

Authentication

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

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



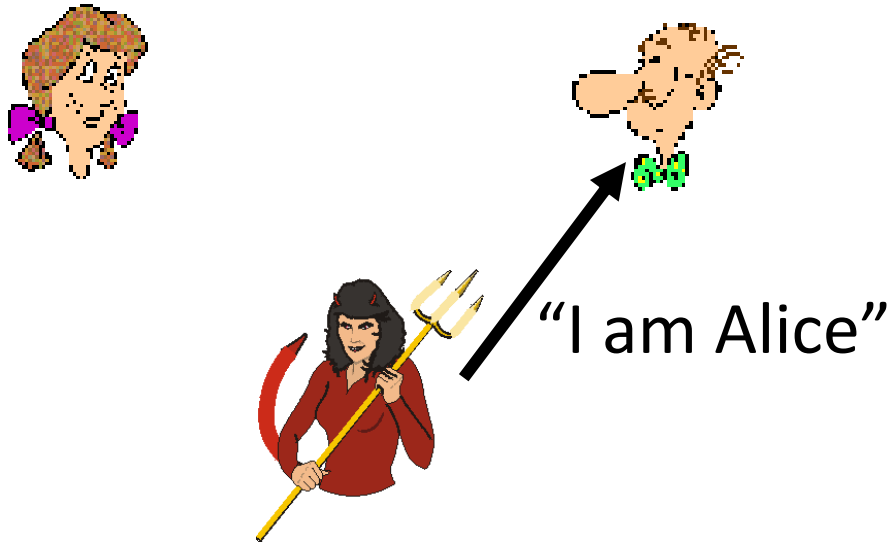
failure scenario??



Authentication

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

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



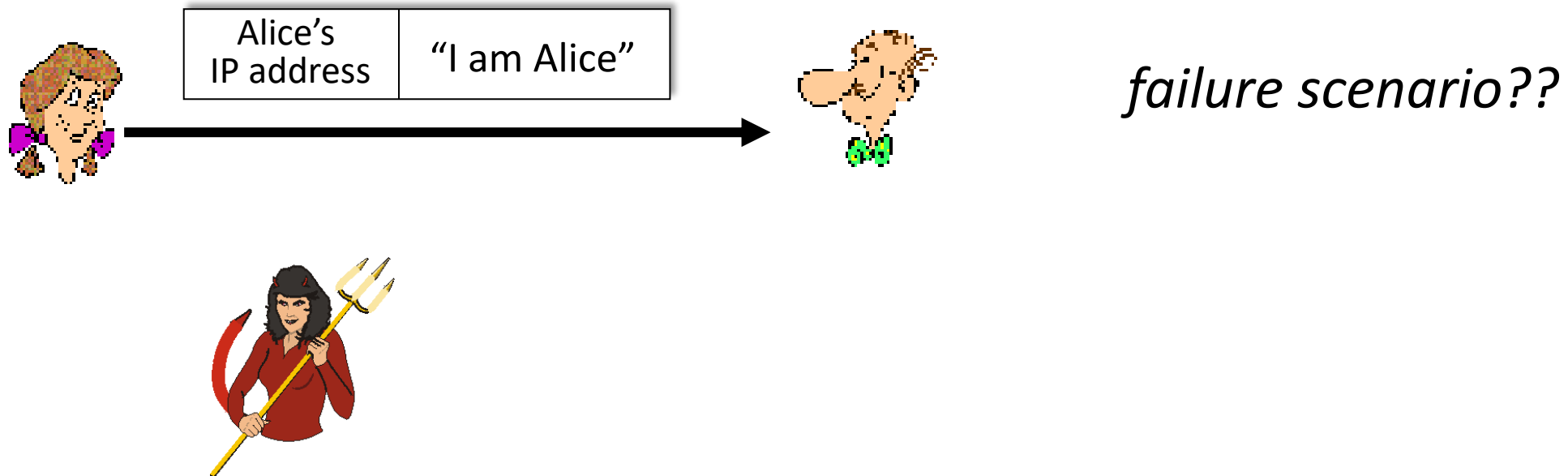
*in a network, Bob
can not “see”
Alice, so Trudy
simply declares
herself to be Alice*



Authentication: another try

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

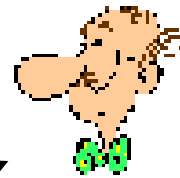
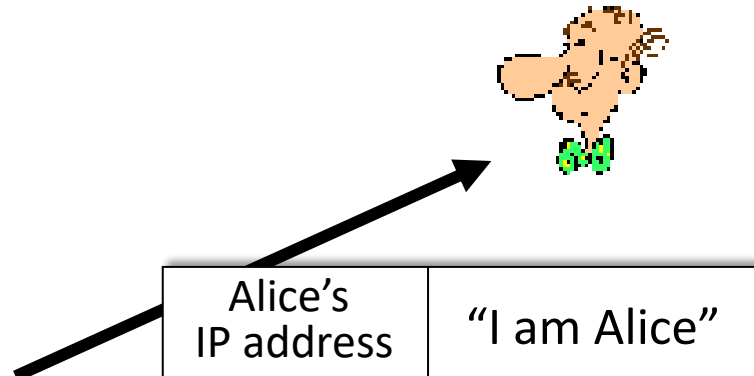
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



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

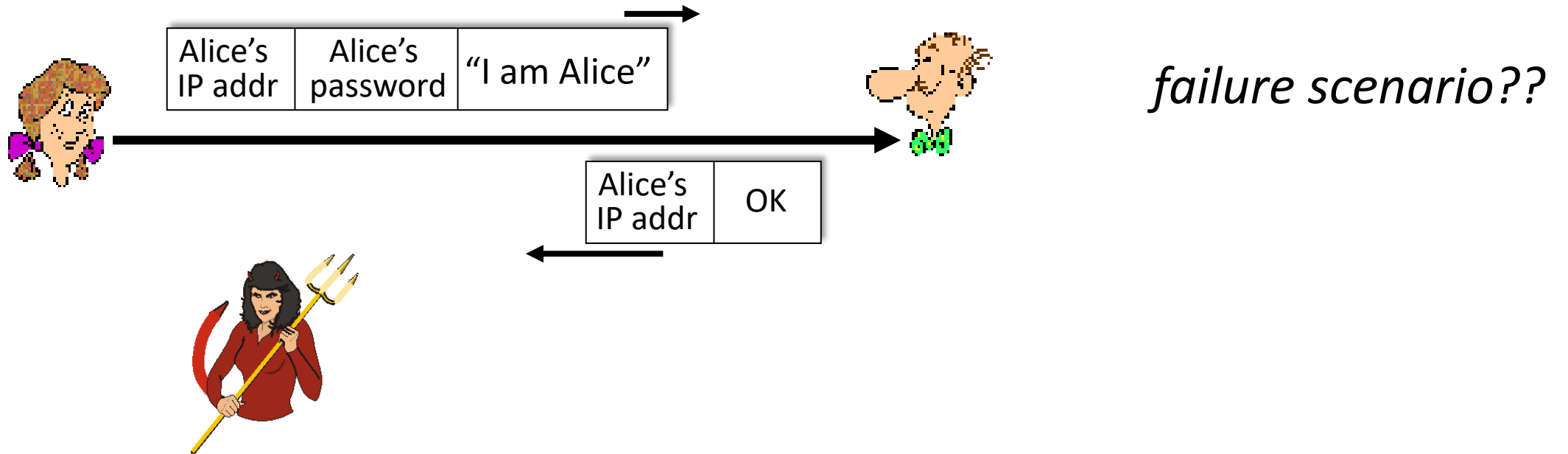


*Trudy can create
a packet “spoofing”
Alice’s address*

Authentication: a third try

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

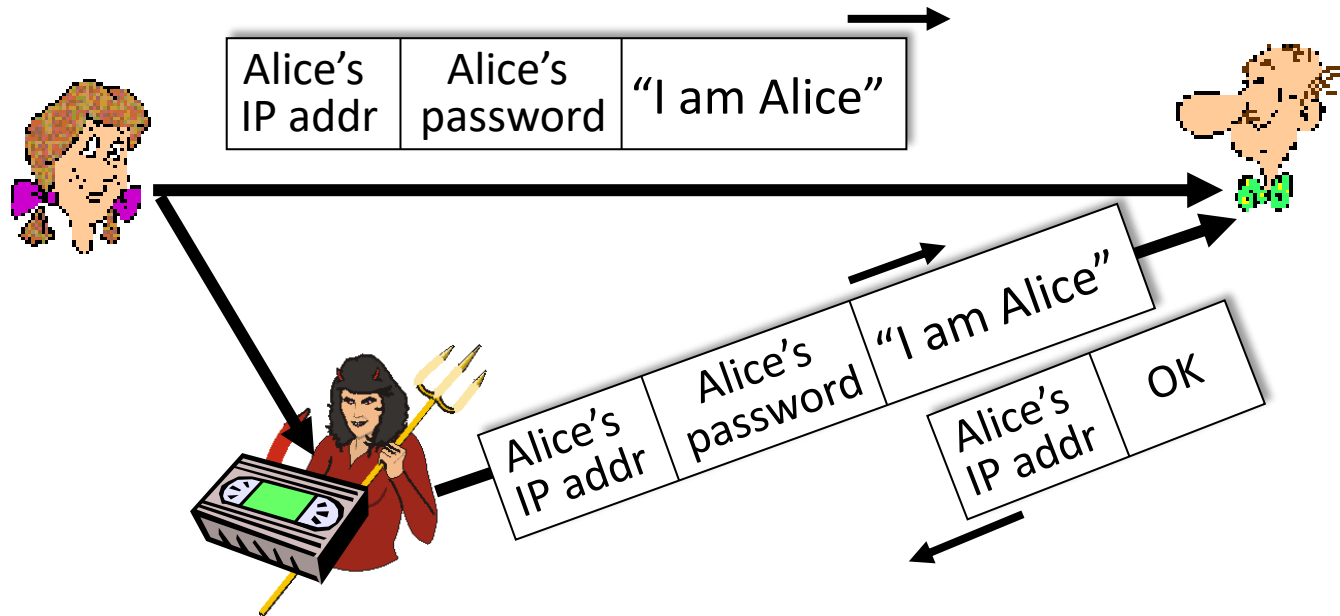
Protocol ap3.0: Alice says “I am Alice” Alice says “I am Alice” and sends her secret password to “prove” it.



