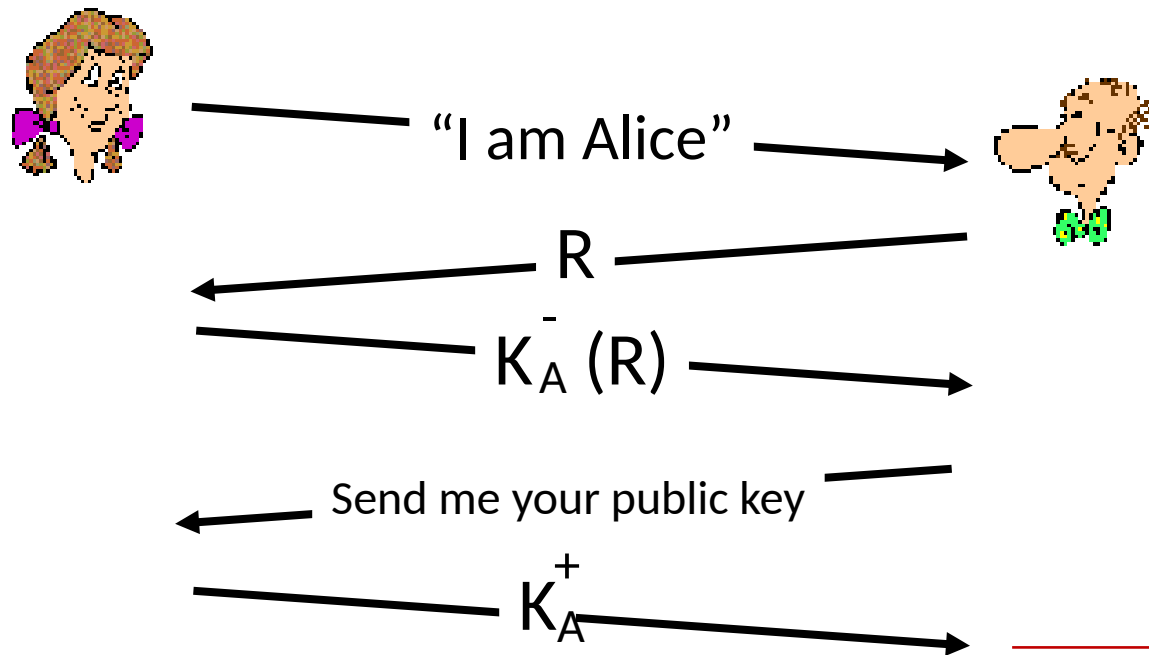


*in a network, Bob
can not “see”
Alice, so Trudy
simply declares
herself to be Alice*



Authentication: ap5.0

ap5.0: use nonce + public key cryptography



Bob computes

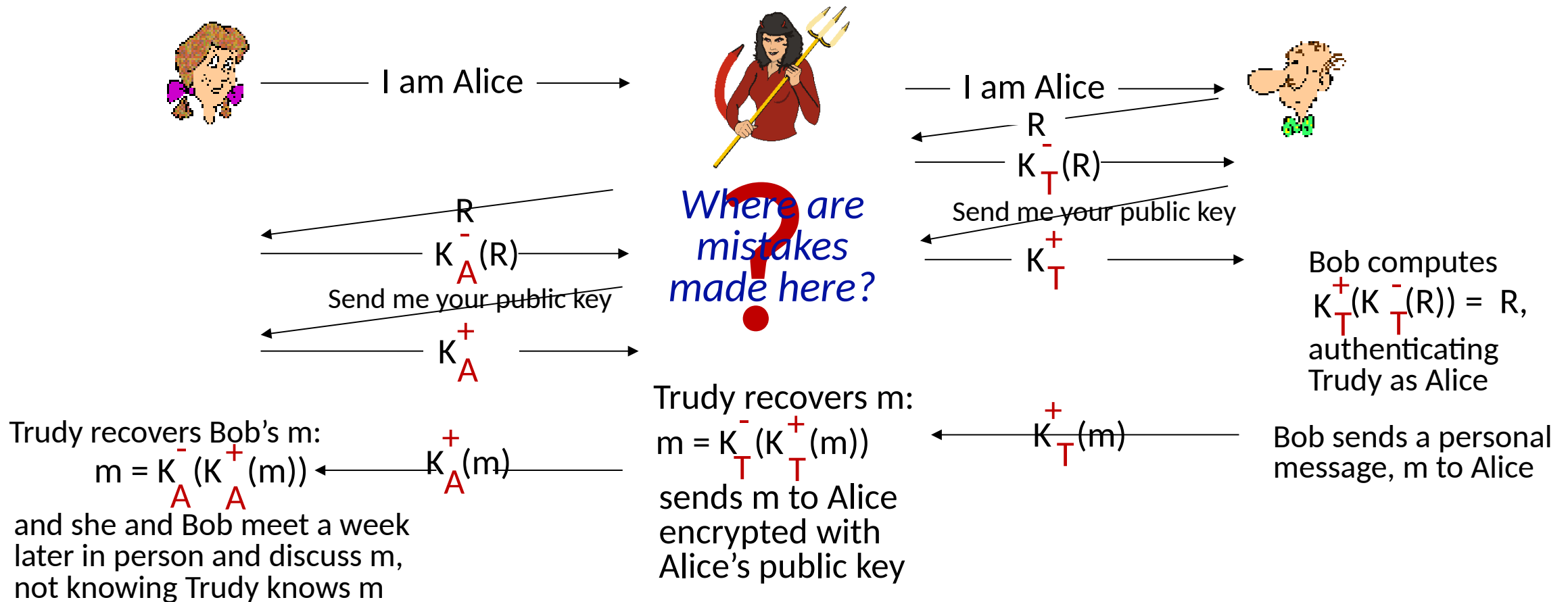
$$K_A^+ (K_A^-(R)) = R$$

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)



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

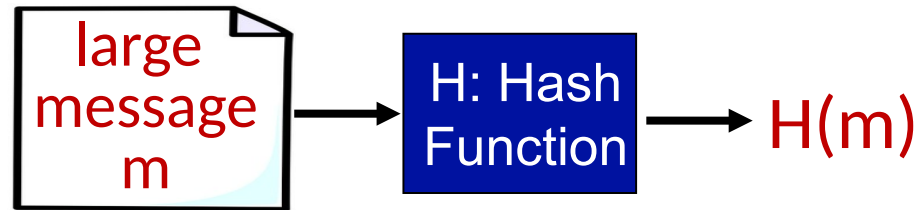


Message digests

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, $H(m)$

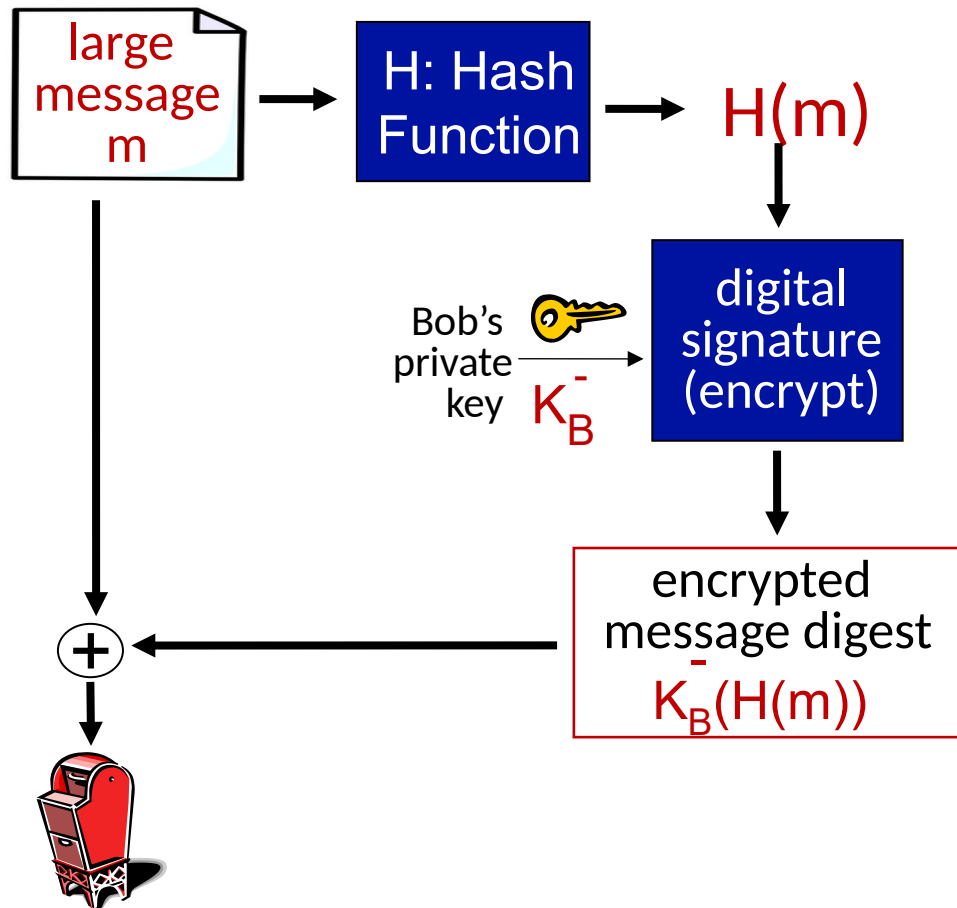


Hash function properties:

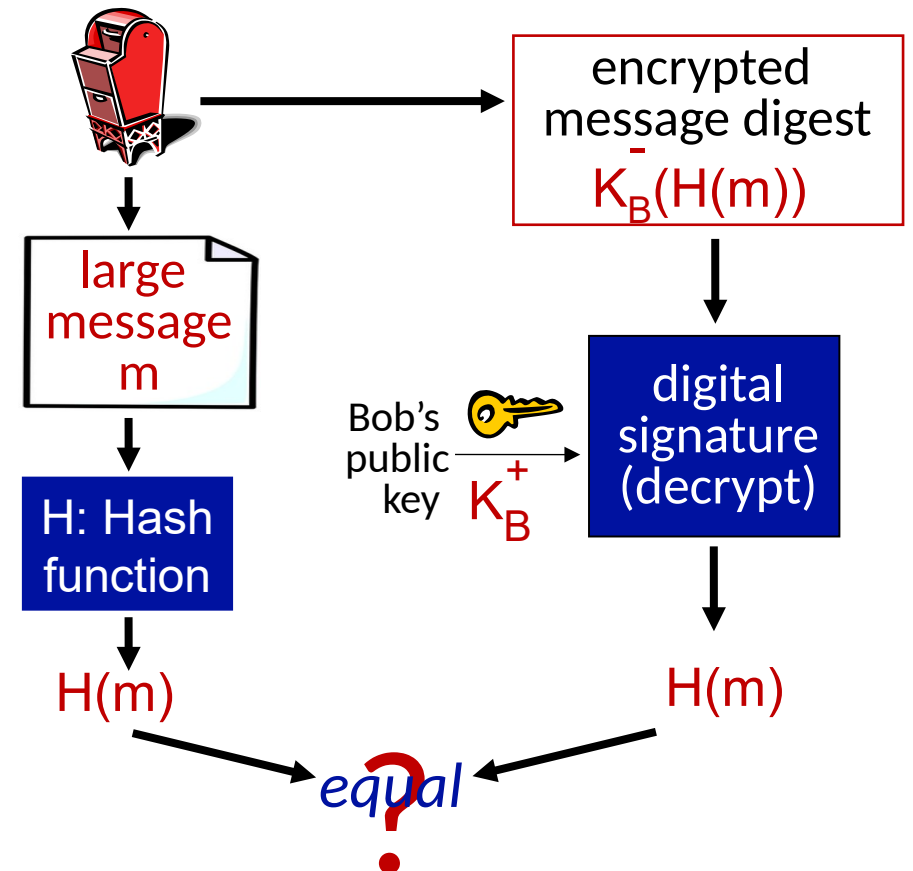
- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x , computationally infeasible to find m such that $x = H(m)$

Digital signature = signed message digest

Bob sends digitally signed message:

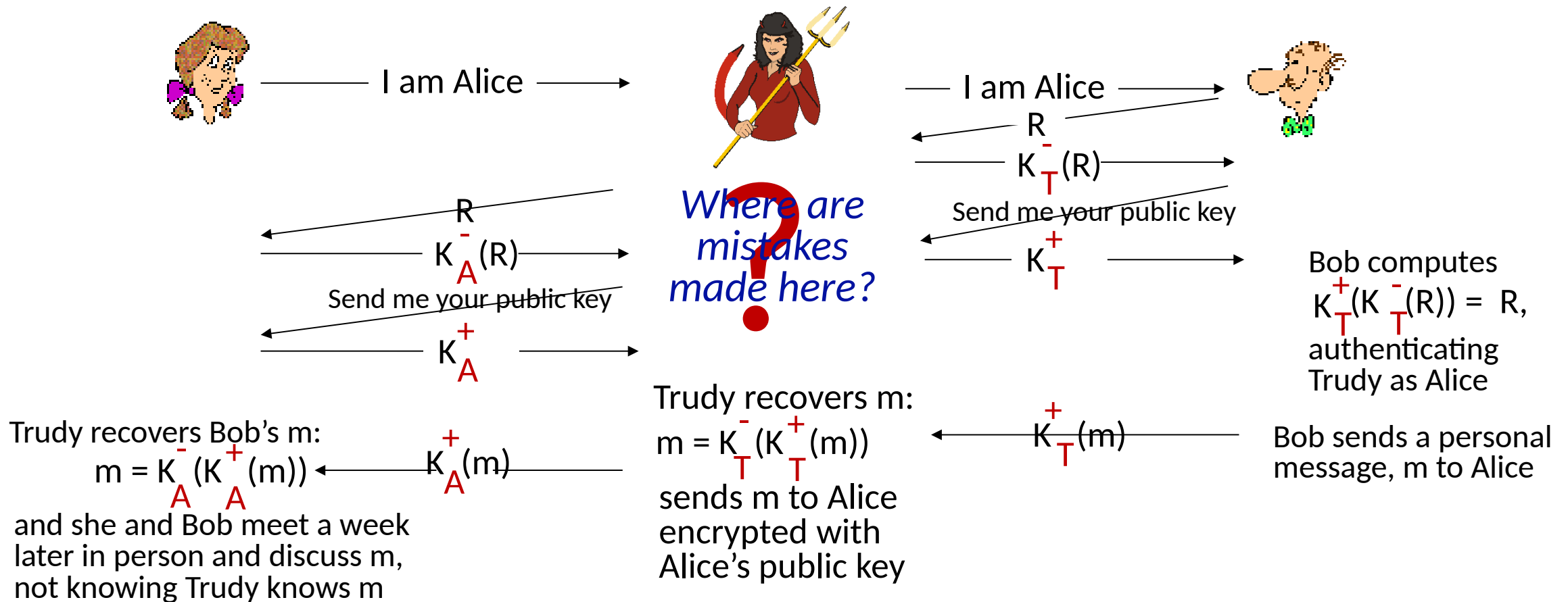


Alice verifies signature, integrity of digitally signed message:



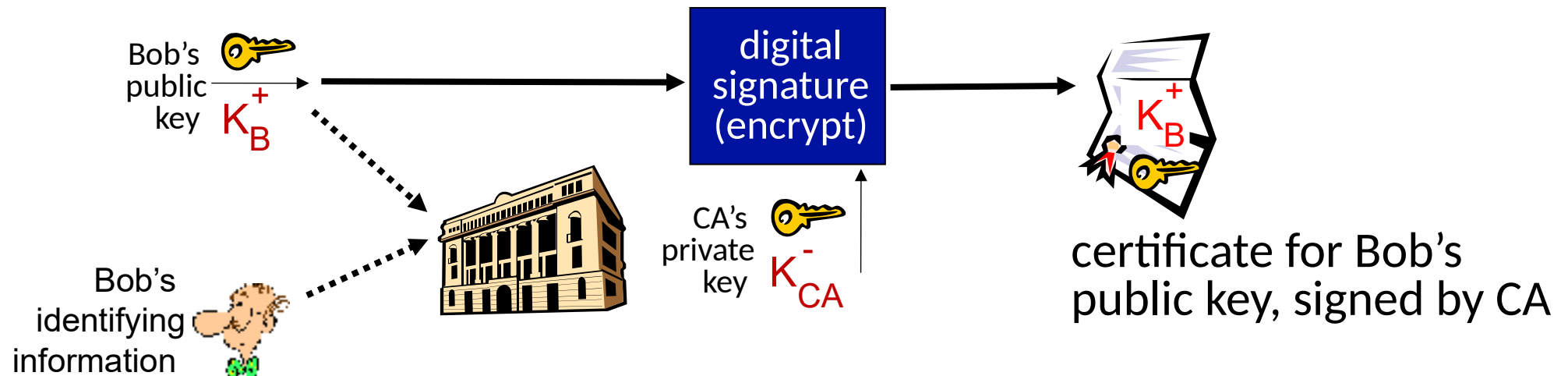
Authentication: ap5.0 – let's fix it!!

Recall the problem: Trudy poses as Alice (to Bob) and as Bob (to Alice)



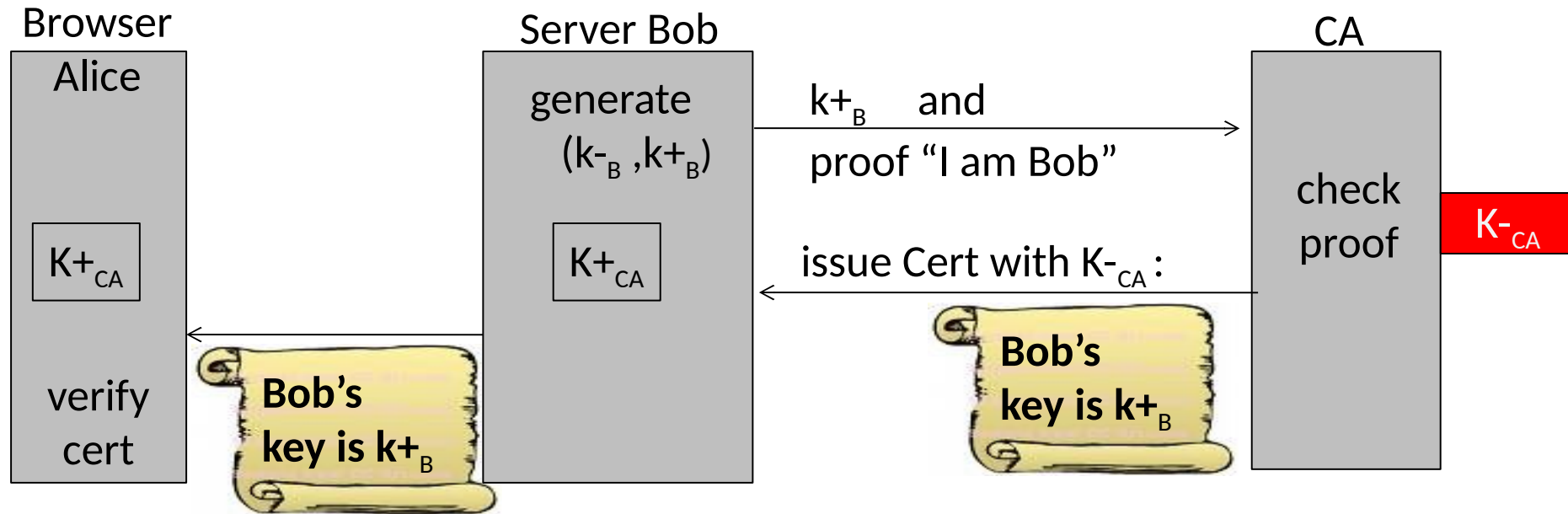
Public key Certification Authorities (CA)

- **Certification authority (CA):** binds public key to particular entity, E
 - Entity (person, website, router) registers its public key and provides “proof of identity” to CA
 - CA creates certificate binding identity E to E’s public key
 - Certificate containing E’s public key digitally signed by CA: CA says “this is E’s public key”



Certificates: Binding Bob's ID to his PK

How does Alice (browser) obtain Bob's public key k^+_B ?



