南京大学

NANJING UNIVERSITY

计算机网络中的安全

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)

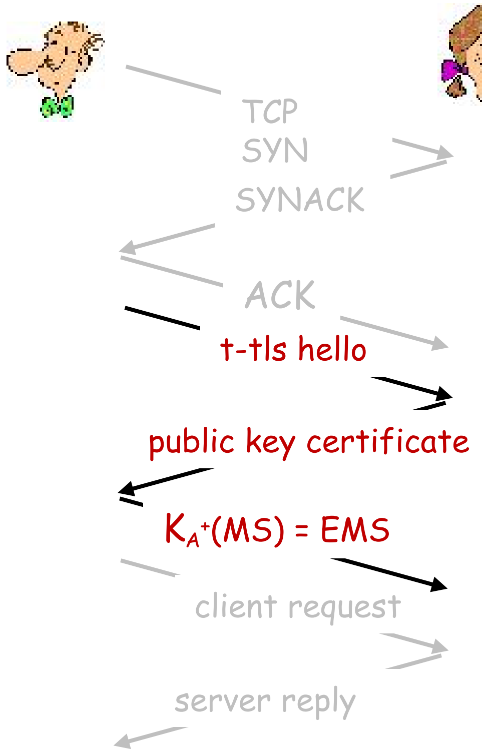
- widely deployed security protocol above the transport layer
  - supported by almost all browsers, web servers: https (port 443)
- provides:
  - confidentiality: via *symmetric encryption*
  - integrity: via *cryptographic hashing*
  - authentication: via *public key cryptography*

  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

t-tls: encrypting data

- recall: TCP provides data *byte stream* abstraction
- Q: can we encrypt data in-stream as written into TCP socket?
  - A: where would MAC go? If at end, no message integrity until all data received and connection closed!
  - solution: break stream in series of "records"
    - ✓ each client-to-server record carries a MAC, created using M_c
    - ✓ receiver can act on each record as it arrives
- t-tls record encrypted using symmetric key, $K_{c,}$ passed to TCP:

$K_c($

length	data	MAC

$)$

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)
  - use nonce

t-tls: connection close

- truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is
- solution: record types, with one type for closure
  - type 0 for data; type 1 for close
- MAC now computed using data, type, sequence #

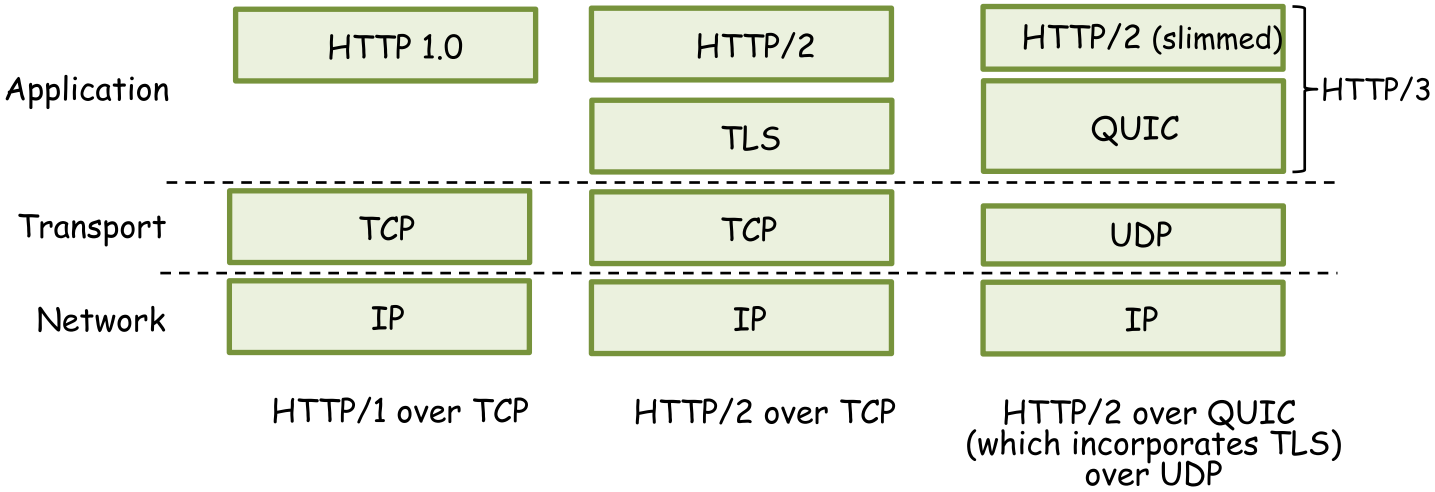
K_c(

length	type	data	MAC

)

Transport-layer security (TLS)

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

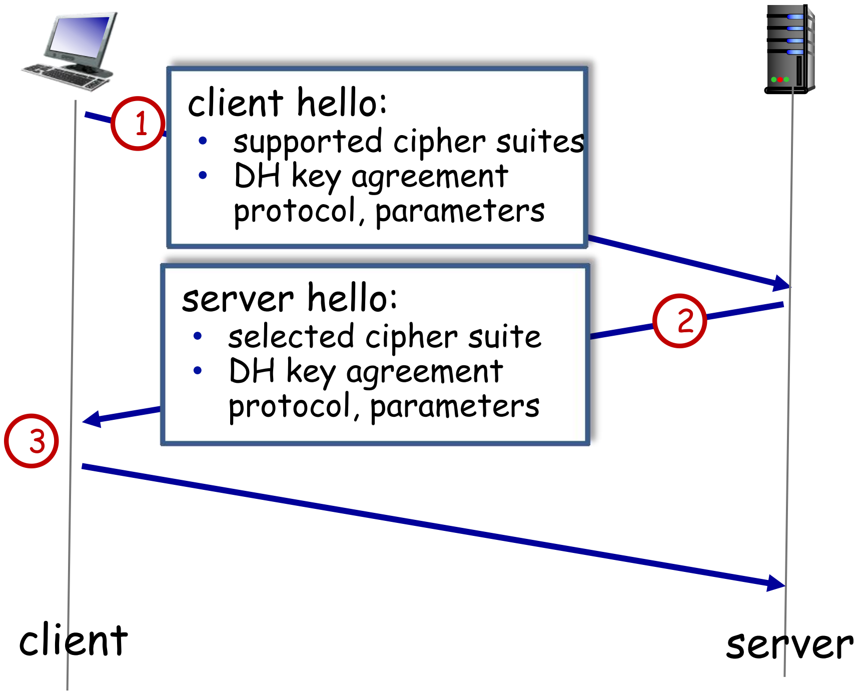


TLS: 1.3 cipher suite

- "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
  - combined encryption and authentication algorithm ("authenticated encryption") for data rather than serial encryption, authentication
    - ✓ 4 based on AES
  - HMAC uses SHA (256 or 284) cryptographic hash function