Authentication: a third try

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

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

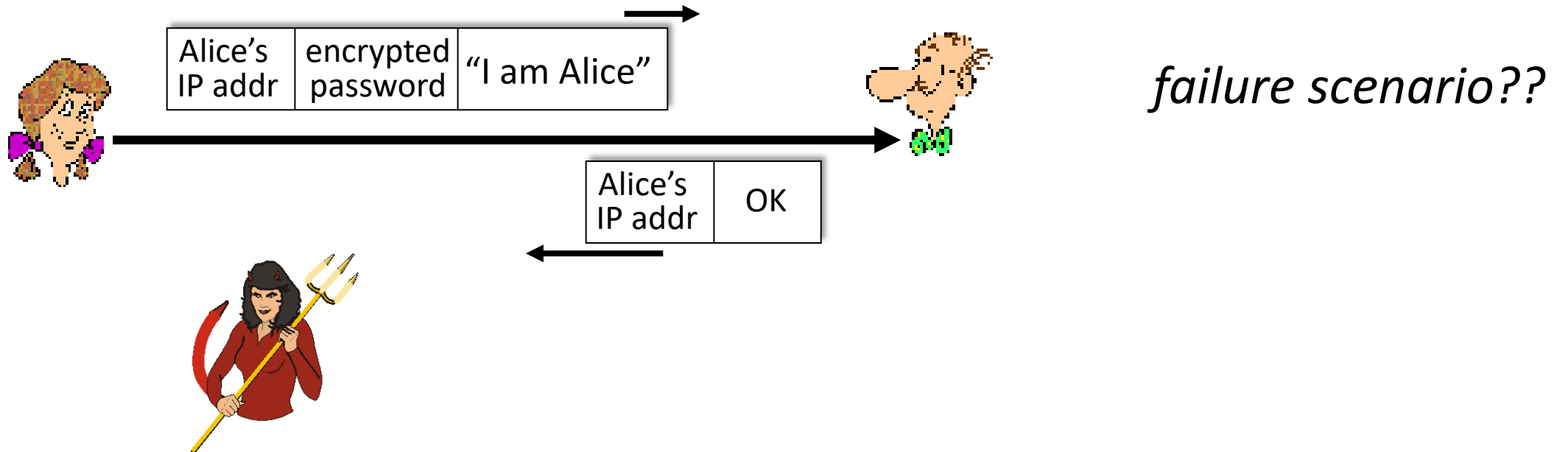


*playback attack:
Trudy records
Alice's packet
and later
plays it back to Bob*

Authentication: a modified third try

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

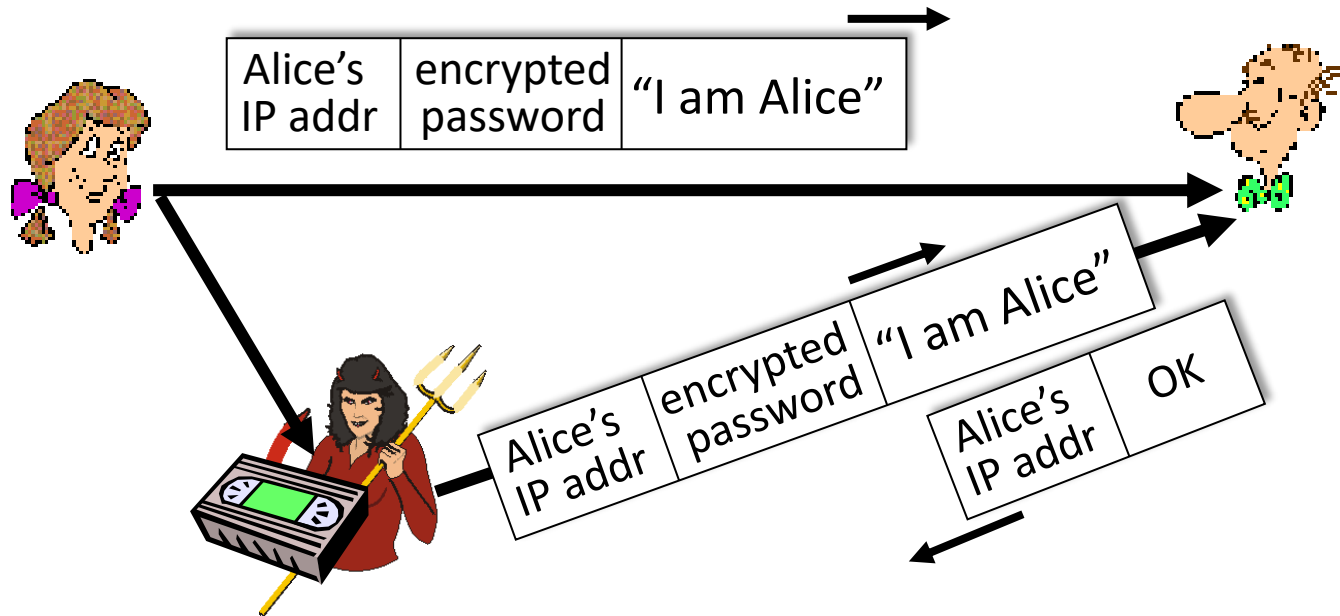
Protocol ap3.0: Alice says “I am Alice” Alice says “I am Alice” and sends her encrypted secret password to “prove” it.



Authentication: a modified third try

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

Protocol ap3.0: Alice says “I am Alice” 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

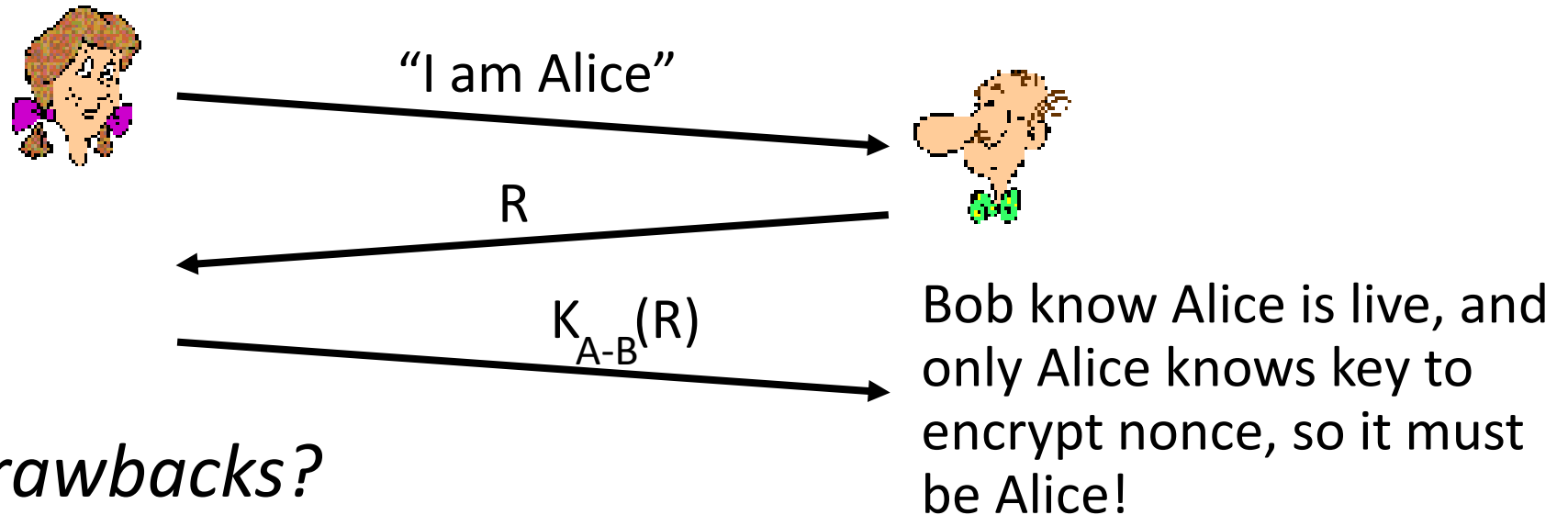
Authentication: a fourth try

Goal: avoid playback attack

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

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

- Alice must return R, encrypted with shared secret key

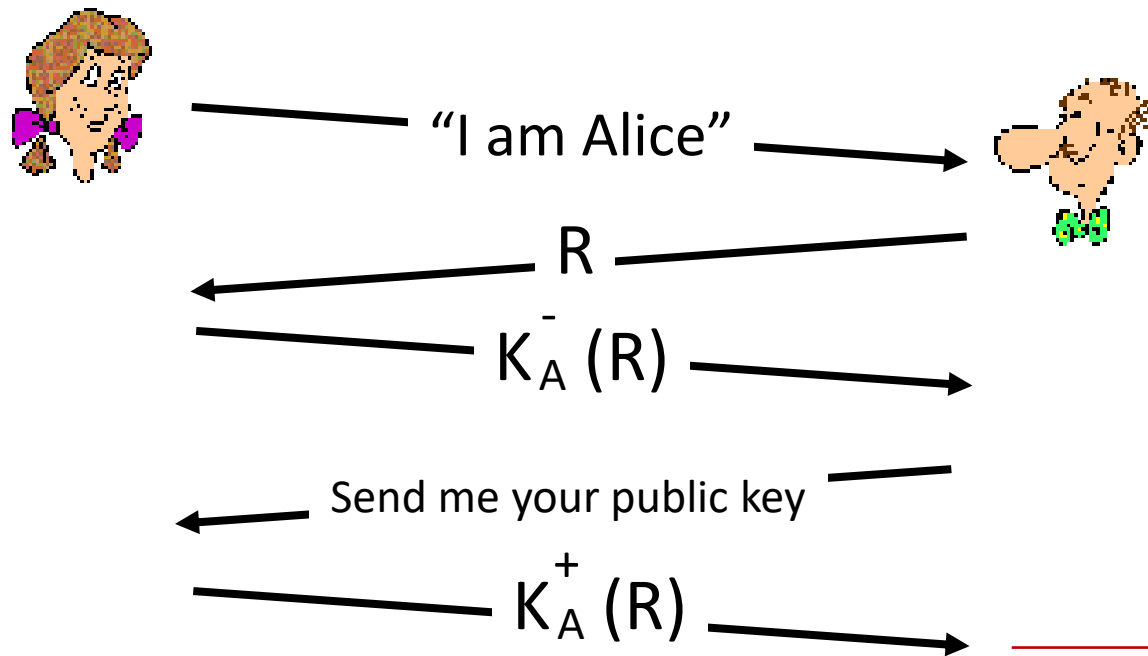


Failures, drawbacks?

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

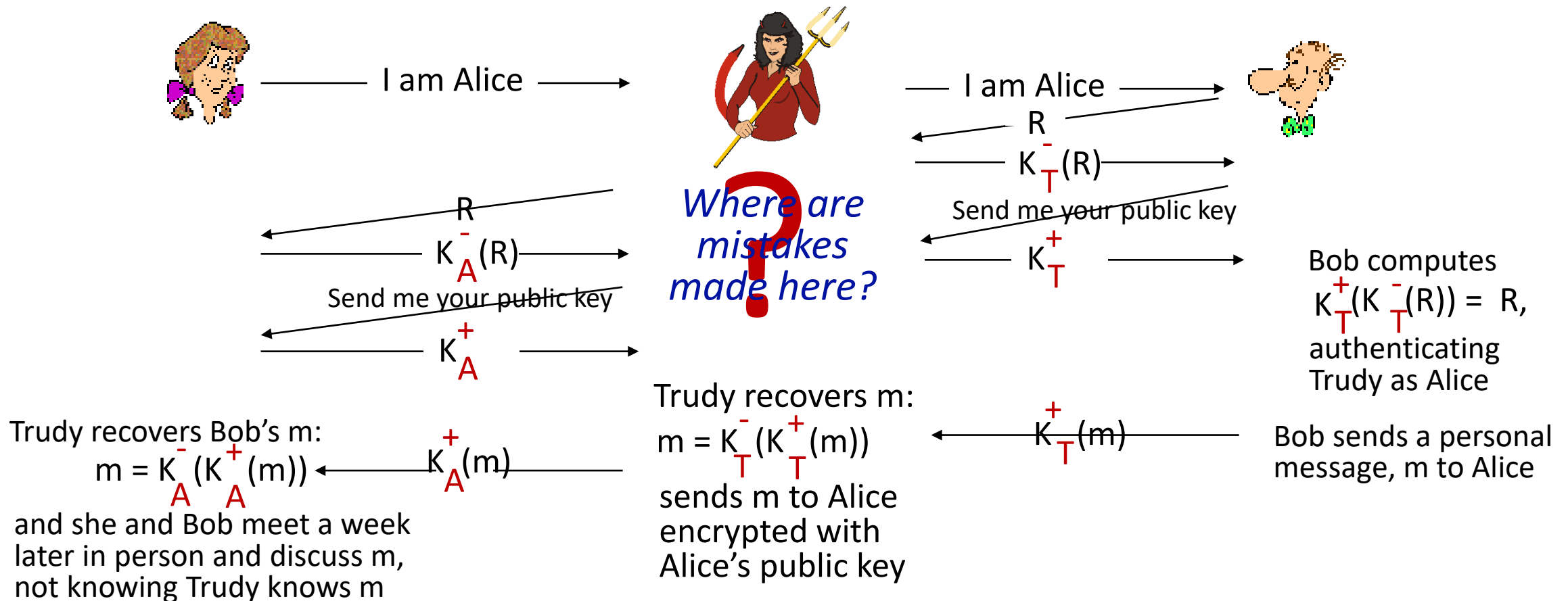
$$K_A^+ (K_A^-(R)) = R$$

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!