Bob uses Cert for an extended period (e.g. one year)

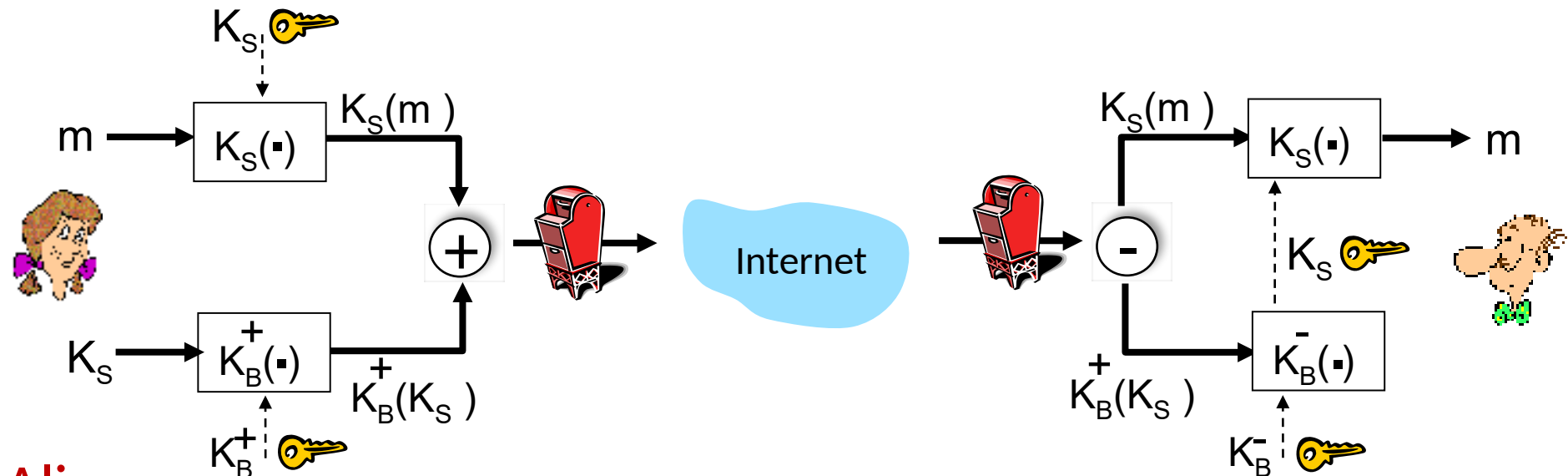
Chapter 8 outline

- 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.

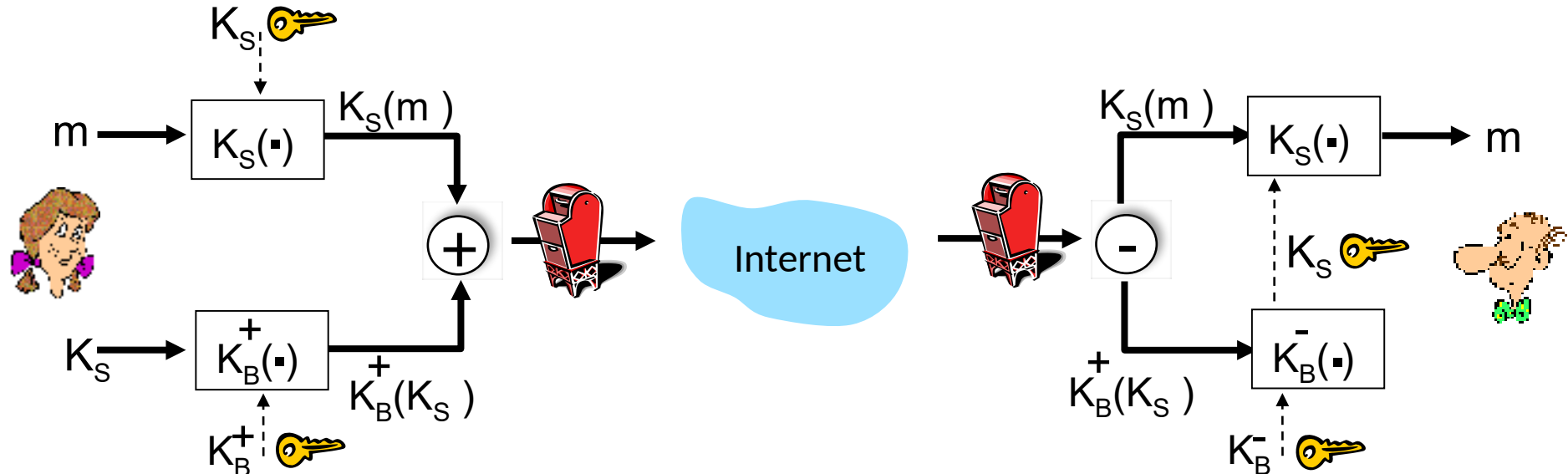


Alice:

- generates random *symmetric* 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: confidentiality (more)

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

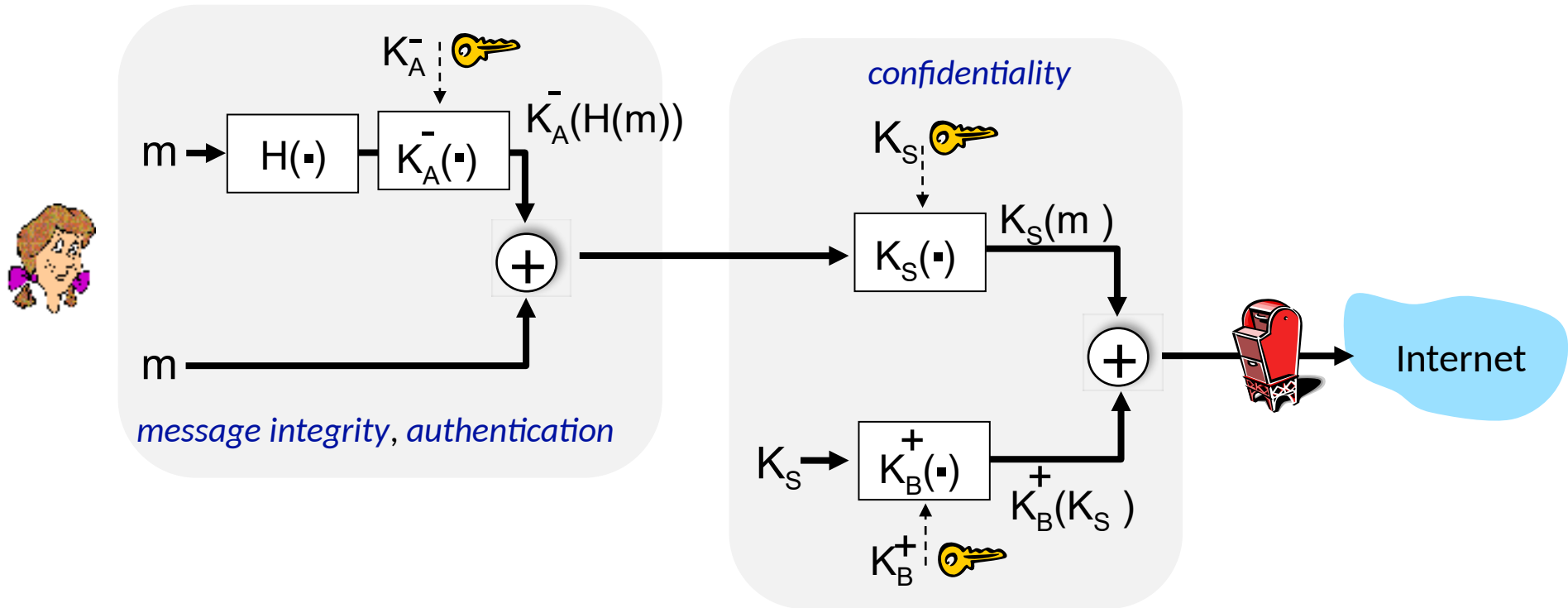


Bob:

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

Secure e-mail: integrity, authentication

Alice sends m to Bob, with *confidentiality*, *message integrity*, *authentication*



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

What are Bob's complementary actions?