TLS 1.3 handshake: 1 RTT

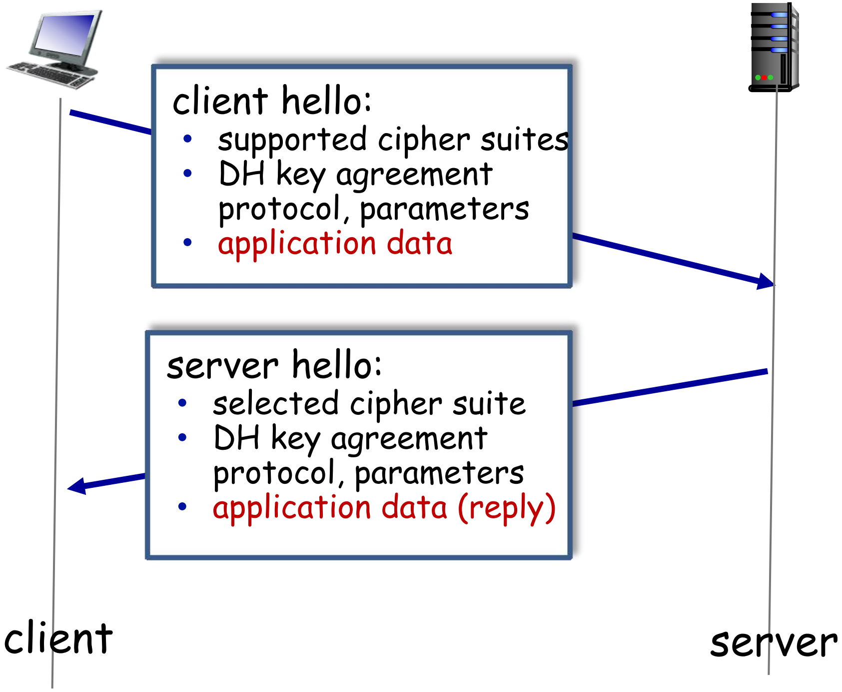


1. client TLS hello msg:
   - *guesses* key agreement protocol, parameters
   - indicates cipher suites it supports

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



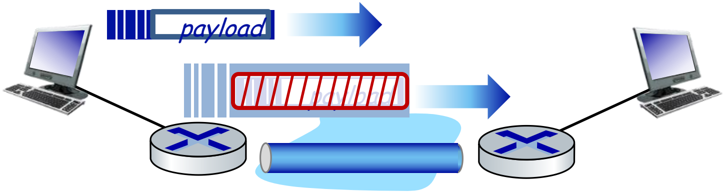
- initial hello message contains encrypted application data!
  - "resuming" earlier connection between client and server
  - application data encrypted using "resumption master secret" from earlier connection
- vulnerable to replay attacks!
  - maybe OK for get HTTP GET or client requests not modifying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

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

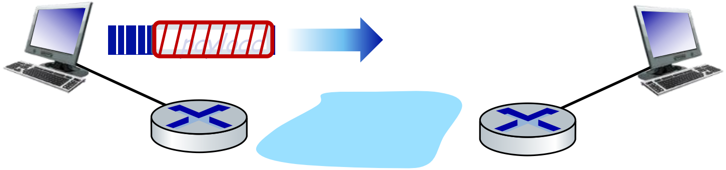
IP Sec

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



tunnel mode:

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



transport mode:

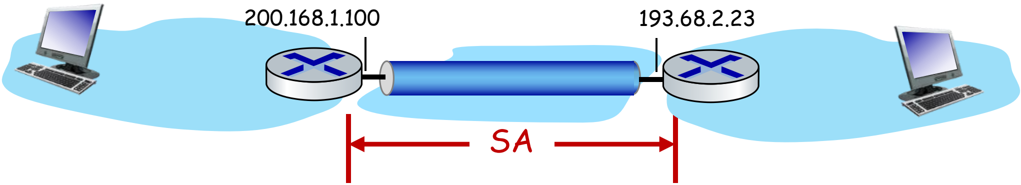
- *only* datagram *payload* is encrypted, authenticated

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)

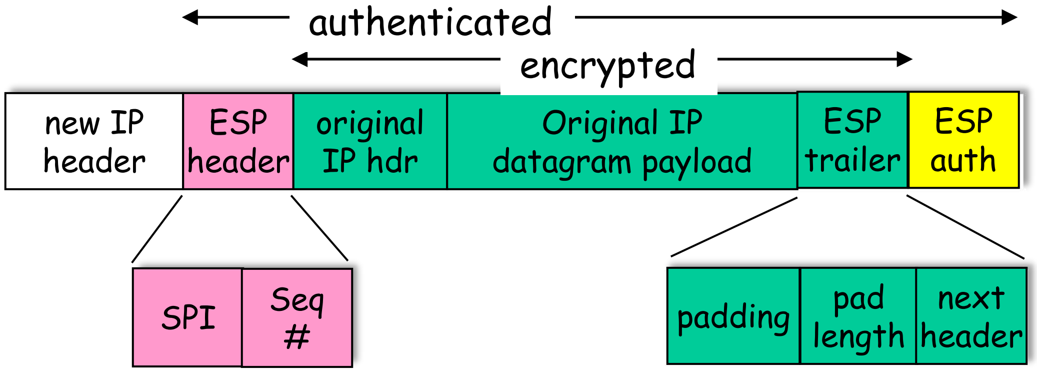
- before sending data, security association (SA) established from sending to receiving entity (directional)
- ending, receiving entitles maintain *state information* about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!