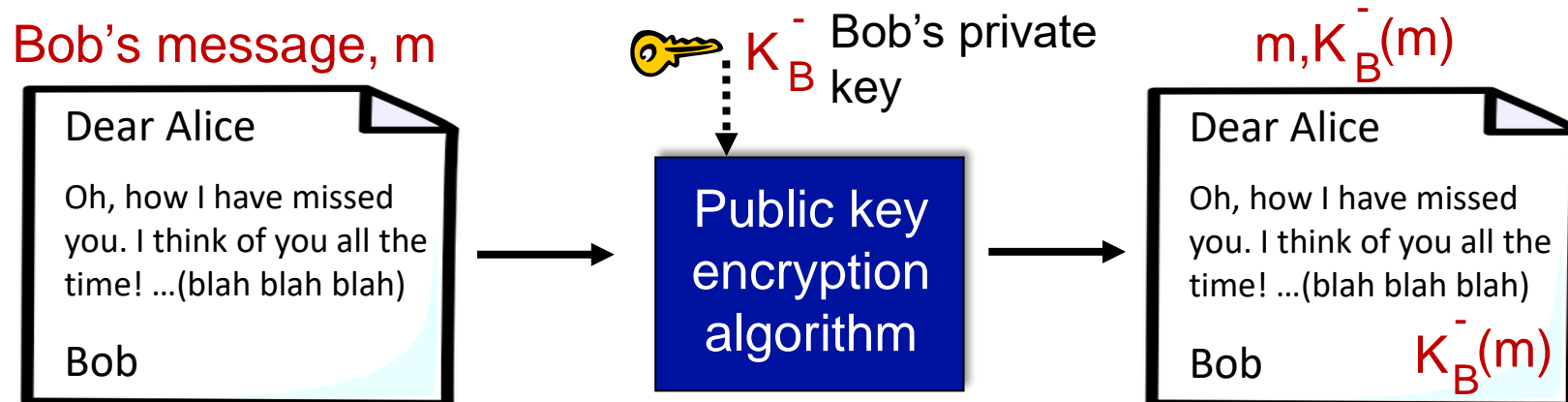
man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



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: $m, \bar{K}_B(m)$
- Alice verifies m signed by Bob by applying Bob's public key \bar{K}_B to $\bar{K}_B(m)$ then checks $\bar{K}_B(\bar{K}_B(m)) = m$.
- If $\bar{K}_B(\bar{K}_B(m)) = m$, 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:

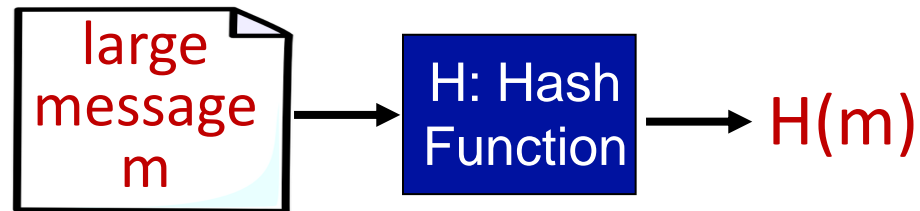
- ✓ Alice can take m , and signature $\bar{K}_B(m)$ to court and prove that Bob signed m

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:

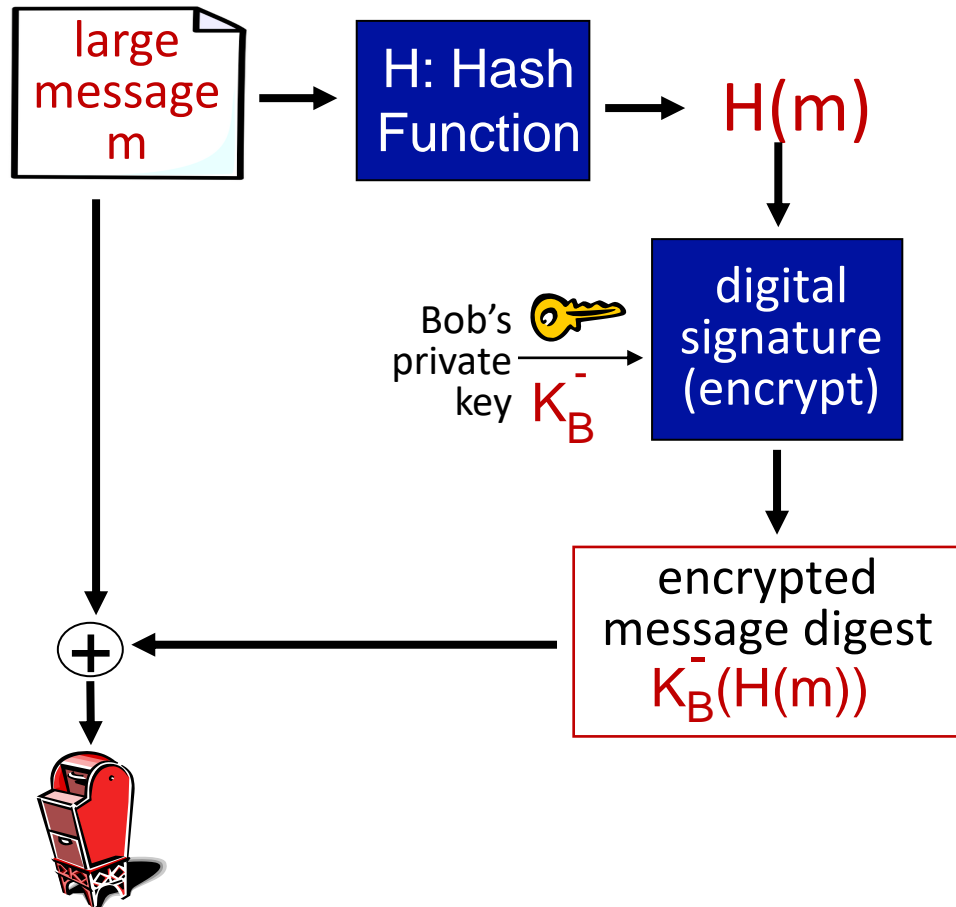
- produces fixed length digest (16-bit sum) of message
- is many-to-one

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

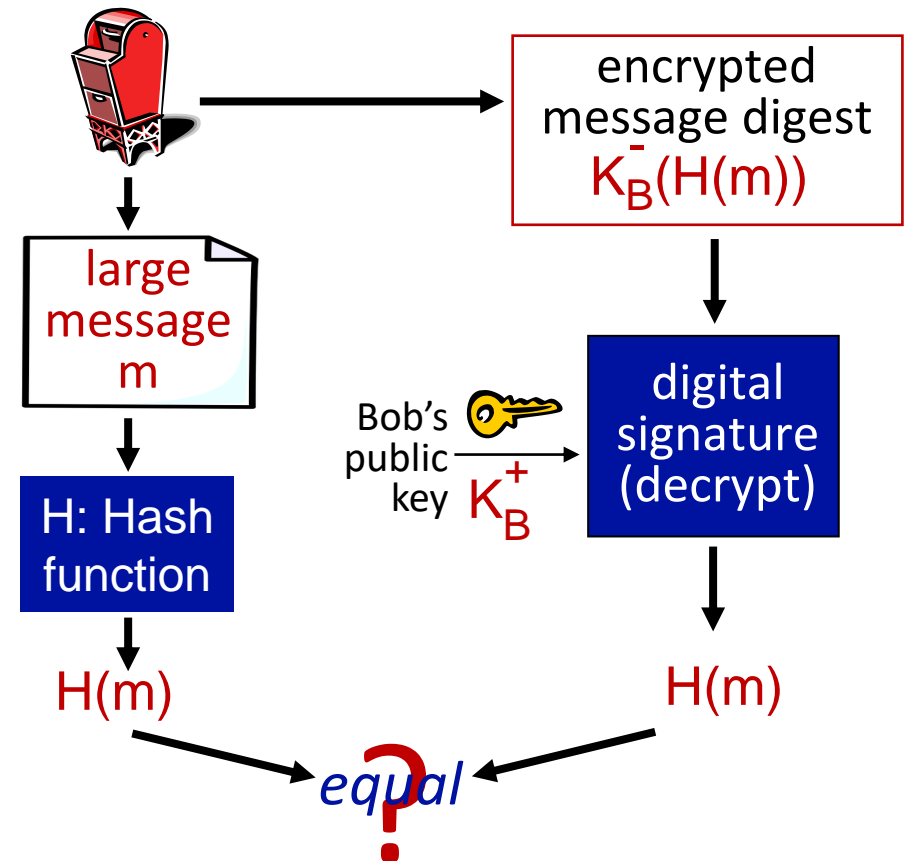
<u>message</u>	<u>ASCII format</u>		<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31		I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39		0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42		9 B O B	39 42 D2 42
<hr/>			<hr/>	
B2 C1 D2 AC		<i>different messages</i> <i>but identical checksums!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

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:

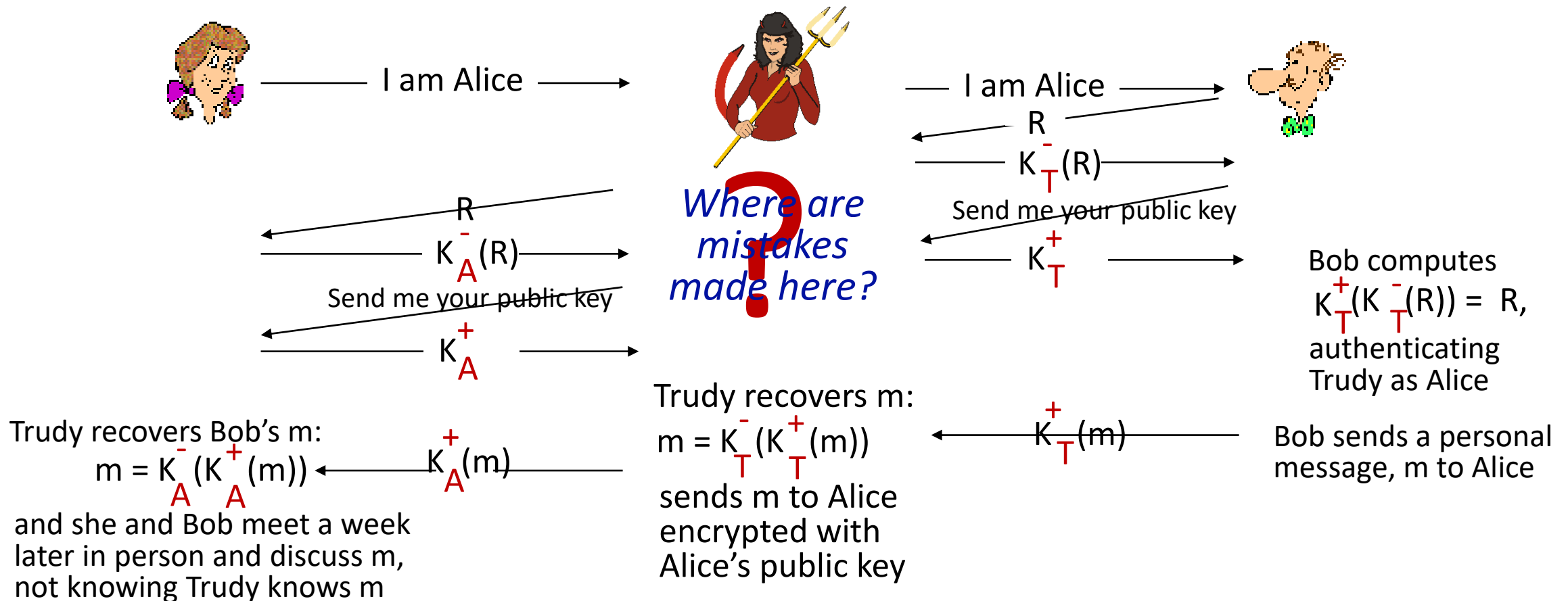


Hash function algorithms

- MD5 hash function widely used (RFC 1321)
 - computes 128-bit message digest in 4-step process.
 - arbitrary 128-bit string x , appears difficult to construct msg m whose MD5 hash is equal to x
- SHA-1 is also used
 - US standard [NIST, FIPS PUB 180-1]
 - 160-bit message digest