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)

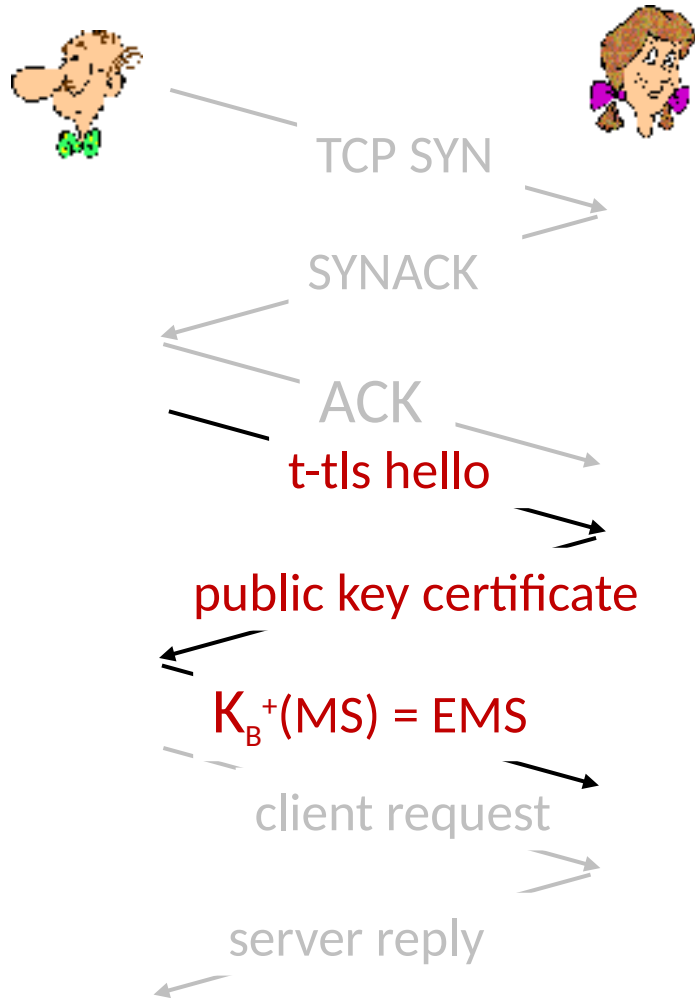
- widely deployed security protocol above the transport layer
 - supported by almost all browsers, web servers: https (port 443)
- provides:
 - **confidentiality**: via *symmetric encryption*
 - **integrity**: via *cryptographic hashing*
 - **authentication**: via *public key cryptography*

} *all techniques we have studied!*
- history:
 - early research, implementation: secure network programming, secure sockets
 - secure socket layer (SSL) deprecated [2015]
 - TLS 1.3: RFC 8846 [2018]

Transport-layer security: what's needed?

- let's *build* a toy TLS protocol, *t-tls*, to see what's needed!
- we've seen the “pieces” already:
 - **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

t-tls: initial handshake



t-tls handshake phase:

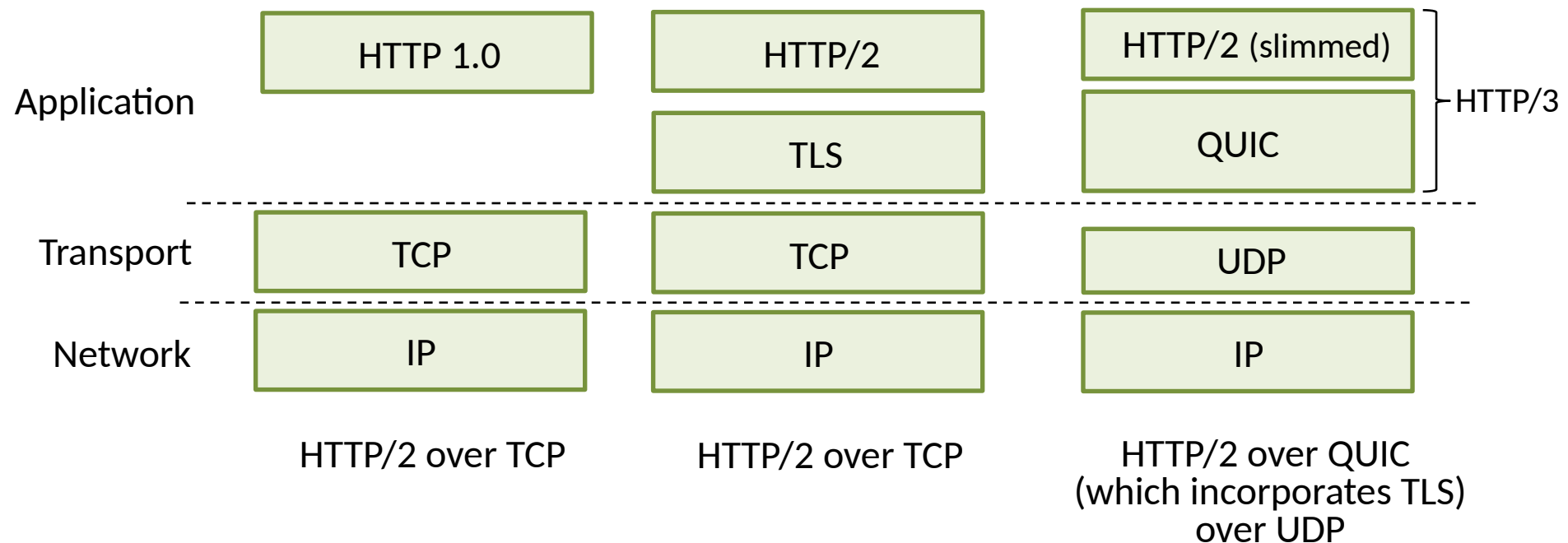
- Bob establishes TCP connection with Alice
- Bob verifies that Alice is really Alice
- Bob sends Alice a master secret key (MS), used to generate all other keys for TLS session
- potential issues:
 - 3 RTT before client can start receiving data (including TCP handshake)

t-tls: cryptographic keys

- considered bad to use same key for more than one cryptographic function
 - 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 to create new keys

Transport-layer security (TLS)

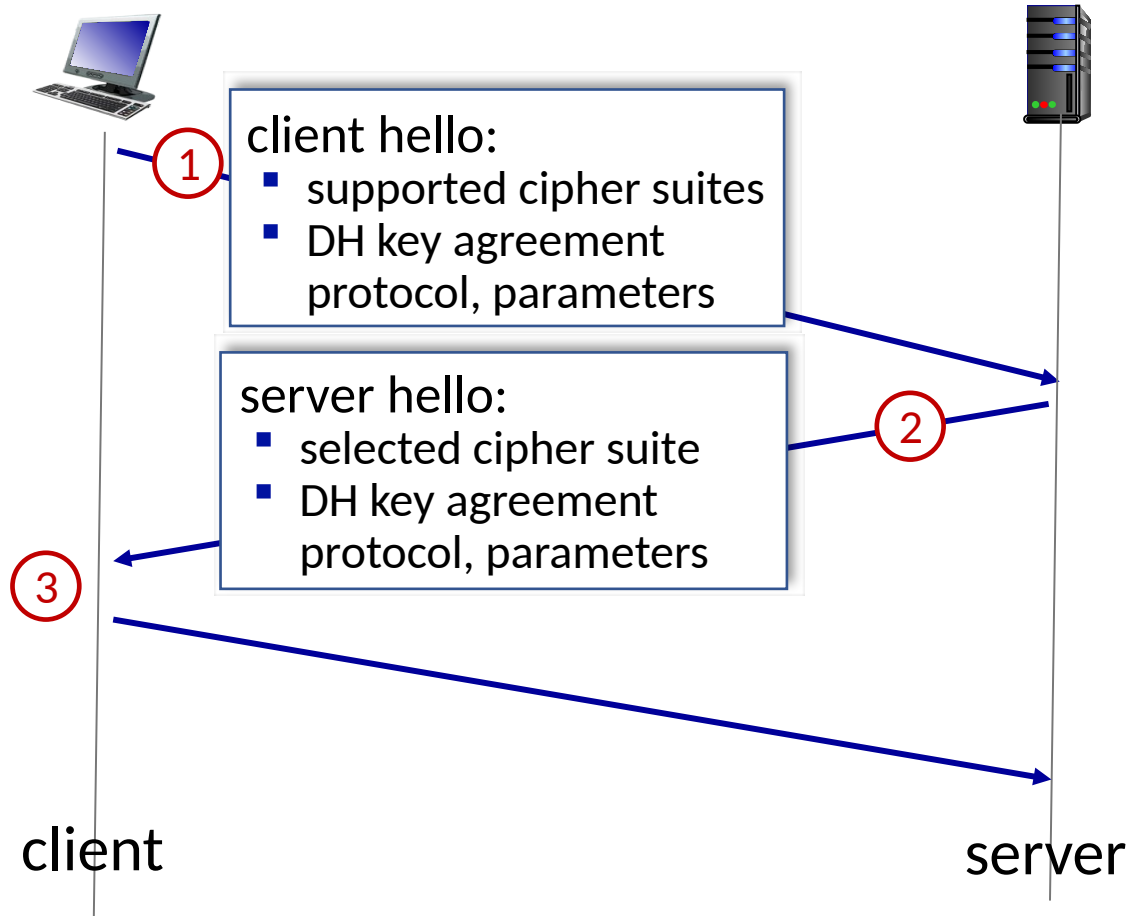
- TLS provides an API that *any* application can use
- an HTTP view of TLS:



TLS: 1.3 cipher suite

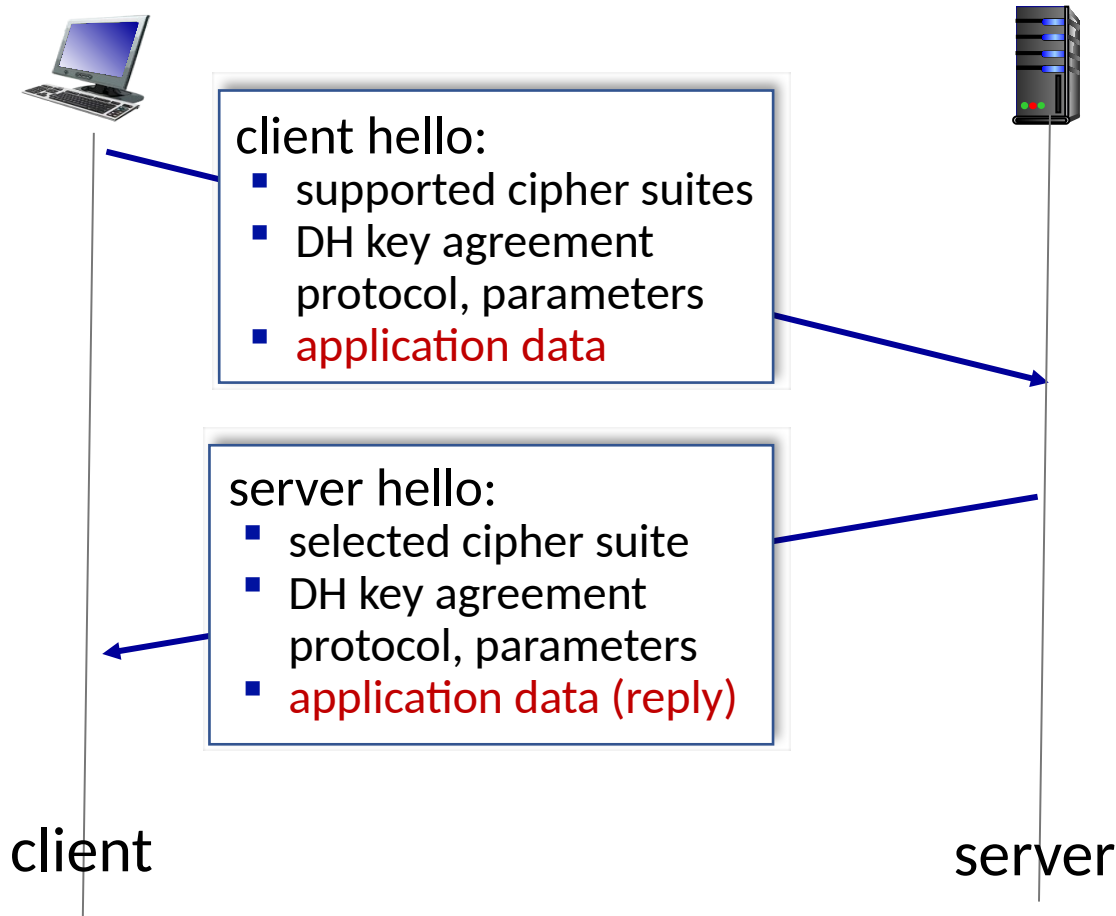
- “cipher suite”: algorithms that can be used for key generation, encryption, MAC, digital signature
- TLS: 1.3 (2018): more limited cipher suite choice than TLS 1.2 (2008)
 - only 5 choices, rather than 37 choices
 - *requires* Diffie-Hellman (DH) for key exchange, rather than DH or RSA
 -
 -

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

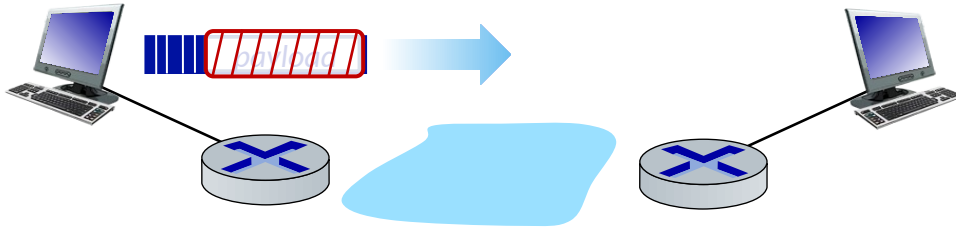
Chapter 8 outline

- 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



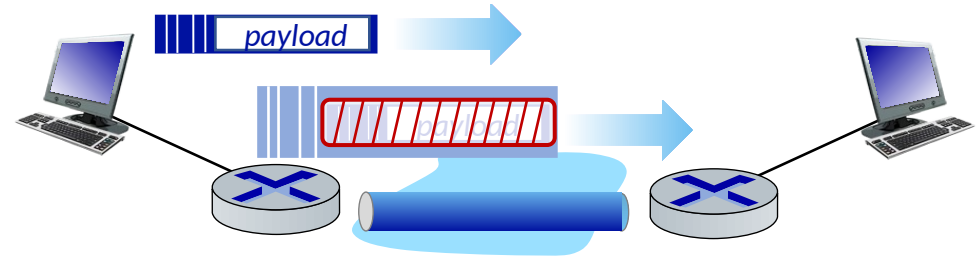
IP Sec

- 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:

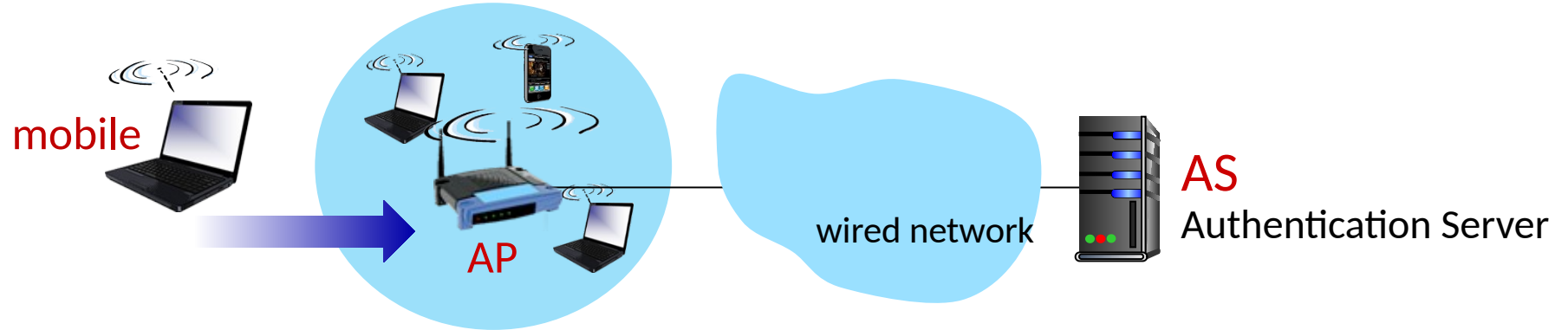
- entire datagram is encrypted, authenticated
- encrypted datagram encapsulated in new datagram with new IP header, tunneled to destination

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



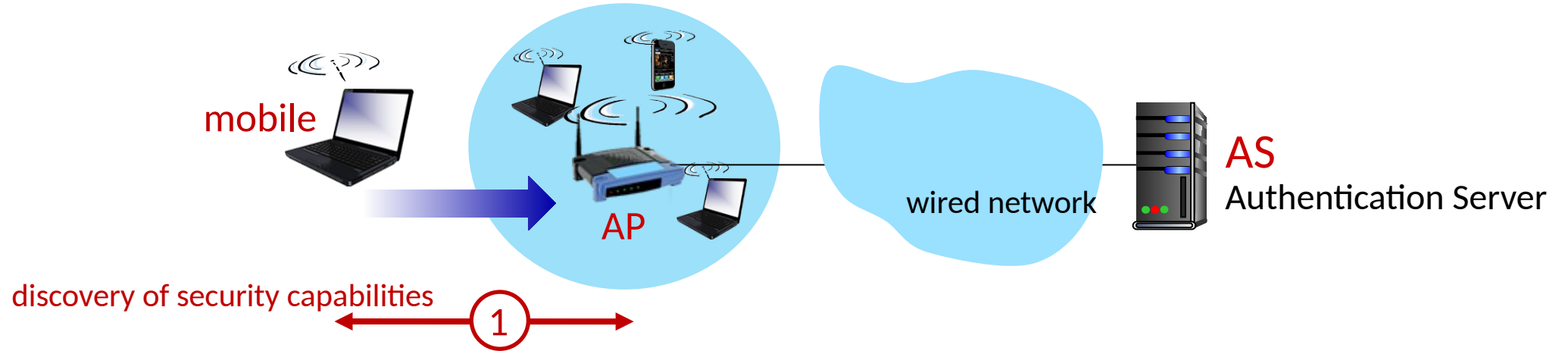
802.11: authentication, encryption



Arriving mobile must:

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

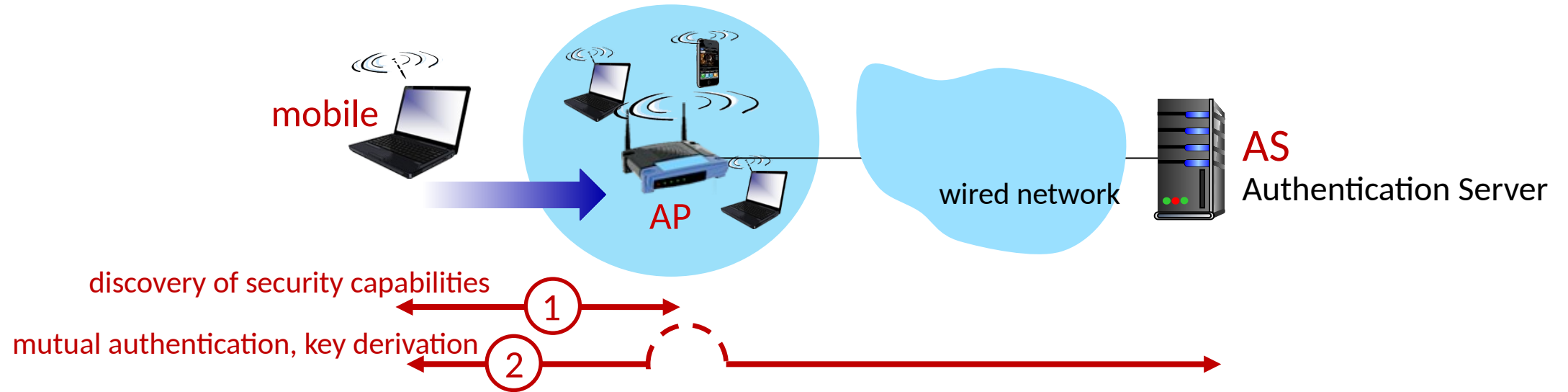
802.11: authentication, encryption



① discovery of security capabilities:

-
-

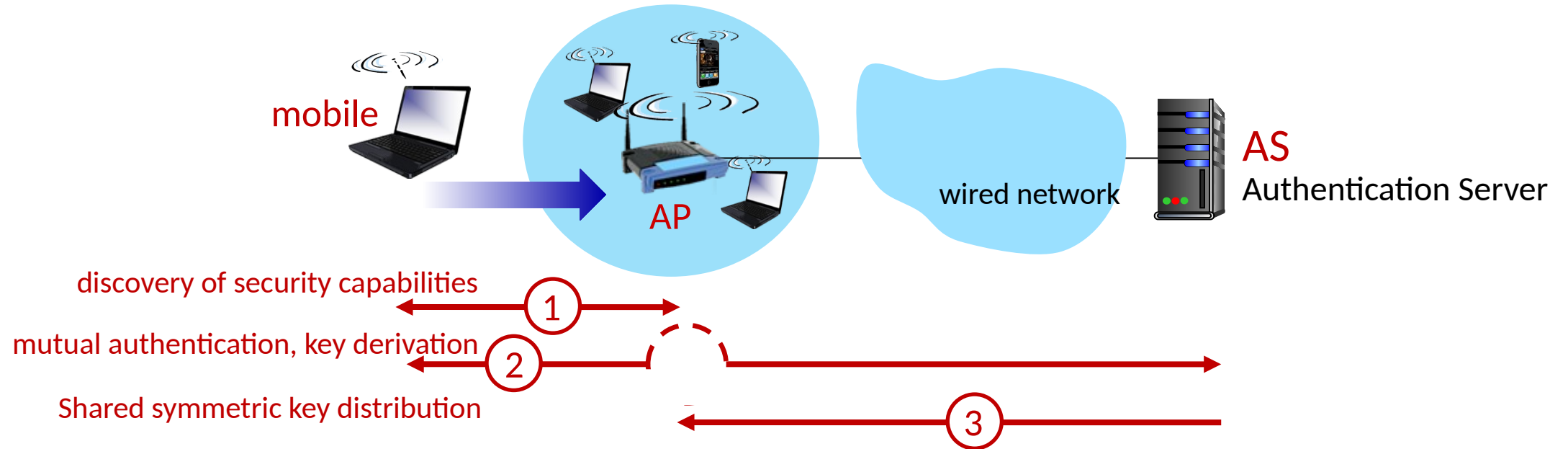
802.11: authentication, encryption



② mutual authentication and shared symmetric key derivation:

-
-
-

802.11: authentication, encryption

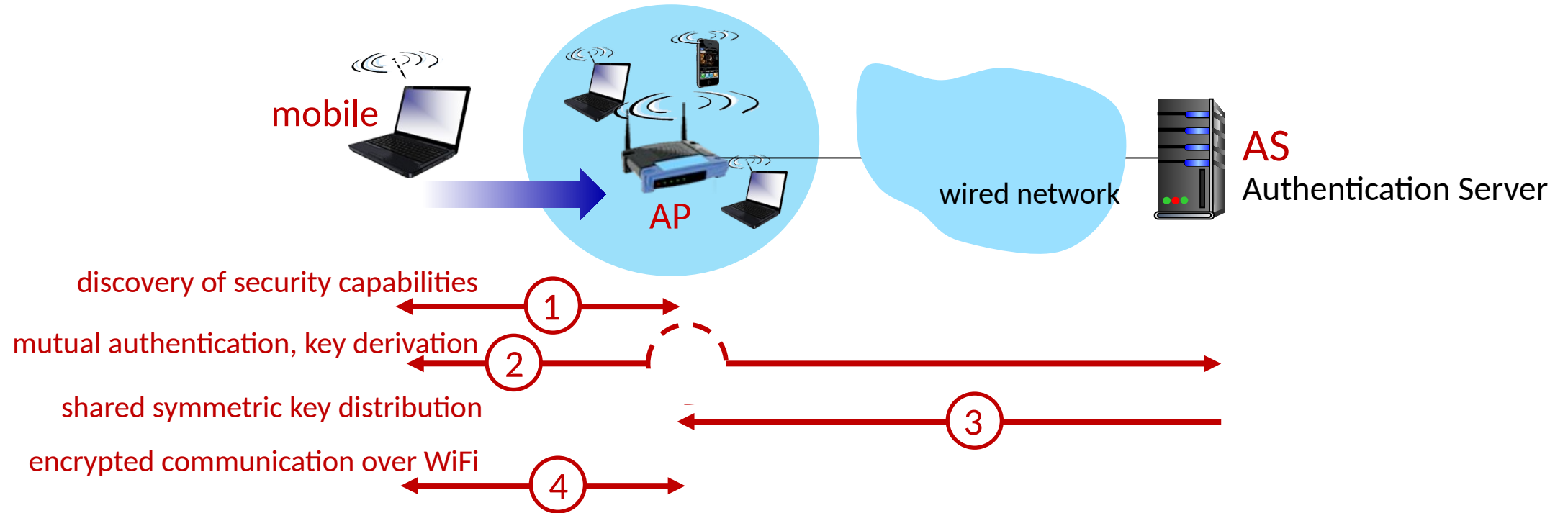


③ shared symmetric session key distribution (e.g., for AES encryption)

■

■

802.11: authentication, encryption



④ encrypted communication between mobile and remote host via AP

-
-

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



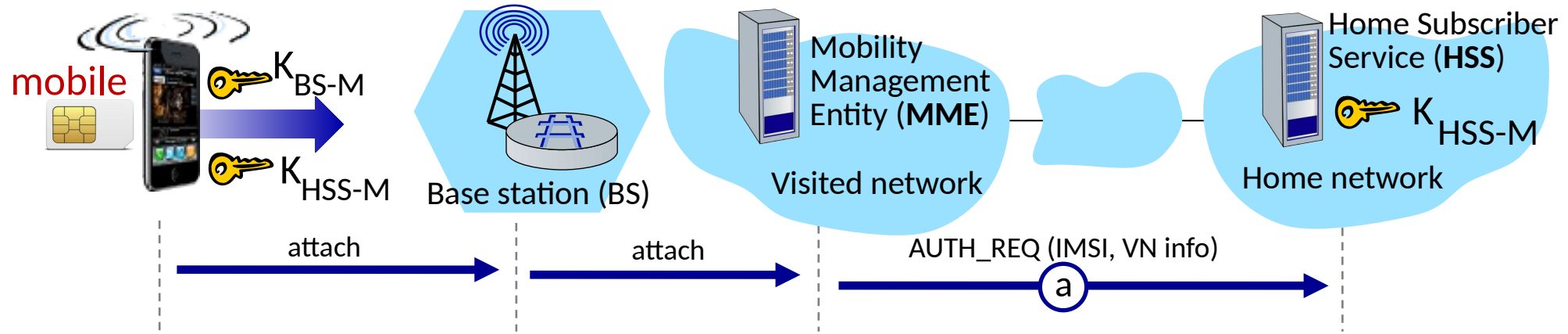
- arriving mobile must:
 - associate with BS: (establish) communication over 4G wireless link
 - authenticate itself to network, and authenticate network
- notable differences from WiFi
 - mobile's SIMcard provides global identity, contains shared keys
 - services in visited network depend on (paid)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