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



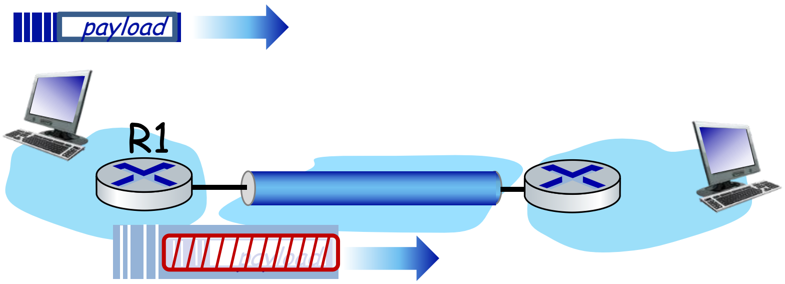
tunnel mode ESP

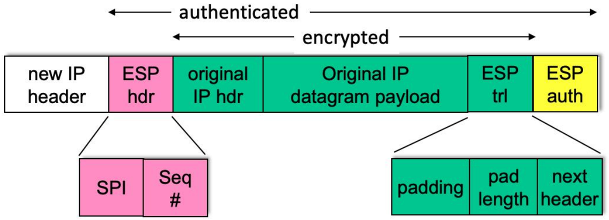
- ESP trailer: padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - sequence number, to thwart replay attacks
- MAC in ESP auth field created with shared secret key

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

没收到的包就不要了d

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

SPD: "what" to do

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.

SAD: "how" to do it

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 (pubic/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

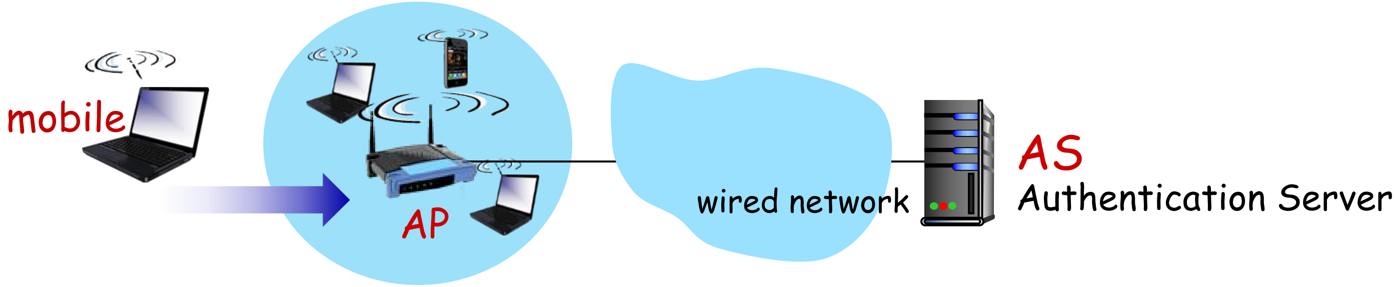
IPsec summary

- 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

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

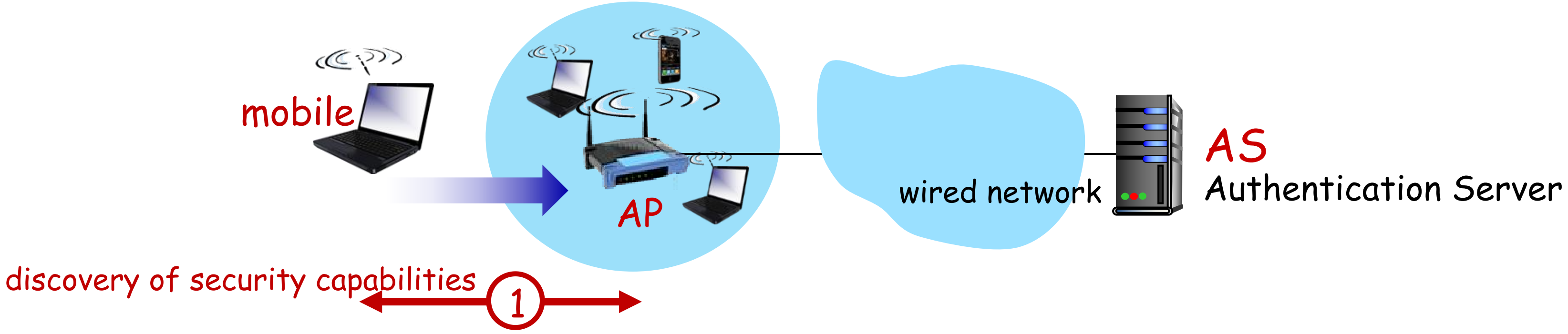
802.11: authentication, encryption



Arriving mobile must:

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

802.11: authentication, encryption

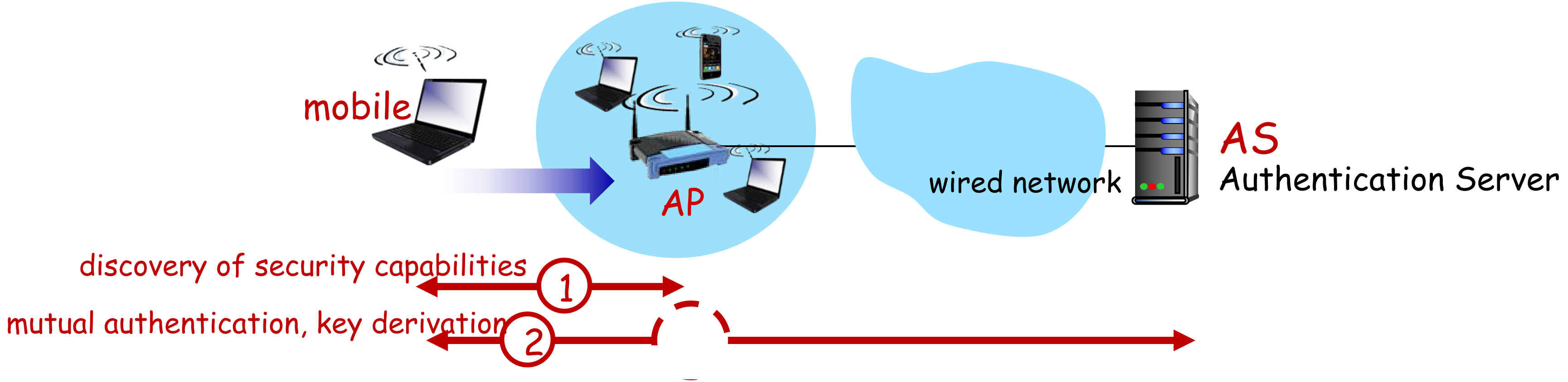


1 discovery of security capabilities:

- 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

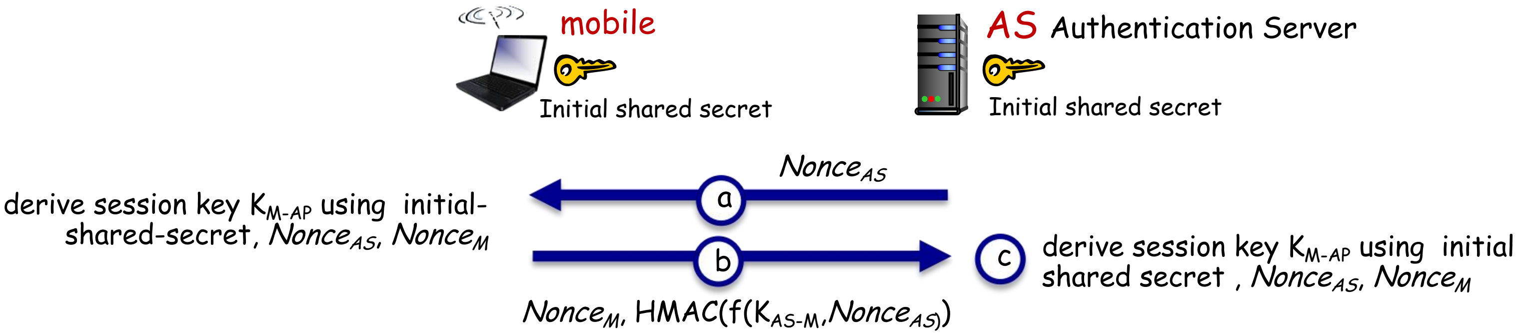
802.11: authentication, encryption



2 mutual authentication and shared symmetric key derivation:

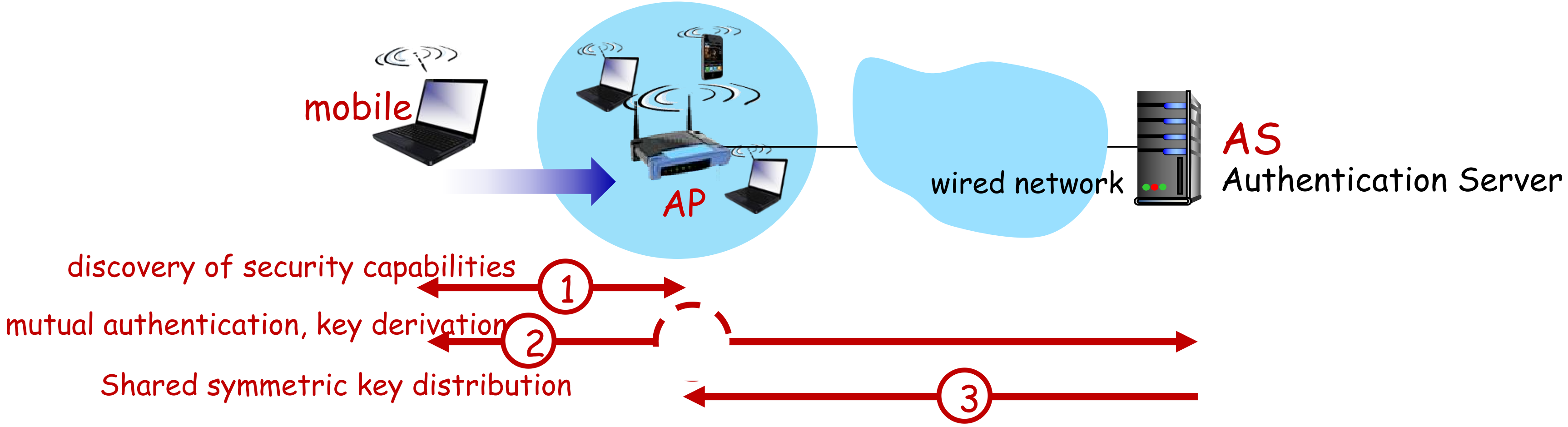
- AS, mobile already have shared common secret (e.g., password)
- AS, mobile use shared secret, nonces (prevent relay attacks), cryptographic hashing (ensure message integrity) to authenticating each other
- AS, mobile derive symmetric session key

802.11: WPA3 handshake



- (a) AS generates *Nonce*$_{AS}$, sends to mobile
- (b) mobile receives *Nonce*$_{AS}$
  - generates *Nonce*$_{M}$
  - generates symmetric shared session key $K_{M\text{-}AP}$ using *Nonce*$_{AS}$, *Nonce*$_{M}$, and initial shared secret
  - sends *Nonce*$_{M}$, and HMAC-signed value using Nonce$_{AS}$ and initial shared secret
- (c) AS derives symmetric shared session key $K_{M\text{-}AP}$