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

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



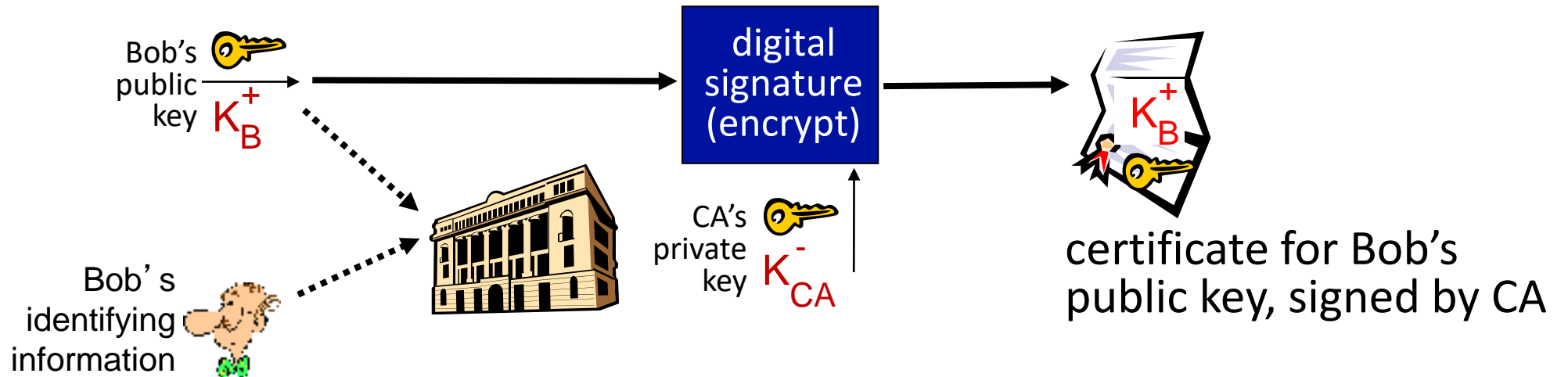
Need for certified public keys

- motivation: Trudy plays pizza prank on Bob
 - Trudy creates e-mail order:
Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
 - Trudy signs order with her private key
 - Trudy sends order to Pizza Store
 - Trudy sends to Pizza Store her public key, but says it's Bob's public key
 - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
 - Bob doesn't even like pepperoni



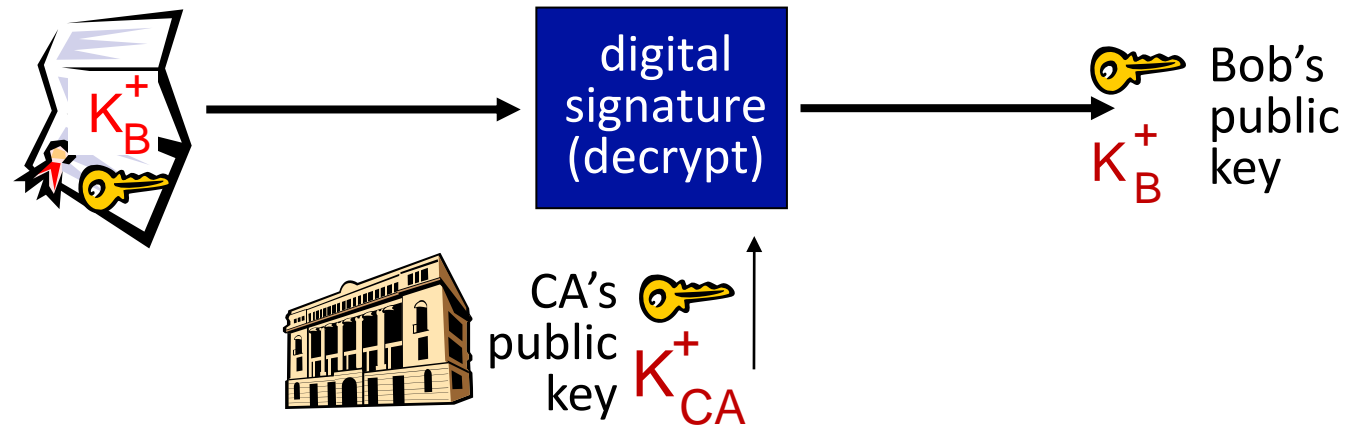
Public key Certification Authorities (CA)

- **certification authority (CA):** binds public key to particular entity, E
- entity (person, website, router) registers its public key with CE 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”



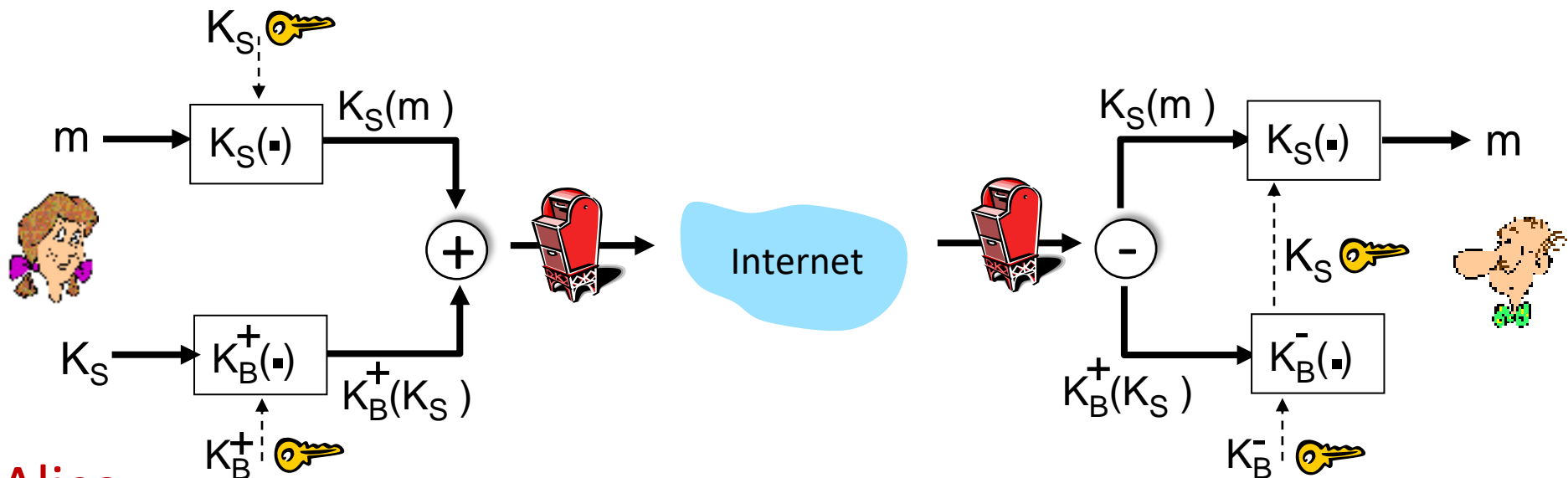
Public key Certification Authorities (CA)

- when Alice wants Bob's public key:
 - gets Bob's certificate (Bob or elsewhere)
 - apply CA's public key to Bob's certificate, get Bob's public key



Secure e-mail: confidentiality

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

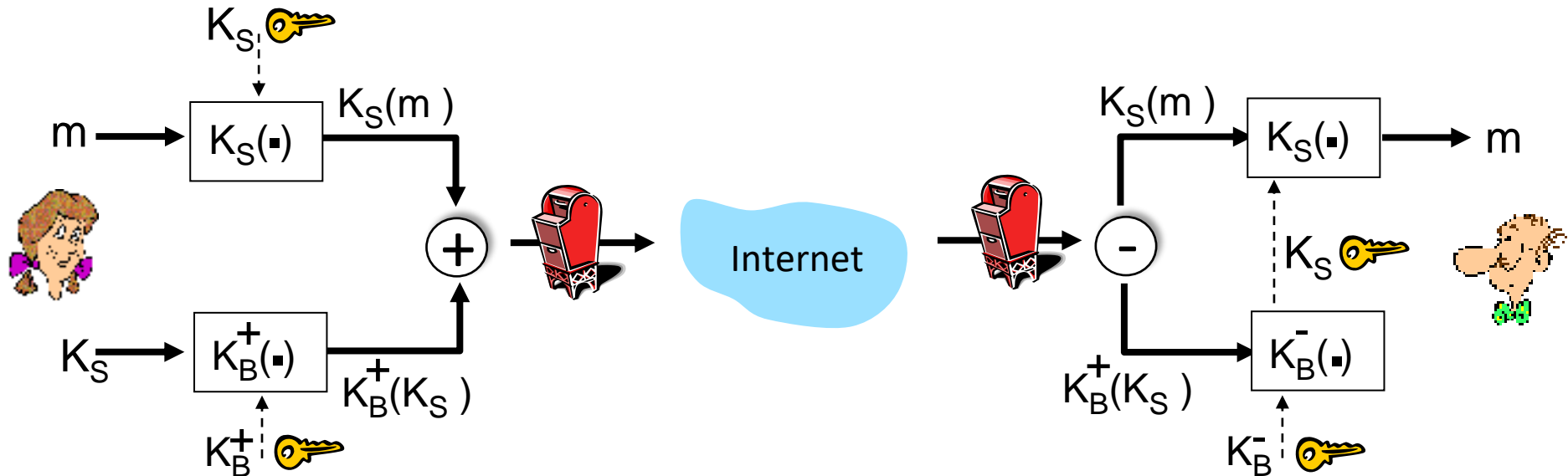


Alice:

- 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.

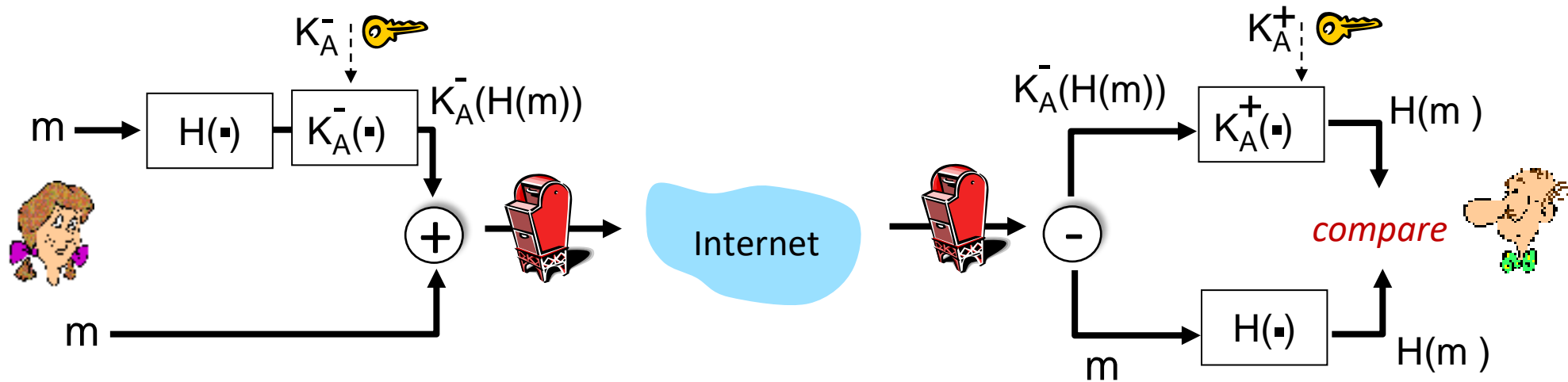


Bob:

- 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

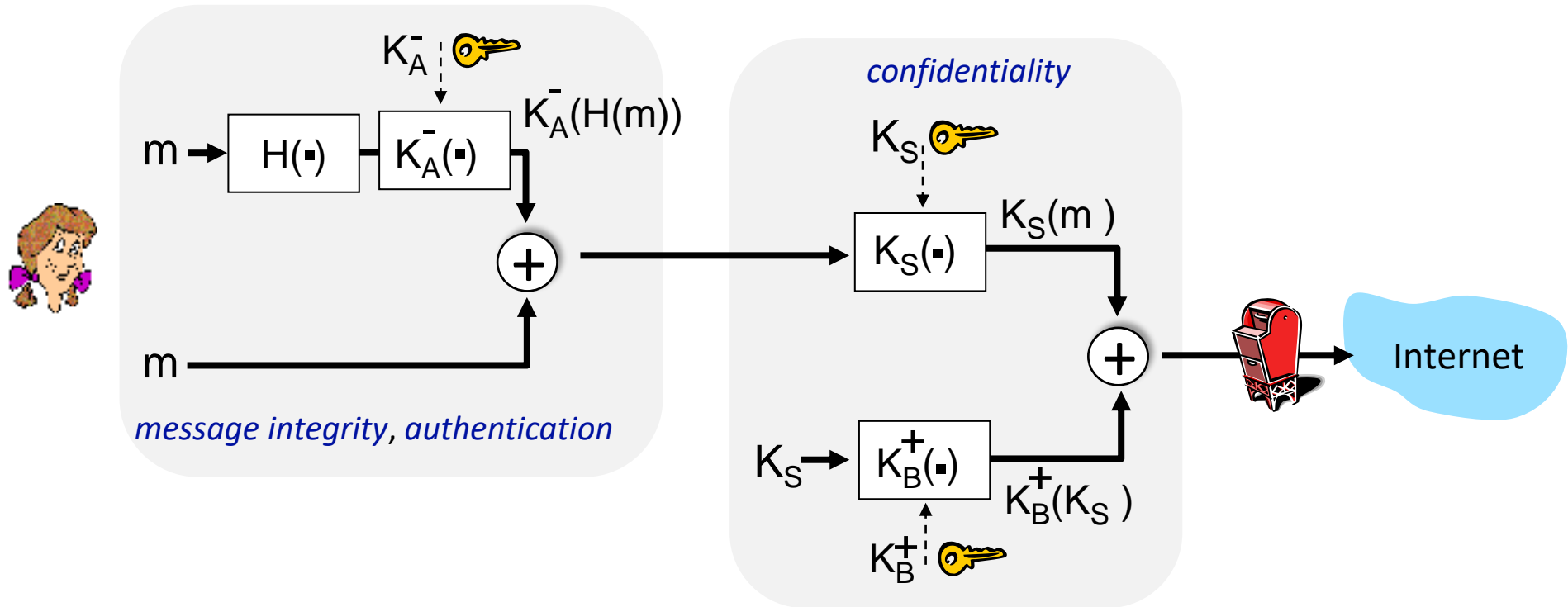
Alice wants to send m to Bob, with *message integrity, authentication*