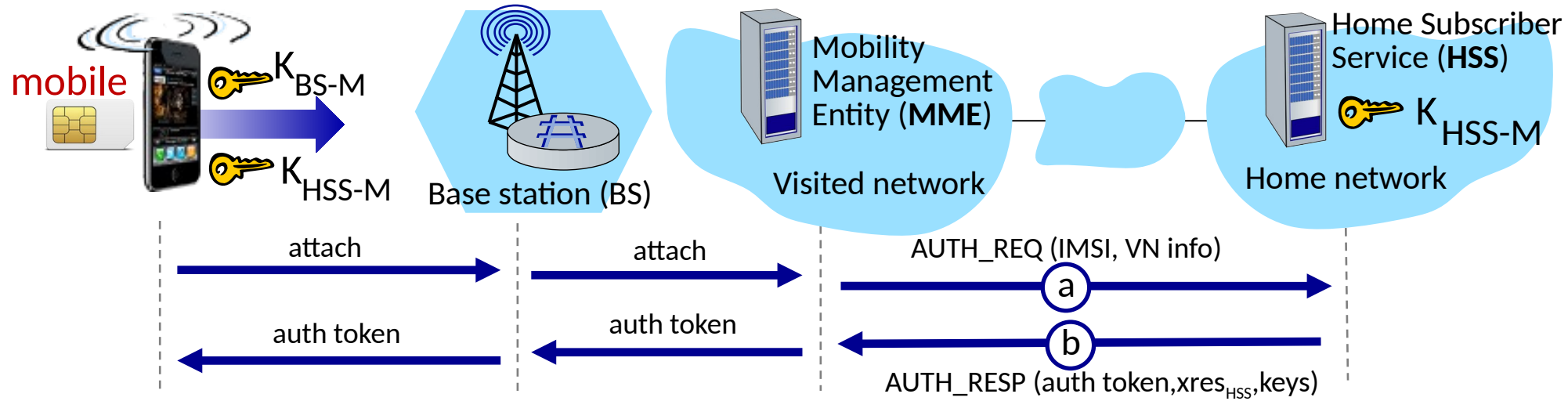
Authentication, encryption in 4G LTE



① authentication request to home network HSS

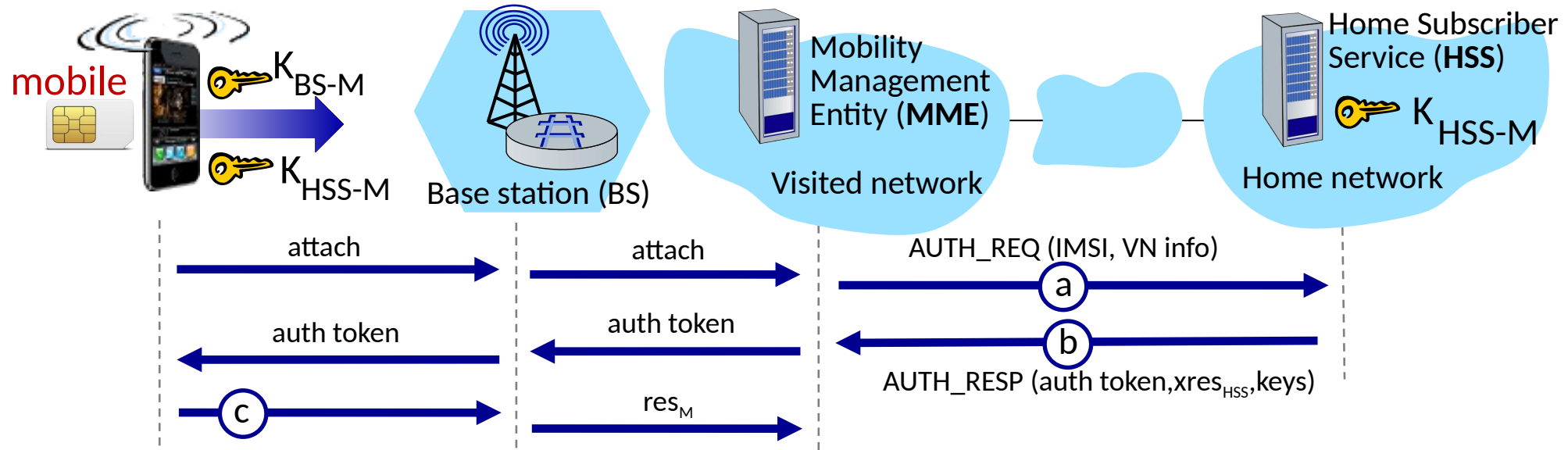
-
-

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, $xres_{HSS}$

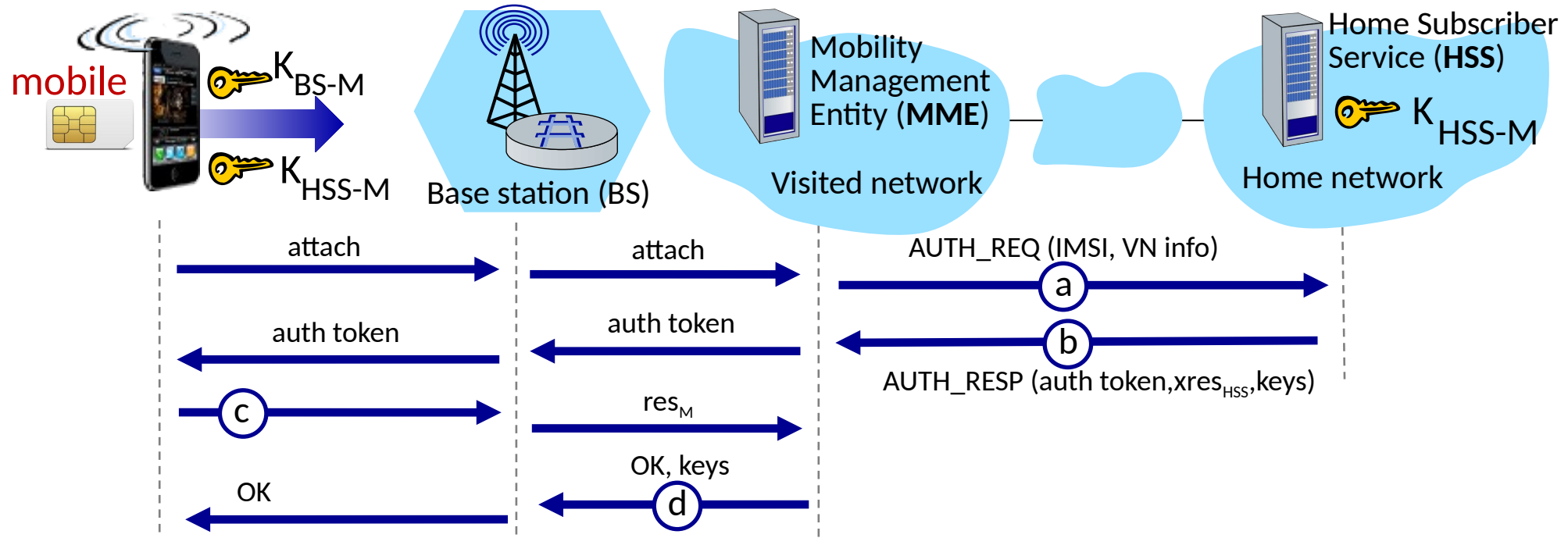
Authentication, encryption in 4G LTE



© authentication response from mobile:

•

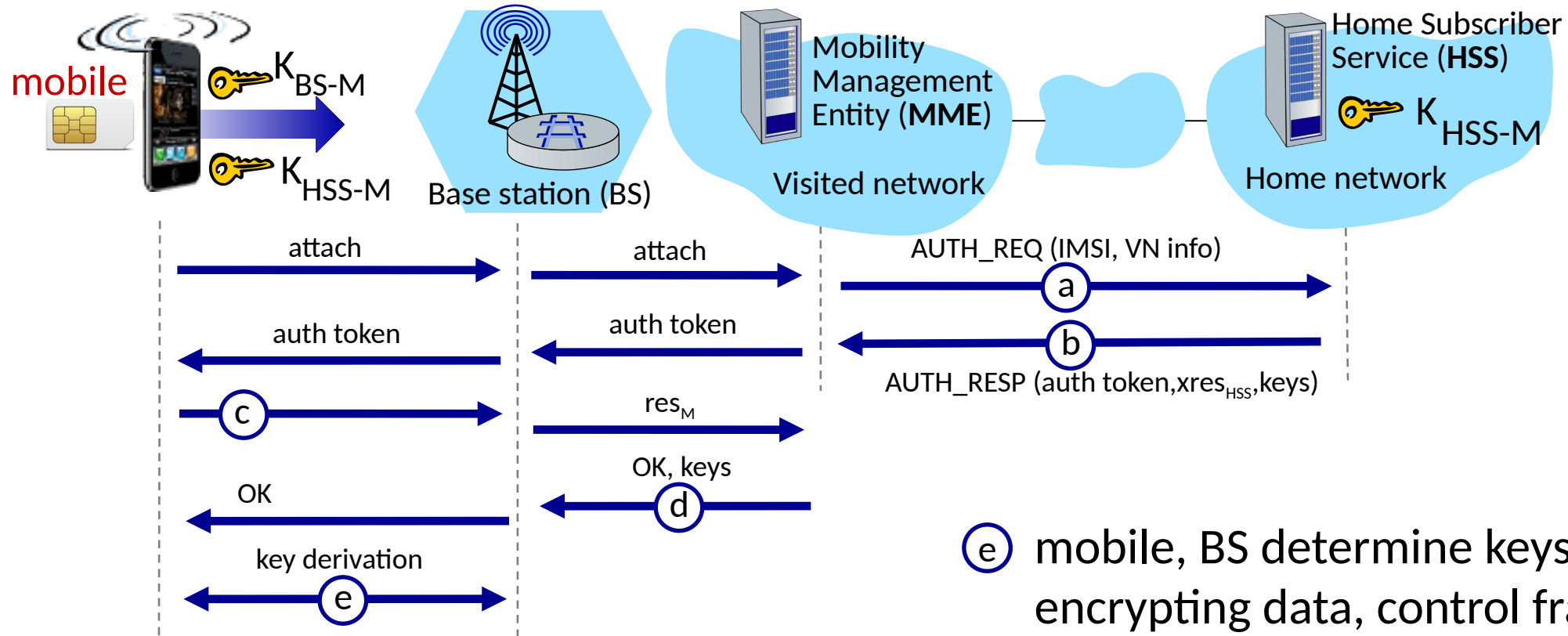
Authentication, encryption in 4G LTE



④ mobile is authenticated by network:

-
-

Authentication, encryption in 4G LTE



- (e) mobile, BS determine keys for encrypting data, control frames over 4G wireless channel