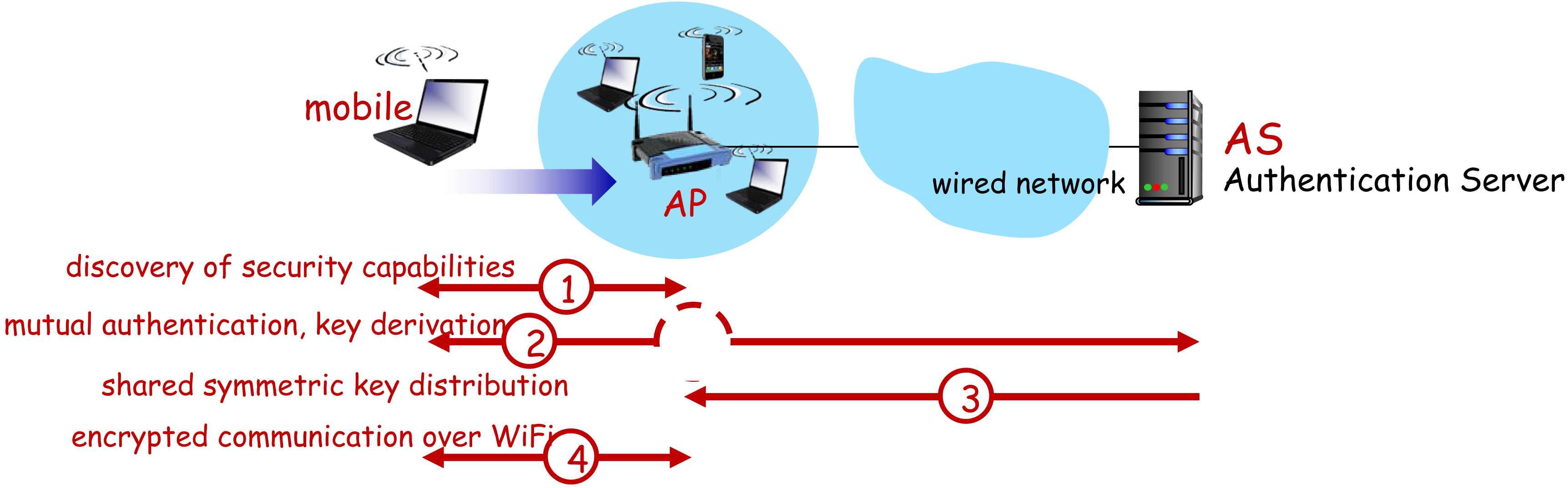
802.11: authentication, encryption



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

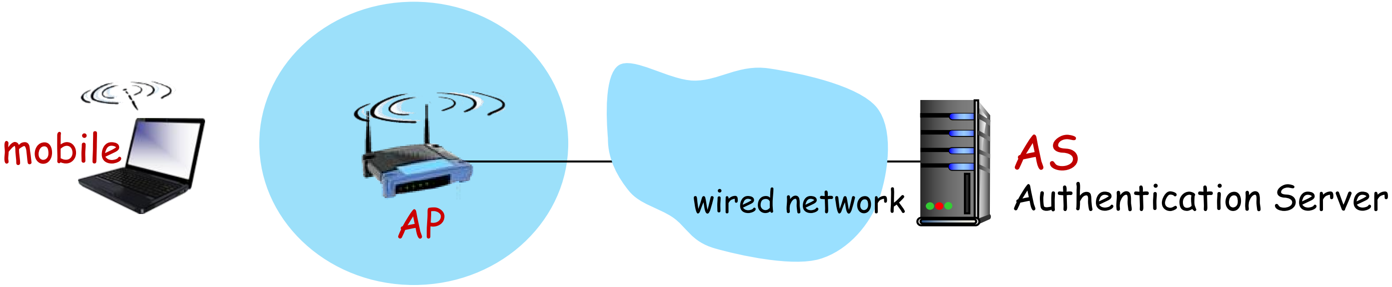
802.11: authentication, encryption



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



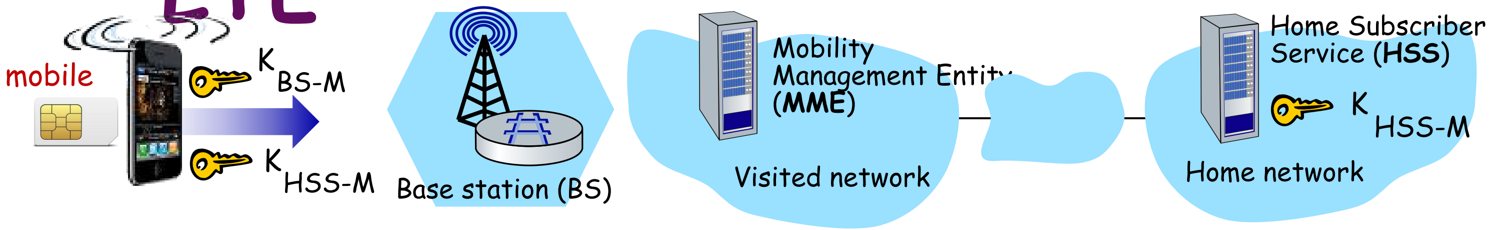
EAP TLS	
EAP	
EAP over LAN (EAPoL)	RADIUS
IEEE 802.11	UDP/IP

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

Authentication, encryption in 4G LTE

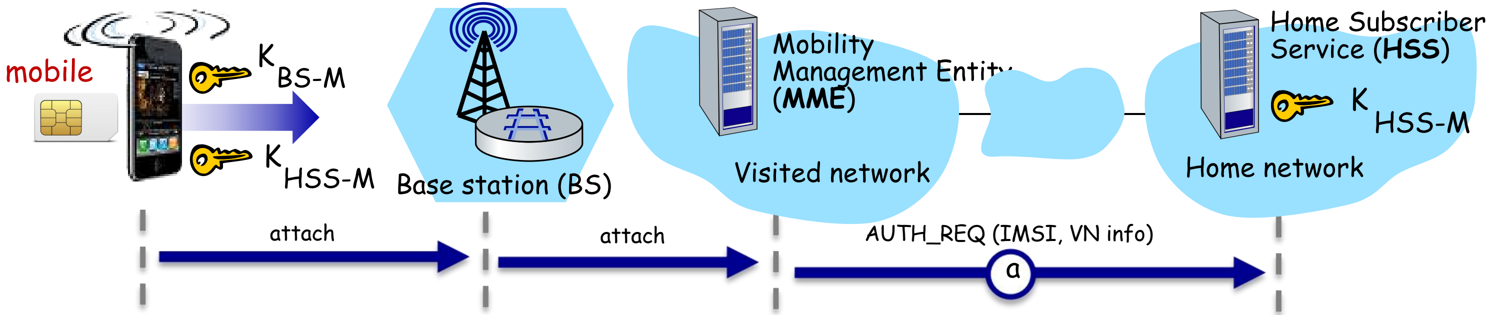
- arriving mobile must:
  - associate with BS: (establish) communication over 4G wireless link
  - authenticate itself to network, and authenticate network
- notable differences from WiFi
  - mobile's SIMcard provides global identity, contains shared keys
  - services in visited network depend on (paid) 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 + HHS in home network, together play role of WiFi AS
  - ultimate authenticator is HSS
  - trust and business relationship between visited and home networks