- Alice digitally signs hash of her message with her private key, providing integrity and authentication
- sends both message (in the clear) and digital signature

Secure e-mail: integrity, authentication

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

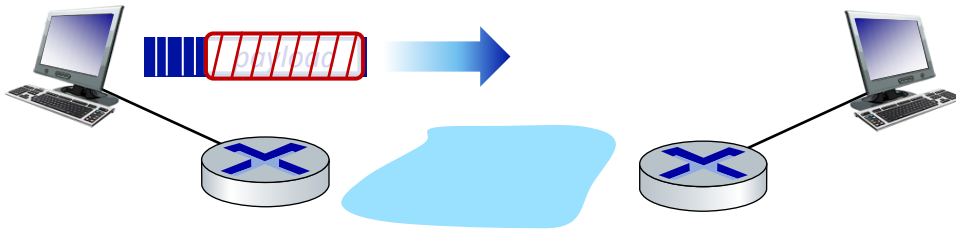


Alice uses three keys: her private key, Bob's public key, new symmetric key

What are Bob's complementary actions?

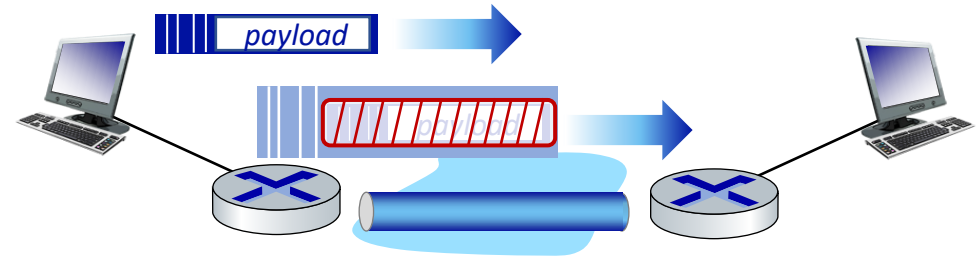
IP Sec

- 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:

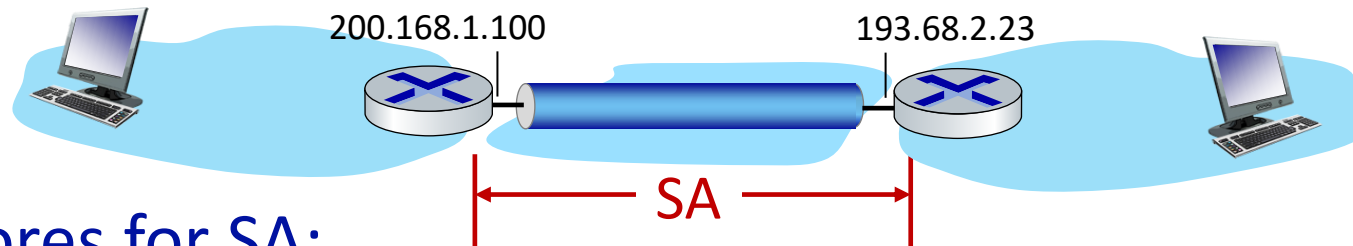
- entire datagram is encrypted, authenticated
- encrypted datagram encapsulated in new datagram with new IP header, tunneled to destination

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

Security associations (SAs)

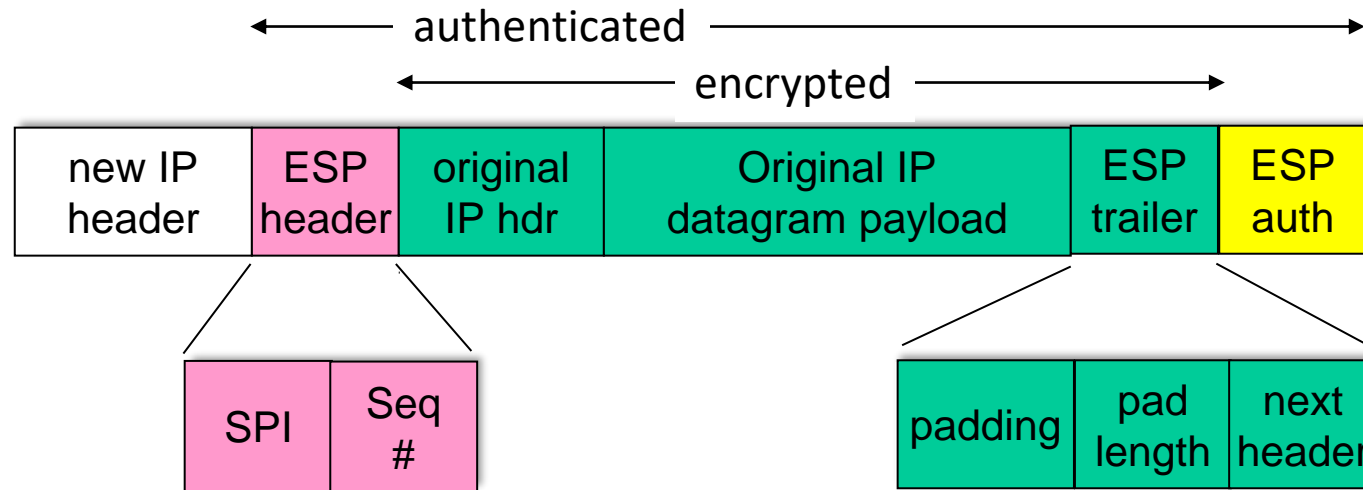
- before sending data, **security association (SA)** established from sending to receiving entity (directional)
- ending, receiving entities maintain *state information* about SA
 - recall: TCP endpoints also maintain state info
 - IP is connectionless; IPsec is connection-oriented!



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 datagram



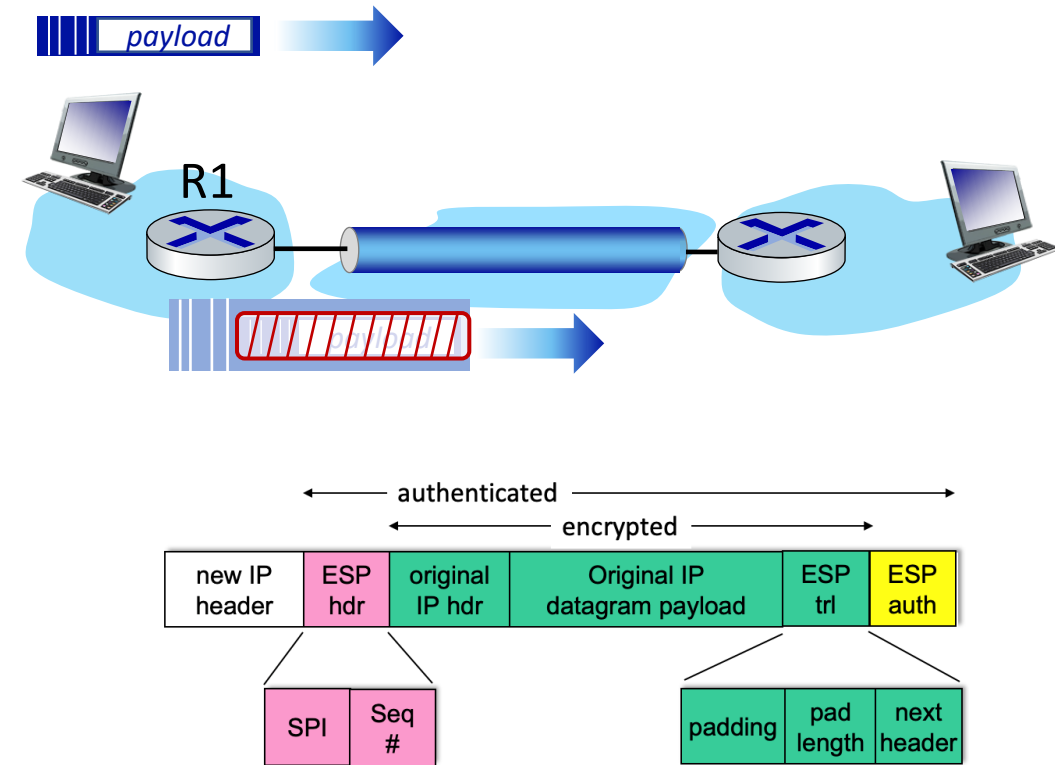
*tunnel mode
ESP*

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

ESP tunnel mode: actions

at R1:

- appends ESP trailer to original datagram (which includes original header fields!)
- encrypts result using algorithm & key specified by SA
- appends ESP header to front of this encrypted quantity
- creates authentication MAC using algorithm and key specified in SA
- appends MAC forming *payload*
- creates new IP header, new IP header fields, addresses to tunnel endpoint



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?

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.

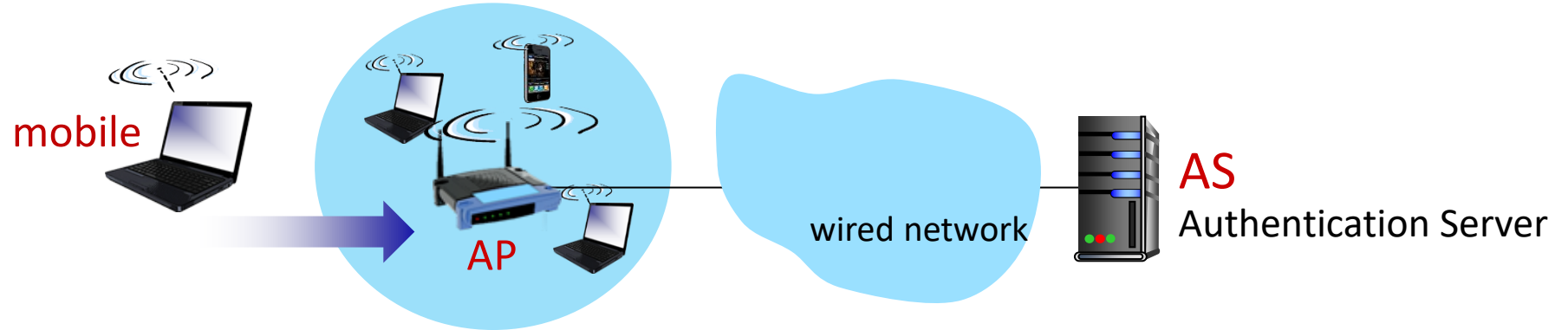
IKE phases

- 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

IPsec summary

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

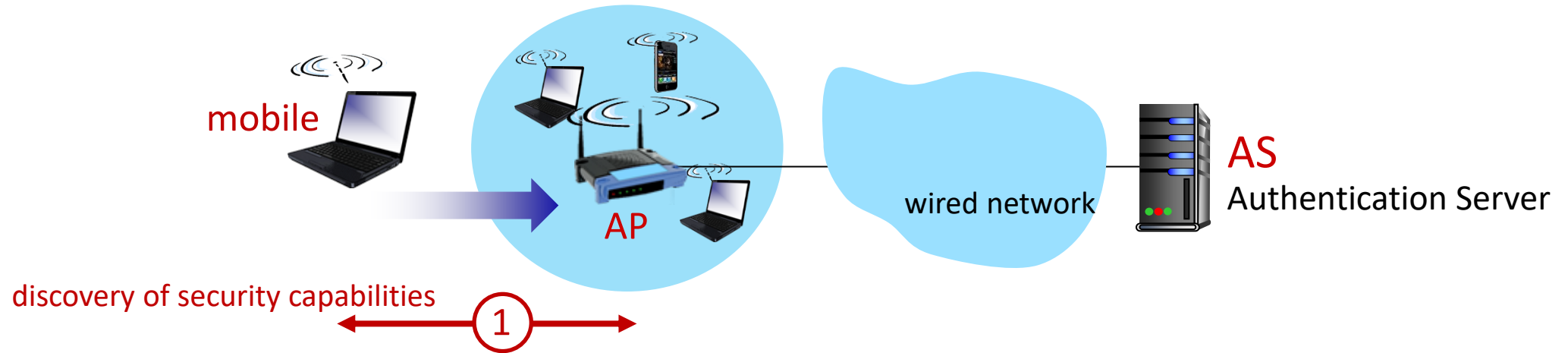
802.11: authentication, encryption



Arriving mobile must:

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

802.11: authentication, encryption

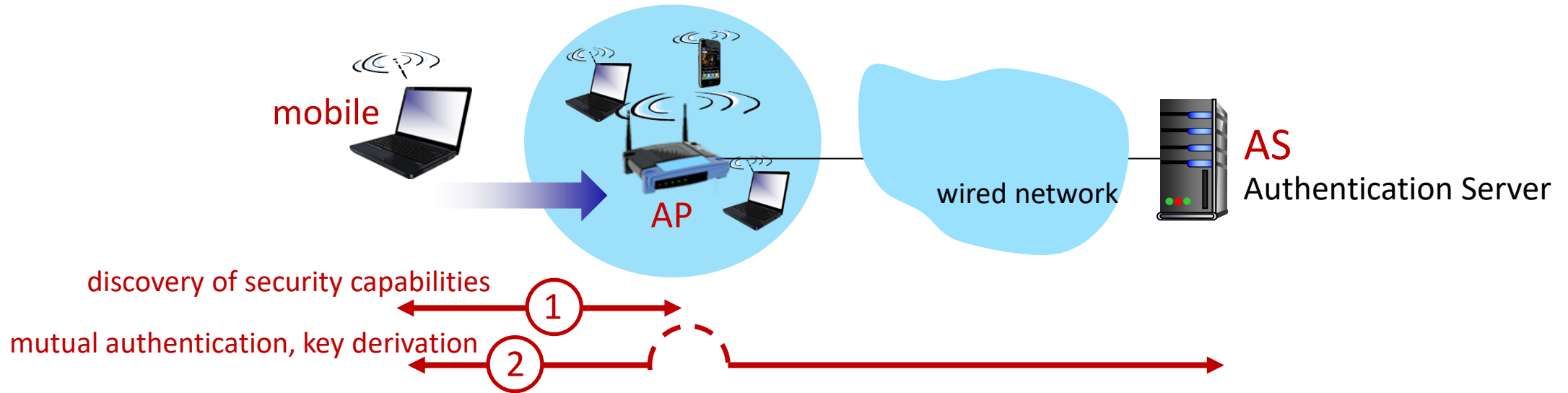


① discovery of security capabilities:

- 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

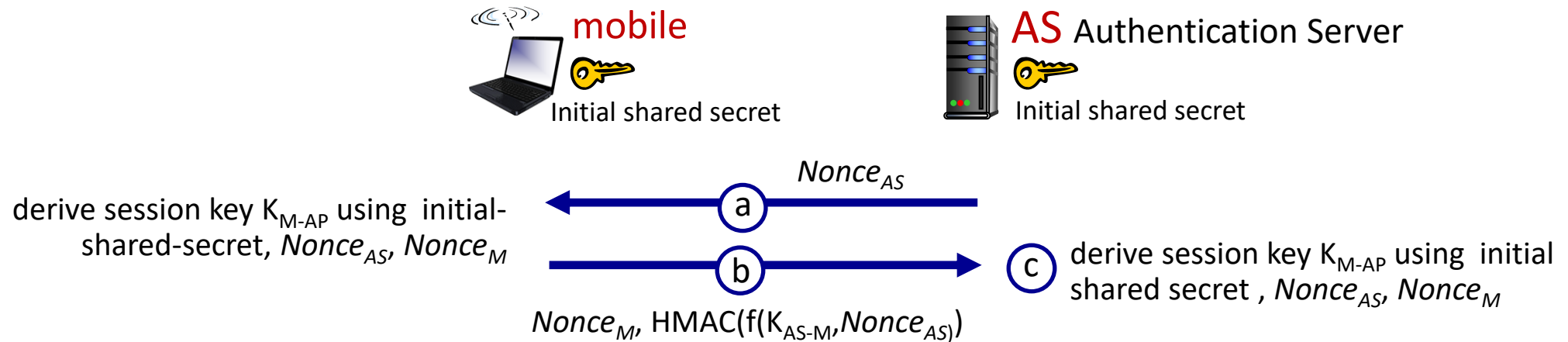
802.11: authentication, encryption



② mutual authentication and shared symmetric key derivation:

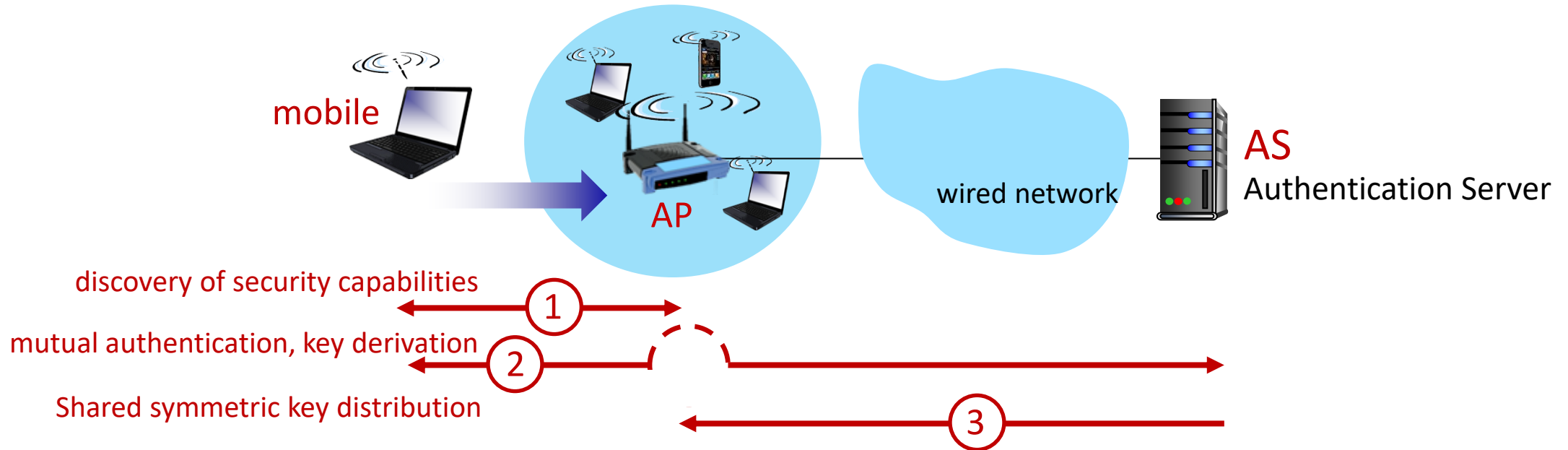
- AS, mobile already have shared common secret (e.g., password)
- AS, mobile use shared secret, nonces (prevent relay attacks), cryptographic hashing (ensure message integrity) to authenticating each other
- AS, mobile derive symmetric session key

802.11: WPA3 handshake



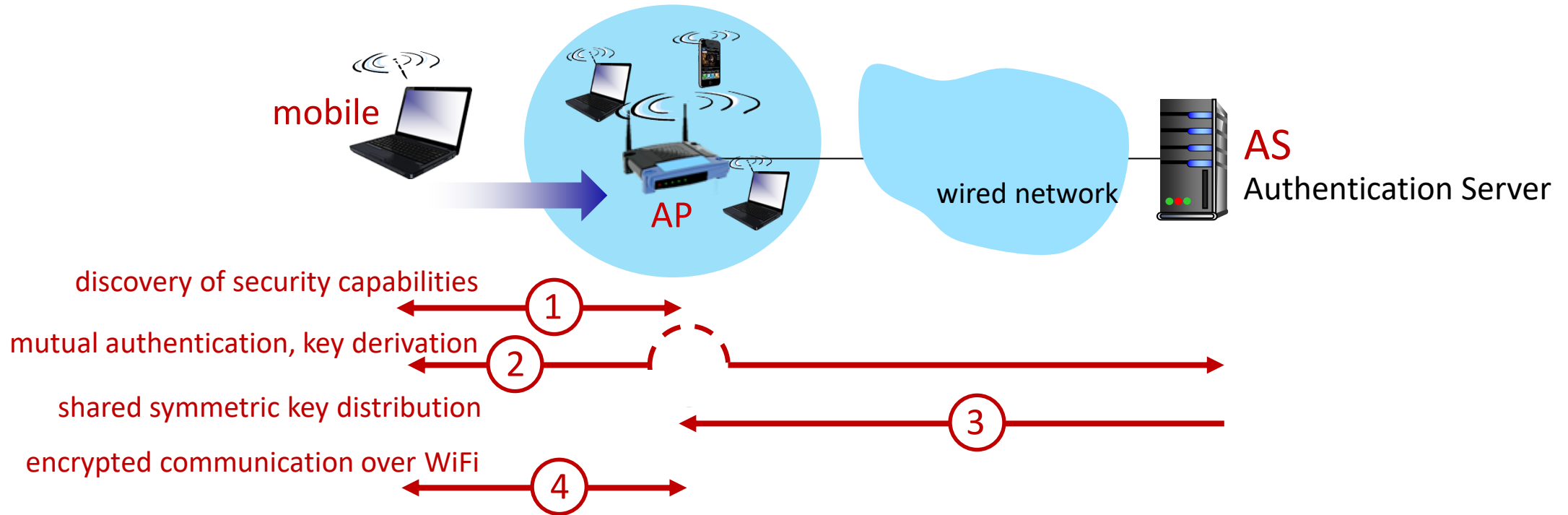
- ① AS generates $Nonce_{AS}$, sends to mobile
- ② mobile receives $Nonce_{AS}$
 - generates $Nonce_M$
 - generates symmetric shared session key K_{M-AP} using $Nonce_{AS}$, $Nonce_M$, and initial shared secret
 - sends $Nonce_M$, and HMAC-signed value using $Nonce_{AS}$ and initial shared secret
- ③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

