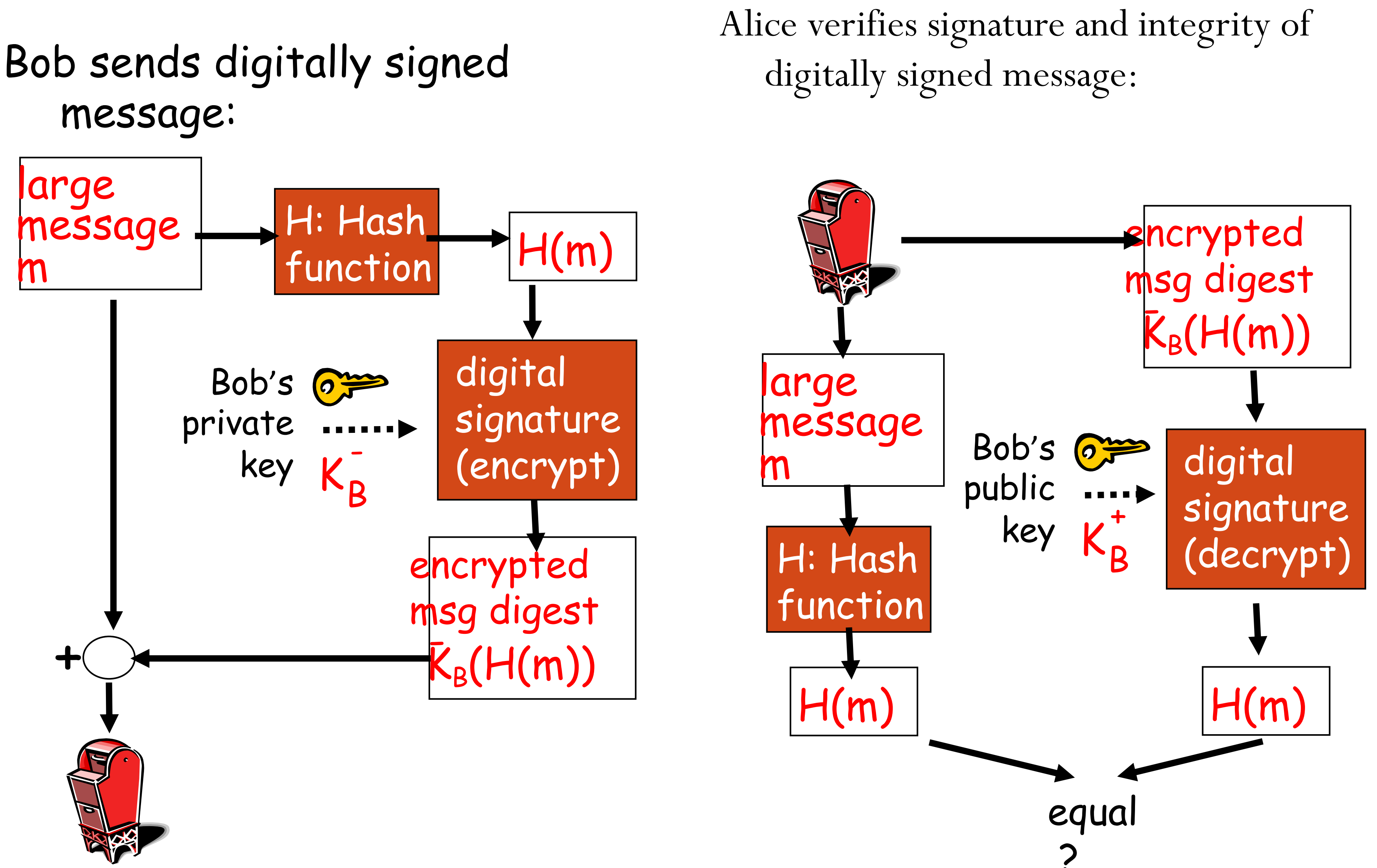
Digital signature = signed message digest