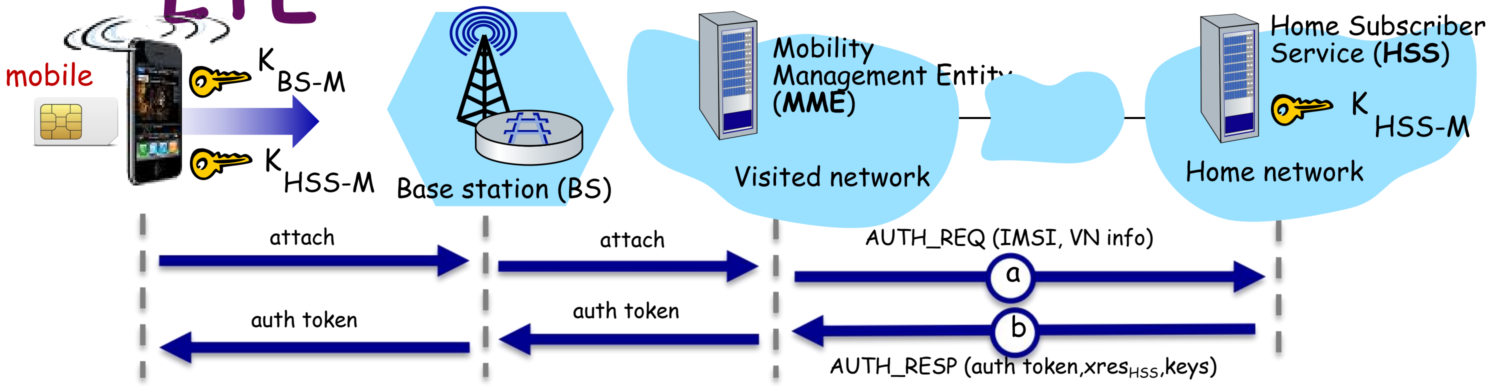
Authentication, encryption in 4G LTE



ⓐ authentication request to home network HSS

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

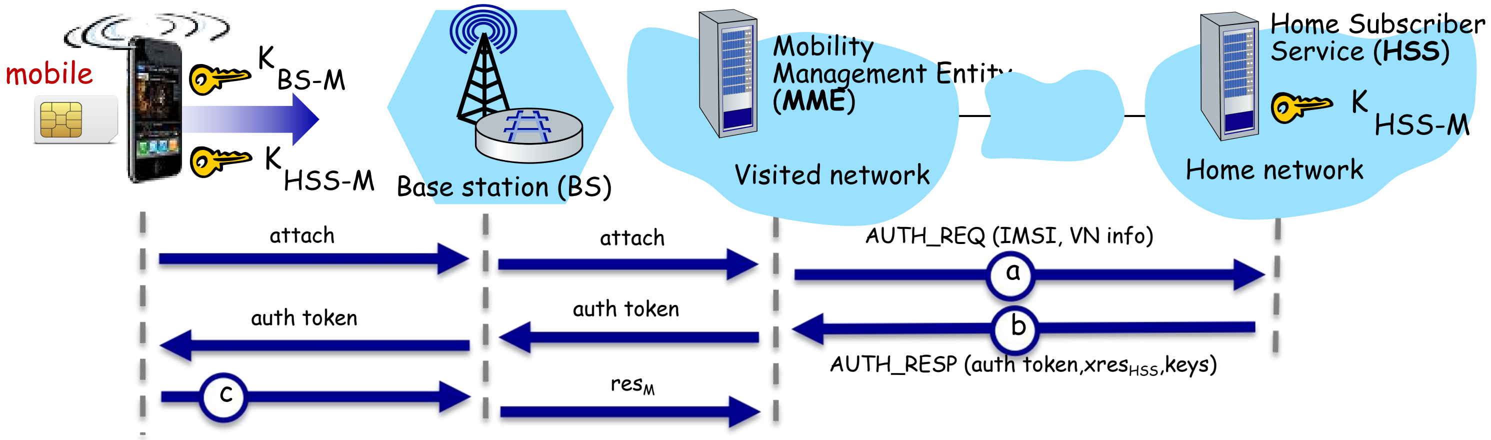
Authentication, encryption in 4G LTE



(b) 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

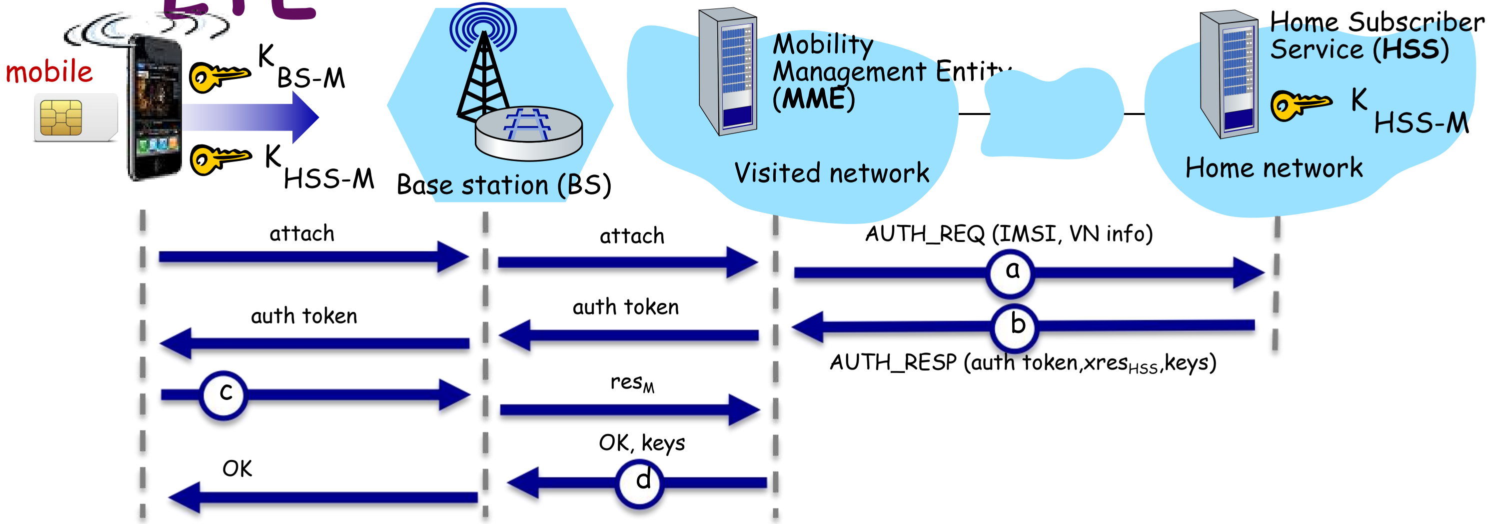


ⓒ authentication response from mobile:

- 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

南京大學 NANJING UNIVERSITY

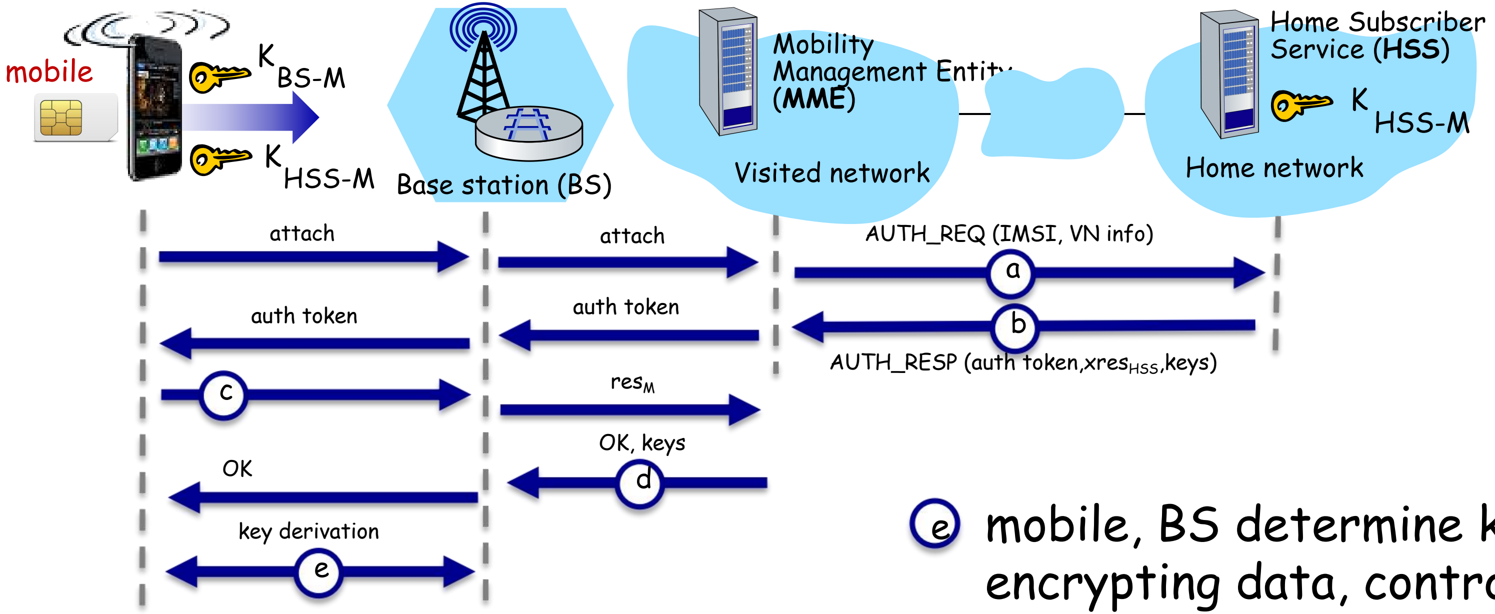
Authentication, encryption in 4G LTE



d 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



e 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

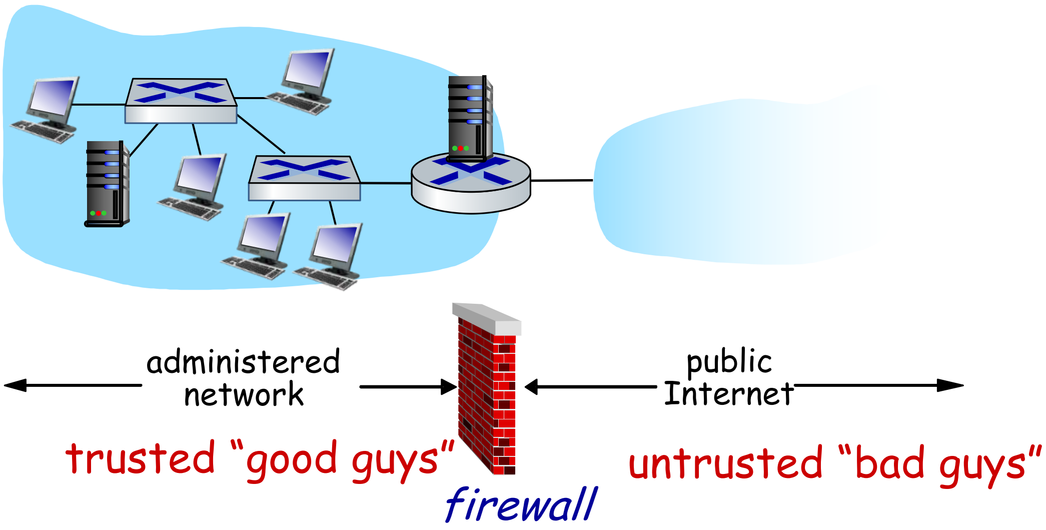
Outline

- 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

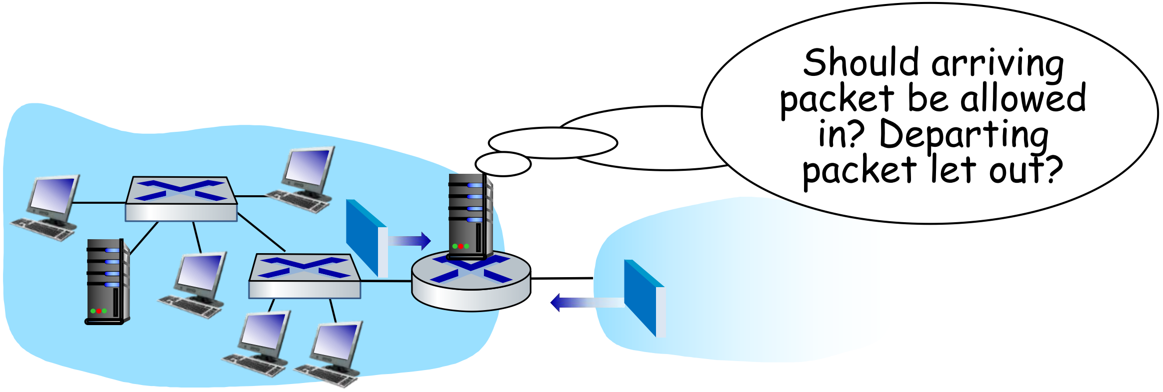
allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

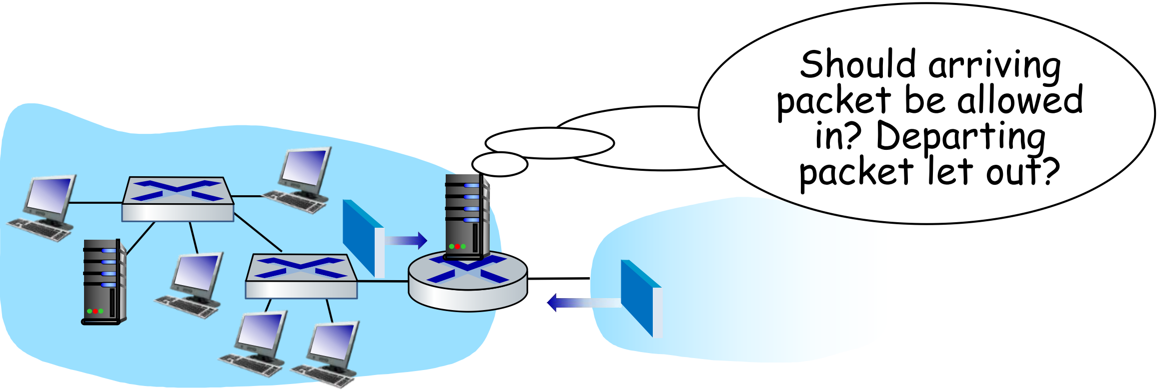
- stateless packet filters
- stateful packet filters
- application gateways

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

NANJING UNIVERSITY