802.11: authentication, encryption



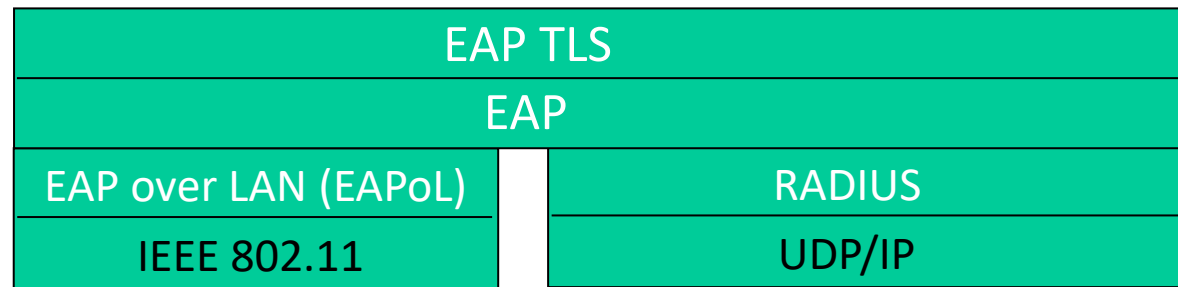
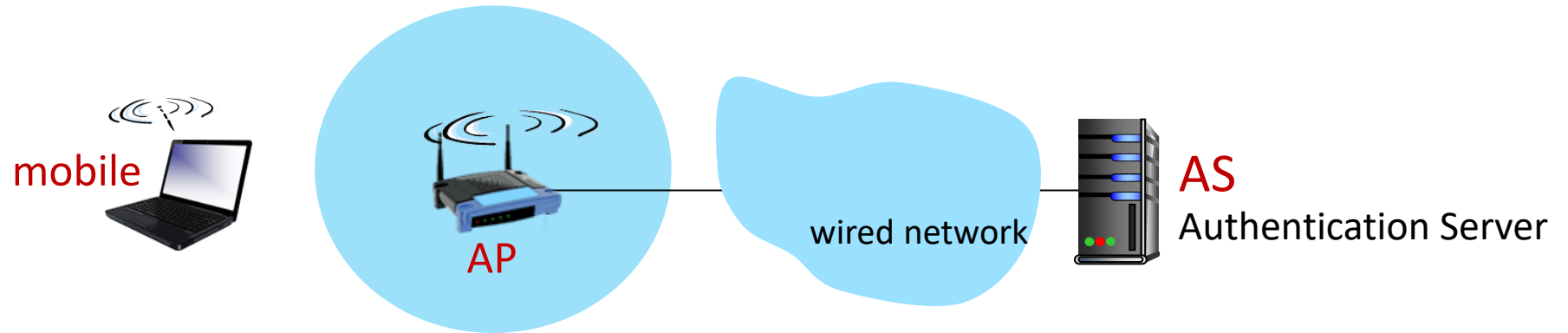
- ③ shared symmetric session key distribution (e.g., for AES encryption)
- same key derived at mobile, AS
 - AS informs AP of the shared symmetric session

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

Authentication, encryption in 4G LTE



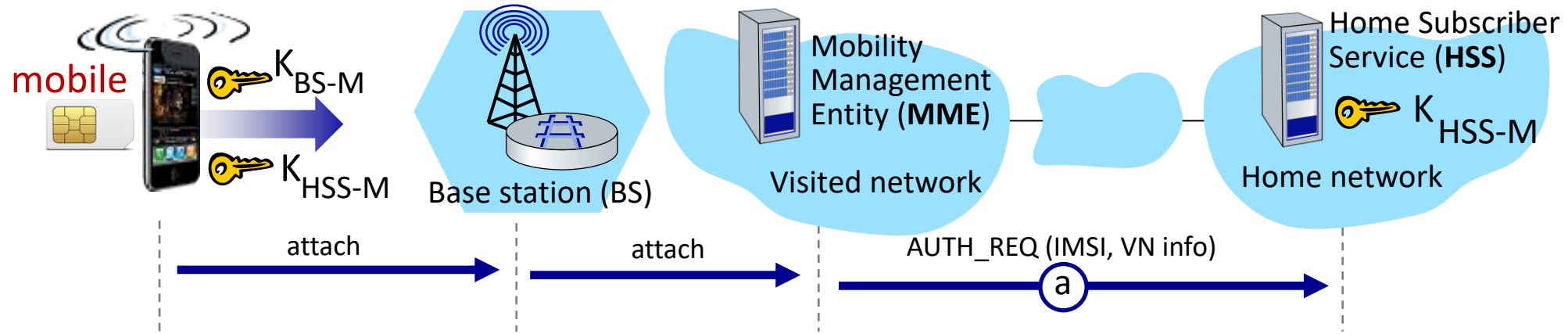
- arriving mobile must:
 - associate with BS: (establish) communication over 4G wireless link
 - authenticate itself to network, and authenticate network
- notable differences from WiFi
 - mobile's SIMcard provides global identity, contains shared keys
 - services in visited network depend on (paid)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

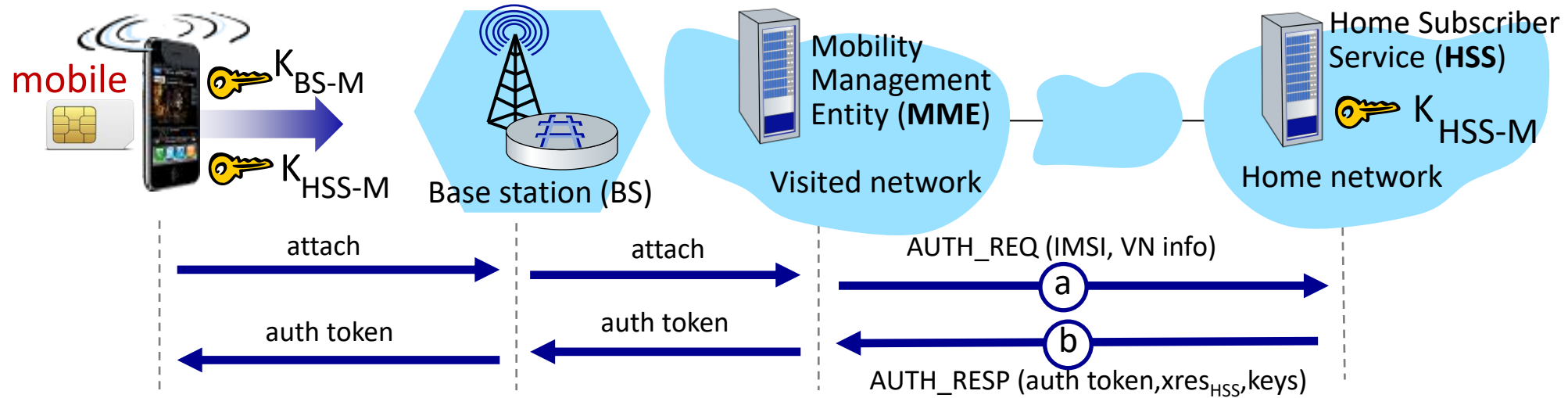
Authentication, encryption in 4G LTE



① authentication request to home network HSS

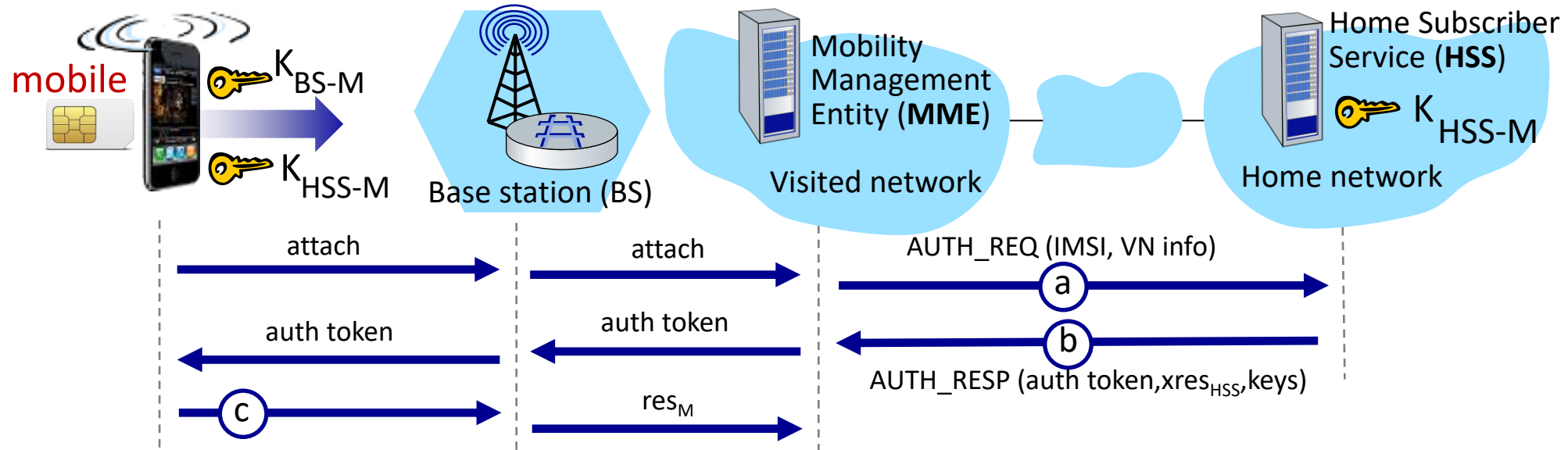
- mobile sends attach message (containing its IMSI, visited network info) relayed from BS to visited MME to home HSS
- IMSI identifies mobile's home network