Chapter 8 outline

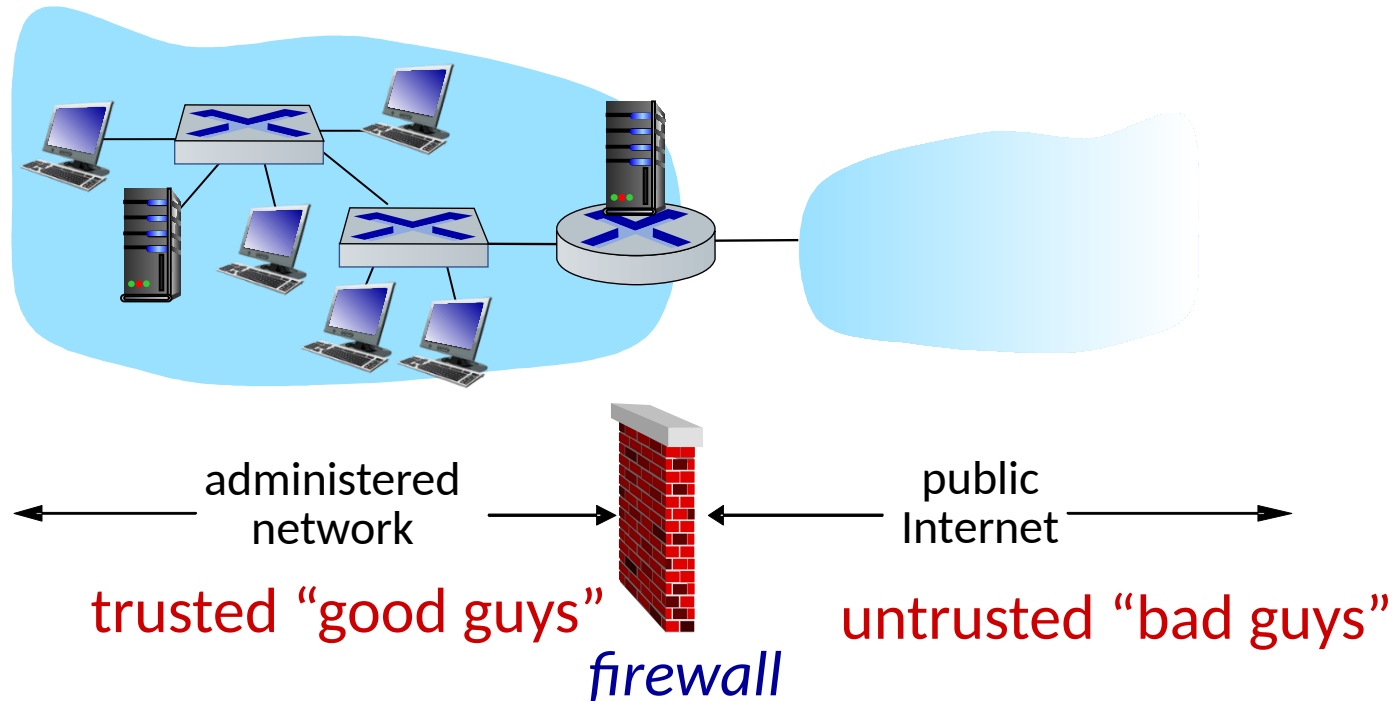
- What is network security?
- Principles of cryptography
- Authentication, message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- **Operational security: firewalls and IDS**



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

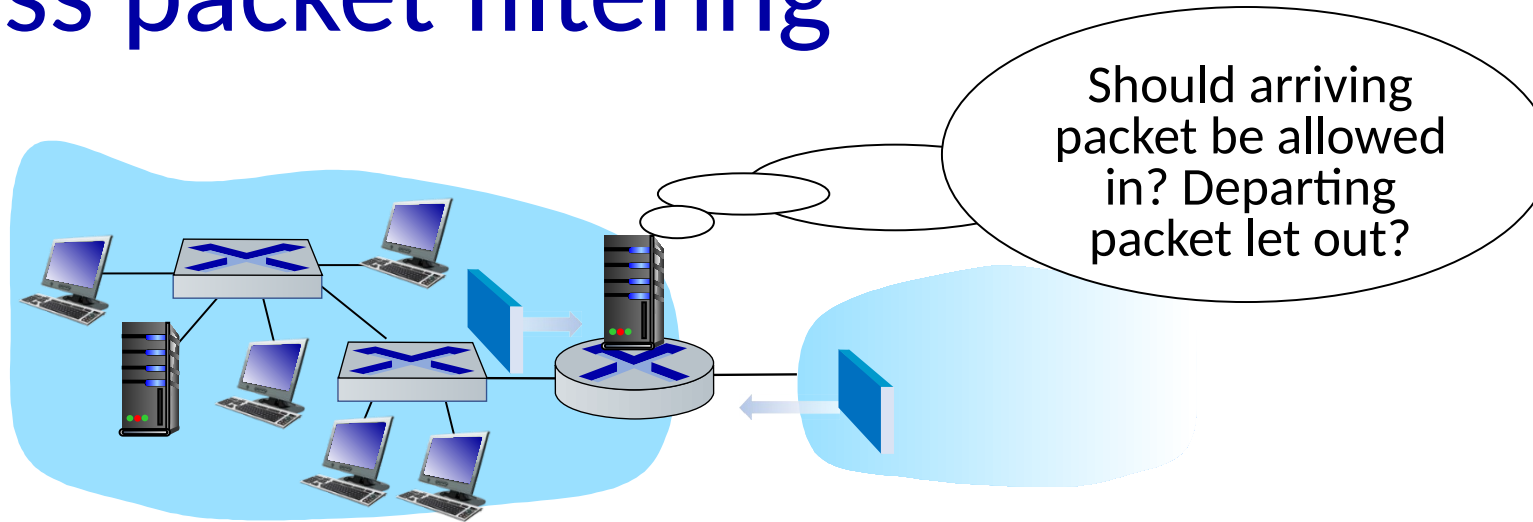
allow only authorized access to inside network

- 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

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

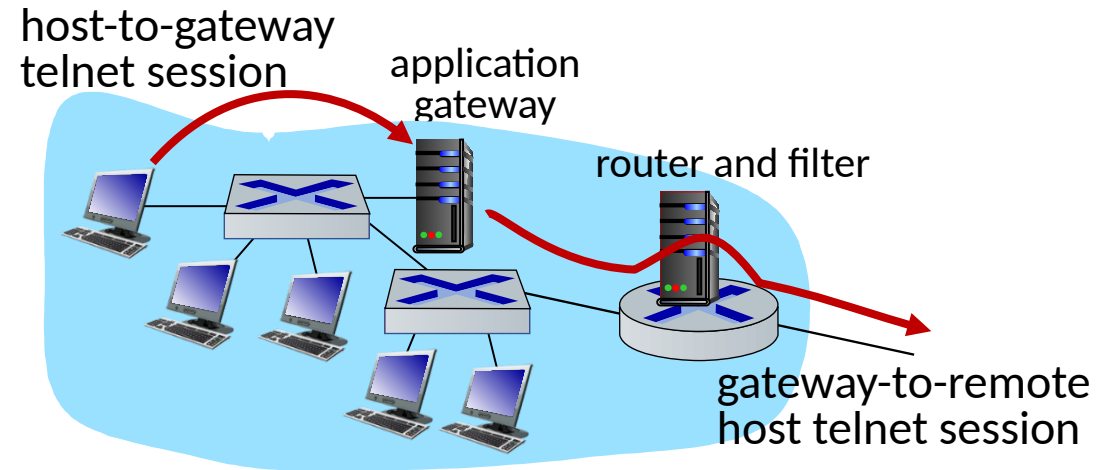
Stateful packet filtering

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

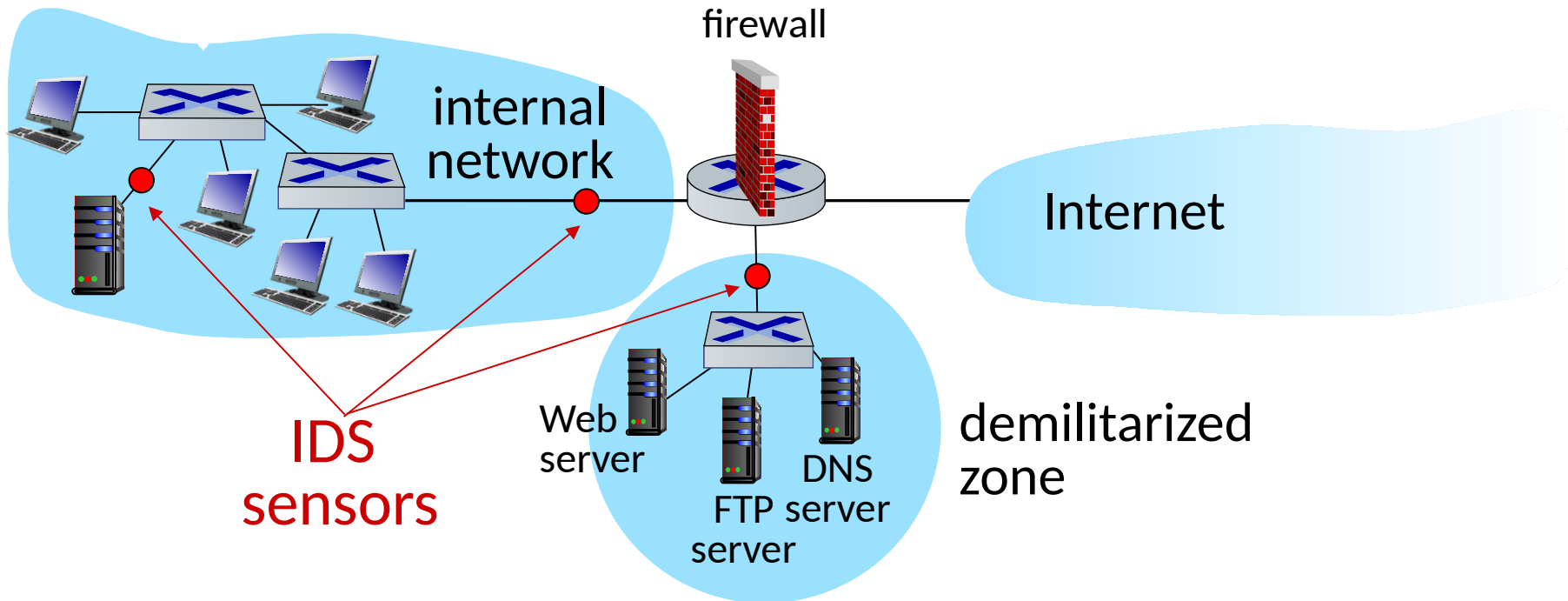
- **IP spoofing:** router can't know if data “really” comes from claimed source
- if multiple apps need special treatment, each has own app. gateway
- client software must know how to contact gateway
 - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
- *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



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