wireless.McMaster.CA

wireless.McMaster.CA

Issued by: thawte SSL CA - G2

Expires: Saturday, February 16, 2019 at 6:59:59 PM Eastern Standard Time

This certificate is marked as trusted for this account

▶ **Trust**

▼ **Details**

Subject Name	
Country	CA
State/Province	ONTARIO
Locality	Hamilton
Organization	McMaster University
Organizational Unit	Technology Services
Common Name	wireless.McMaster.CA
Issuer Name	
Country	US
Organization	thawte, Inc.
Common Name	thawte SSL CA - G2
Serial Number	25 DA 8D 8F 46 75 9D 0C A3 DF 61 62 4B F6 D9 D0
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	none
Not Valid Before	Thursday, January 7, 2016 at 7:00:00 PM Eastern Standard Time
Not Valid After	Saturday, February 16, 2019 at 6:59:59 PM Eastern Standard Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : B6 27 03 AF 50 30 BD E4 7F 1D 70 40 94 36 23 C0 2D A9 FB 26 66 96 96 6C 24 19 8F 07 24 5F 2E 8F D4 AB 3C D7 3D 88 60 6B F0 A7 DB D0 AE 38 AB DB F6 67 18 30 1D DD A1 13 16 23 51 E5 92 D0 28 11 CD F7 52 5A 6C AF 39 22 45 E9 08 BB 8D C4 DB 6A 46 71 F2 A8 95 34 E3 EB 0F 98 B9 F7 05 DB 3A 52 E4 AE 18 EA E1 FE D3 BE D4 F5 DE 65 D5 83 3A 4F 26 B8 94 D9 8E 13 B2 17 5E 79 BD 86 1B F5 66 08 0A FC C8 A8 D8 6C D5 91 32 D7 64 0A 76 2B 9F 7E 7E B7 7B 56 C4 4C 54 AF 78 EB 99 65 D6 61 DD B0 75 14 03 F5 03 88 3A B0 99 1B 17 61 F2 1E CD 46 8F 25 FB 90 C4 8F 36 2B 9A 45 26 A5 07 FB 4E 50 D7 9E C3 88 18 55 68 E0 B0 9D DD DE 59 44 C1 E4 A5 C2 90 C3 C1 5D 4A 81 2A 9B C8 2A 31 3C 23 93 CF D6 DB 6B EB B0 25 12 0C 77 33 EB F6 FF 12 B1 7C 0B DC D0 C6 47 5B 5B 55 EE 52 1B 93 71 88 4F
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes : B0 8C D2 EF FE 48 BF D2 AD DE 8B 79 2D 36 ED B2 4F CB DB E5 94 8E 96 A0 72 77 7C AD 70 CA CB 19 ED F3 FE 17 CD FA 44 DD 28 3E 60 7E 9A 2E 8D 4D EB 46 C0 A2 E7 B3 91 10 18 9B C9 F2 C4 35 5A C2 D7 E8 C0 90 9F 03 78 D4 28 E6 53 B7 C5 4D B1 FE 51 BD 61 27 8B 42 2A 65 DB 3B F4 7F DE 0B 4F 72 4C 5F 5F 93 B8 B4 B3 A5 0A 97 5D A4 07 ED 57 3B 1C 2B E0 85 BD C0 F8 E3 3C 66 0E B1 C7 B5 DC 1C F6 7E 65 80 B8 7A 07 0F 8A F1 1A 75 75 A9 0F 62 97 2B 14 A1 23 43 7D 96 41 0D C8 2B 7C 1E 71 72 B8 72 C2 DA 1B ED 4F 80 B0 65 CB FC BB 6B EA 87 40 88 3D 03 AD 50 44 72 22 04 B8 A0 2B D9 23 C7 06 D2 47 C3 B4 03 04 BE B8 AF 05 01 EE 1E 57 4B 4C EE 58 56 5B 99 54 86 BF D0 D1 3D B6 4B 2A 47 C0 11 6C D9 9E DB E3 AD 37 6E 75 B3 E6 98 09 81 93 FF 4C 2D 12 E5 4D 72 BA 53 09 CE 57 FF D7 0C

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - In 1996 a flaw was found in the design of MD5 ☹ -- "should be considered cryptographically broken and unsuitable for further use"
- SHA-2, SHA-3
  - 224, 256, 384 or 512 bits in digests

Certification for Public Key

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities
- DH

Public key problem:

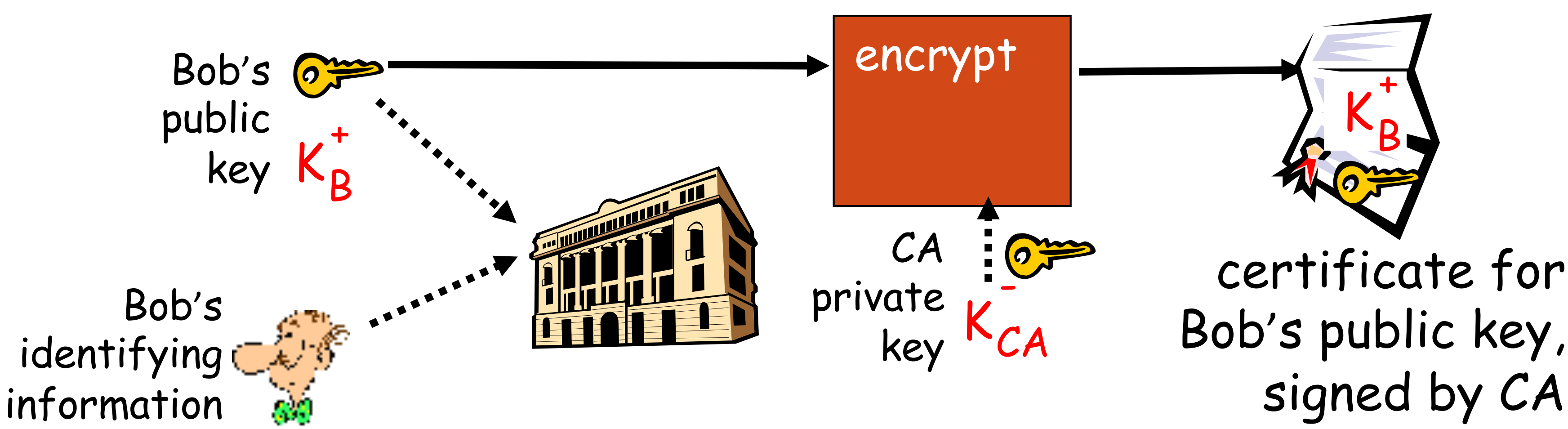
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- trusted certification authority (CA)