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

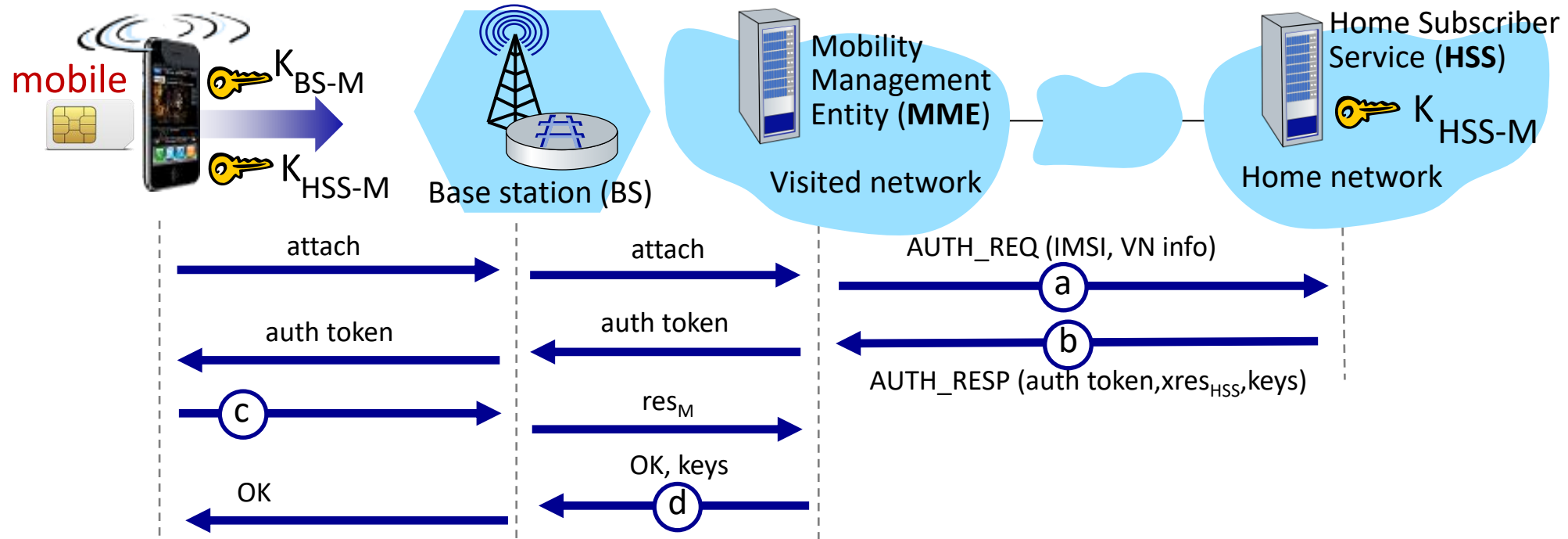
Authentication, encryption in 4G LTE



© 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

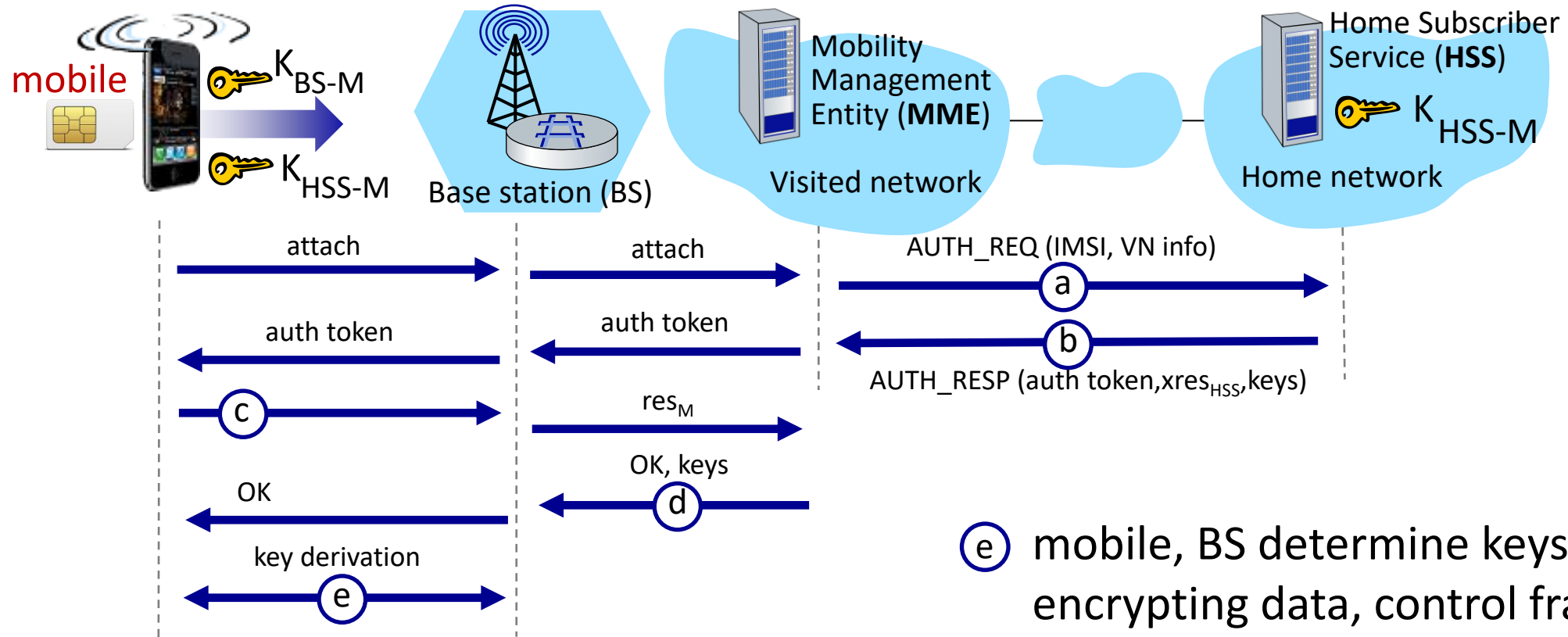
Authentication, encryption in 4G LTE



④ mobile is authenticated by network:

- MME compares mobile-computed value of res_M with the HSS-computed value of $xres_{HSS}$. If they match, mobile is authenticated ! (why?)
- MME 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

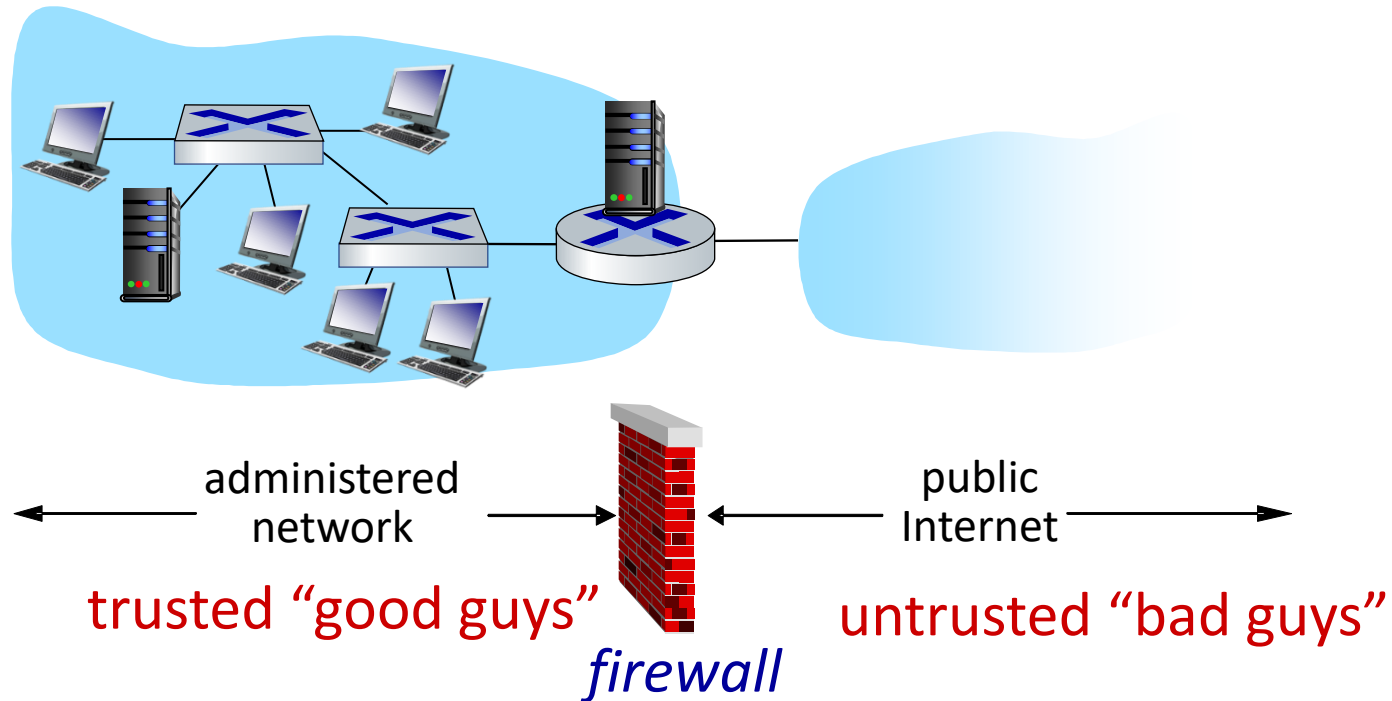
Authentication, encryption: from 4G to 5G

- **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

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

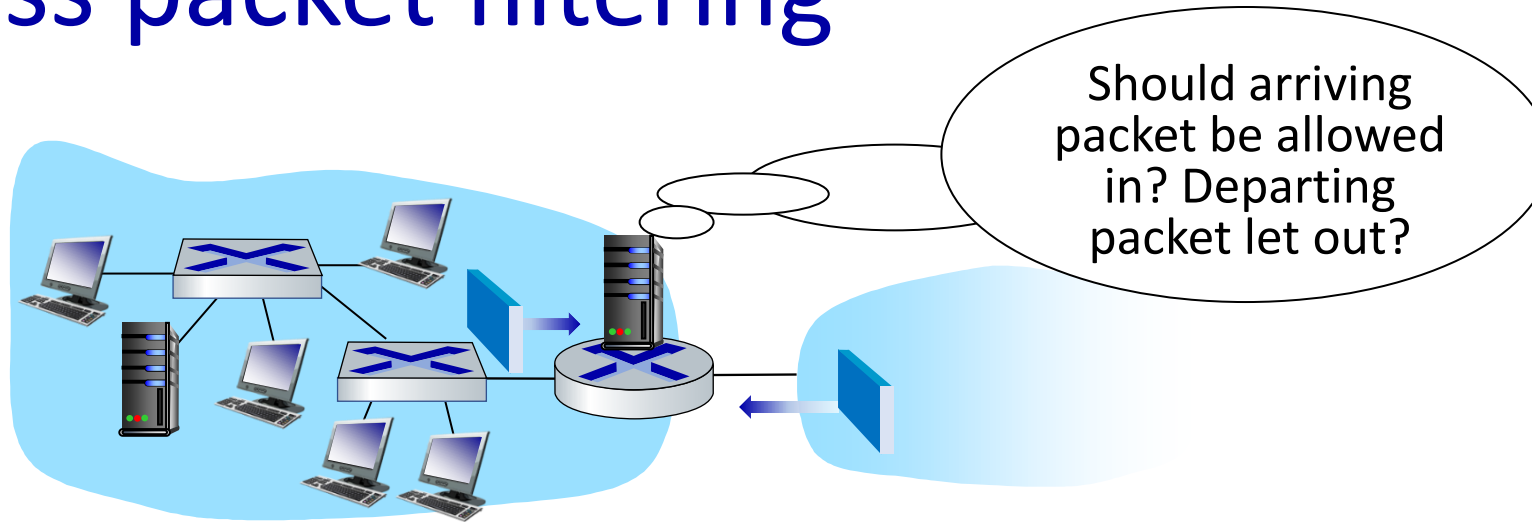
allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

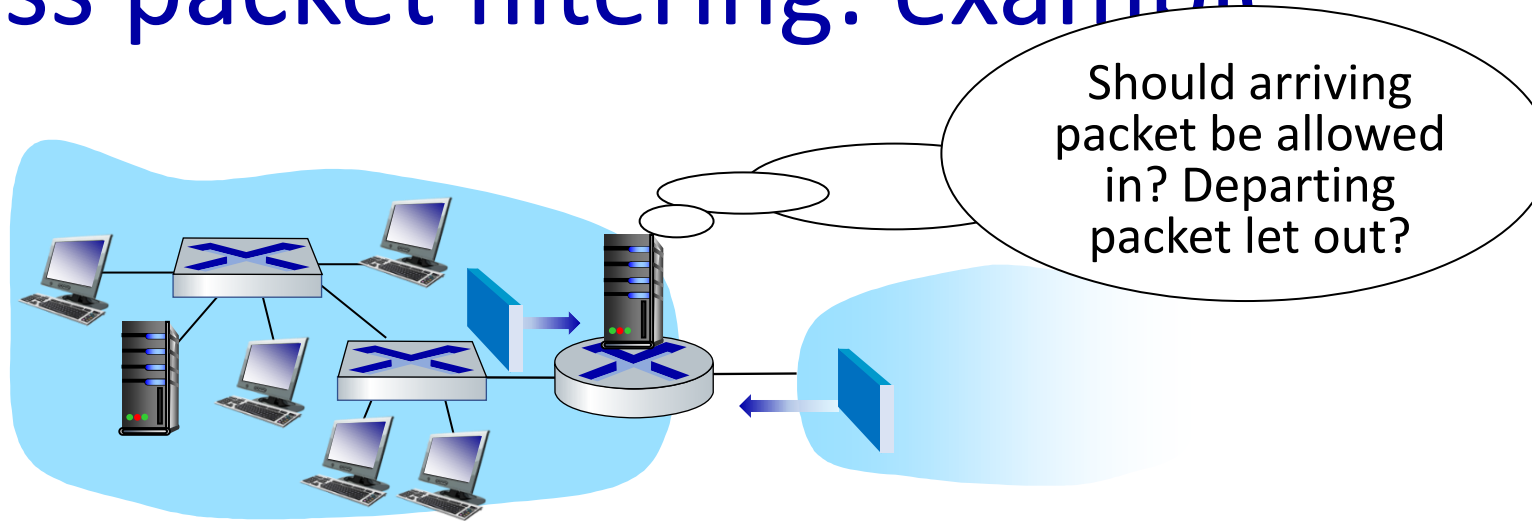
- stateless packet filters
- stateful packet filters
- application gateways

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

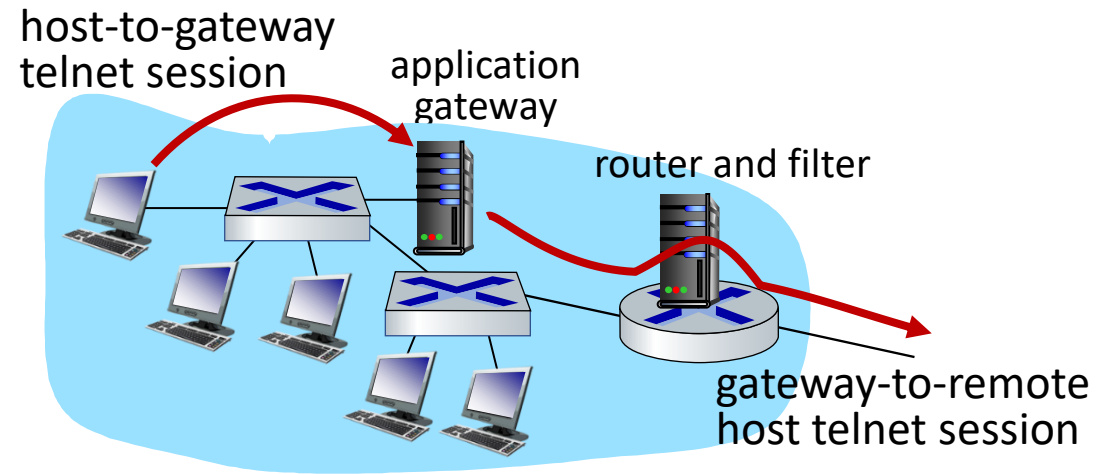
Stateful packet filtering

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