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

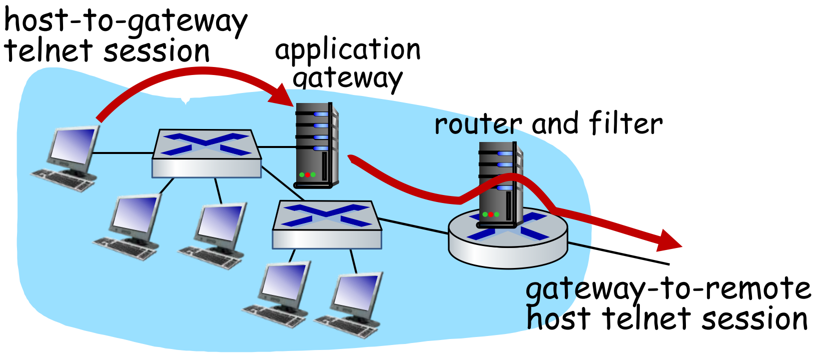
Stateful packet filtering

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	✗
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	✗
deny	all	all	all	all	all	all	

Application gateways

- 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

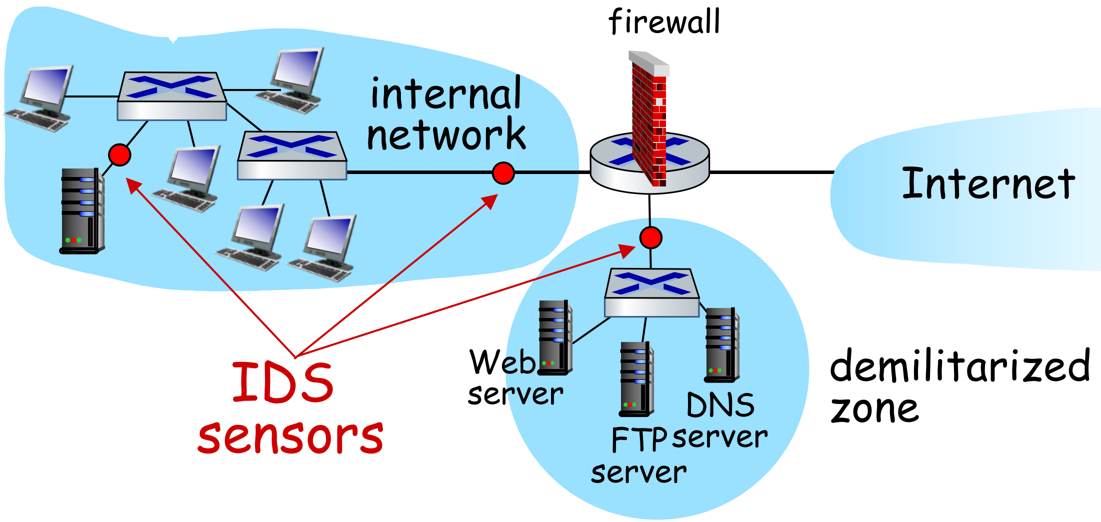
- IP spoofing: router can't know if data "really" comes from claimed source
- if multiple apps need special treatment, each has own app. gateway
- client software must know how to contact gateway
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
-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



提问

Q & A