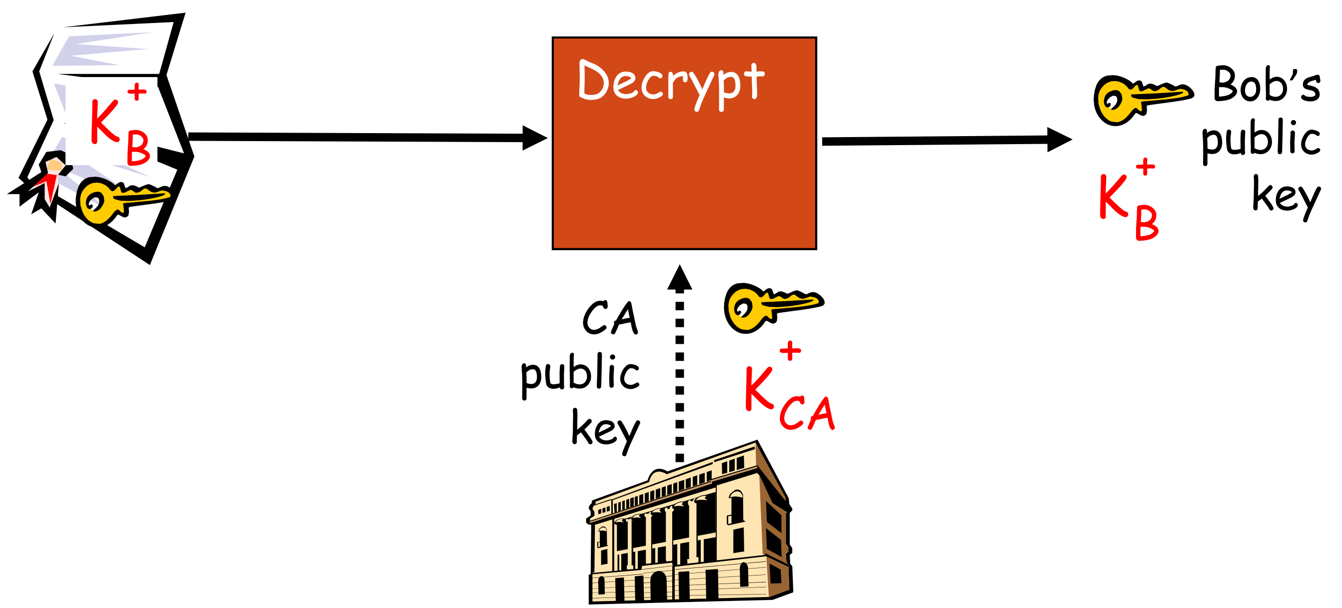
Certification Authorities

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



Certification Authorities

- 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
  - Agree or not?



What have we learned so far?

- Message confidentiality: shared key or public key crypto
- Message integrity: hash
- Authenticity of a digital message: digital signature

What about authenticity of sender/receiver?

- ARP poisoning
- IP/MAC address spoofing
- phishing attacks

Need authentication

Authentication

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

Protocol: assume pre-shared secret between Alice and Bob

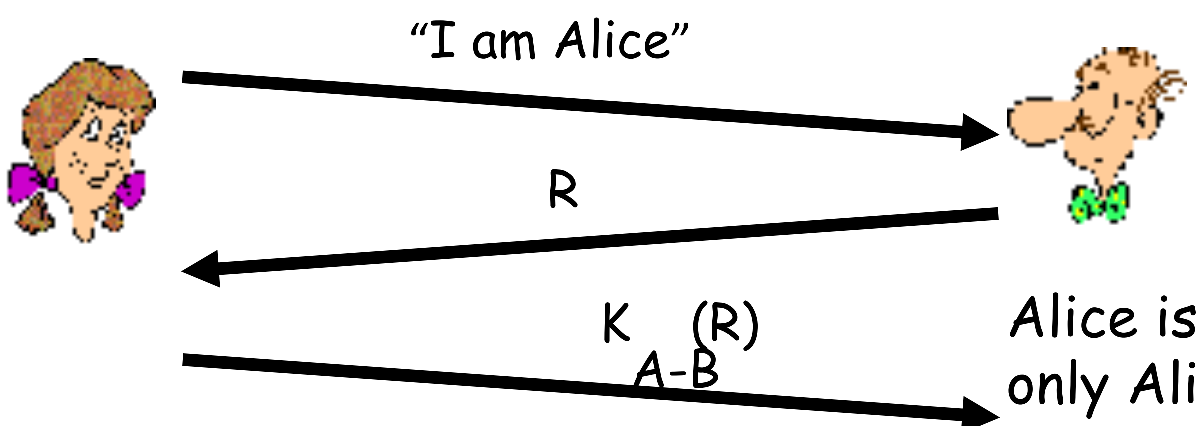
Failure scenario??

Authentication: Symmetric Key Crypto

Goal: avoid IP proofing, playback attack

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

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



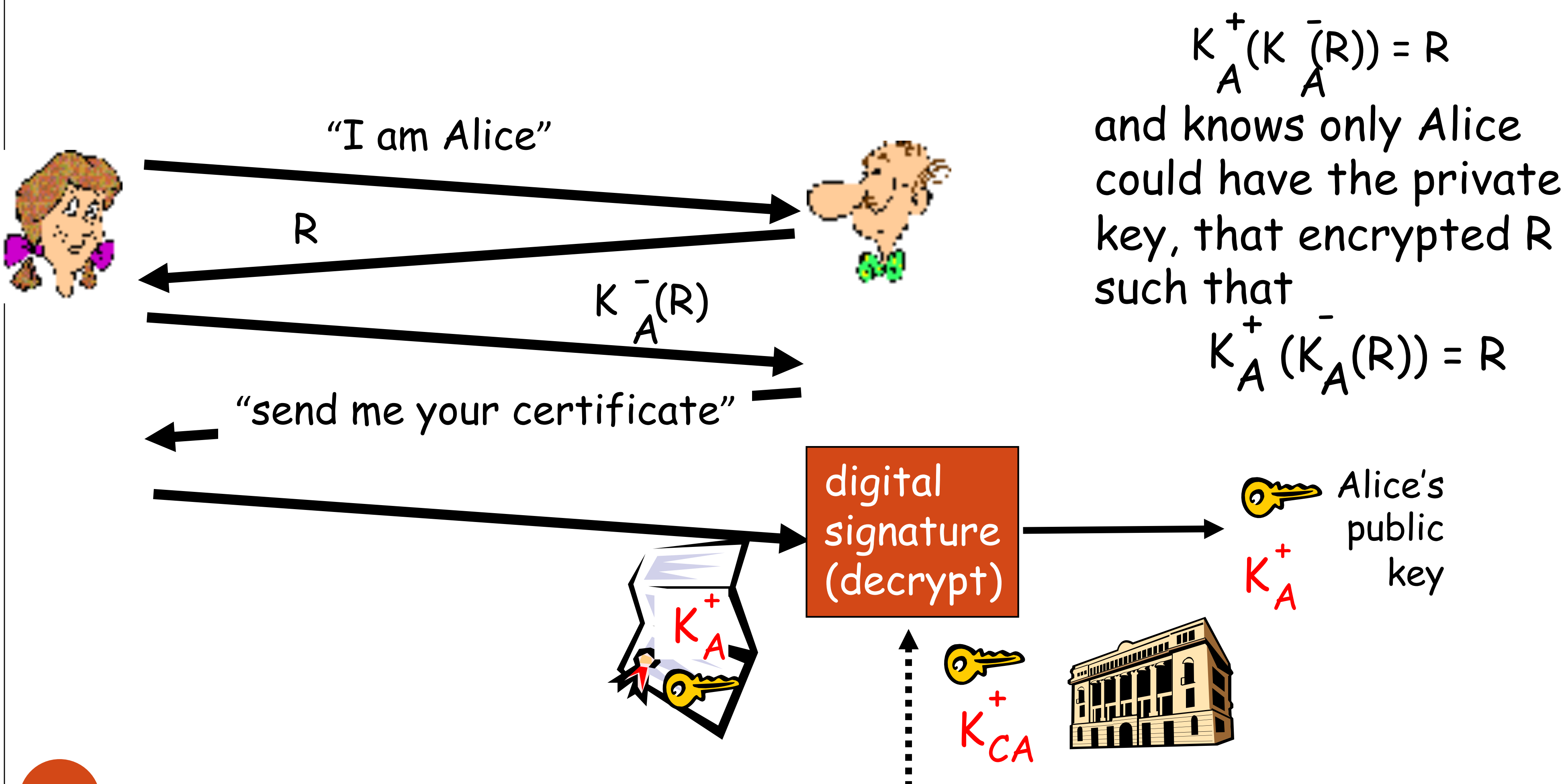
Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Failures, drawbacks?

Authentication: Public Key Crypto

- can we authenticate using public key techniques?

use nonce, public key cryptography

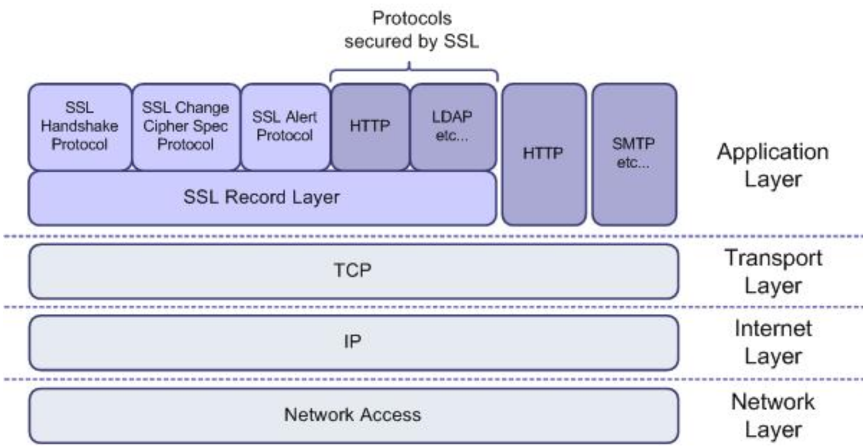


Outline

- Attacks and counter measures
- Security primer
- Security protocols
  - SSL
  - 802.11i
  - IPsec VPN

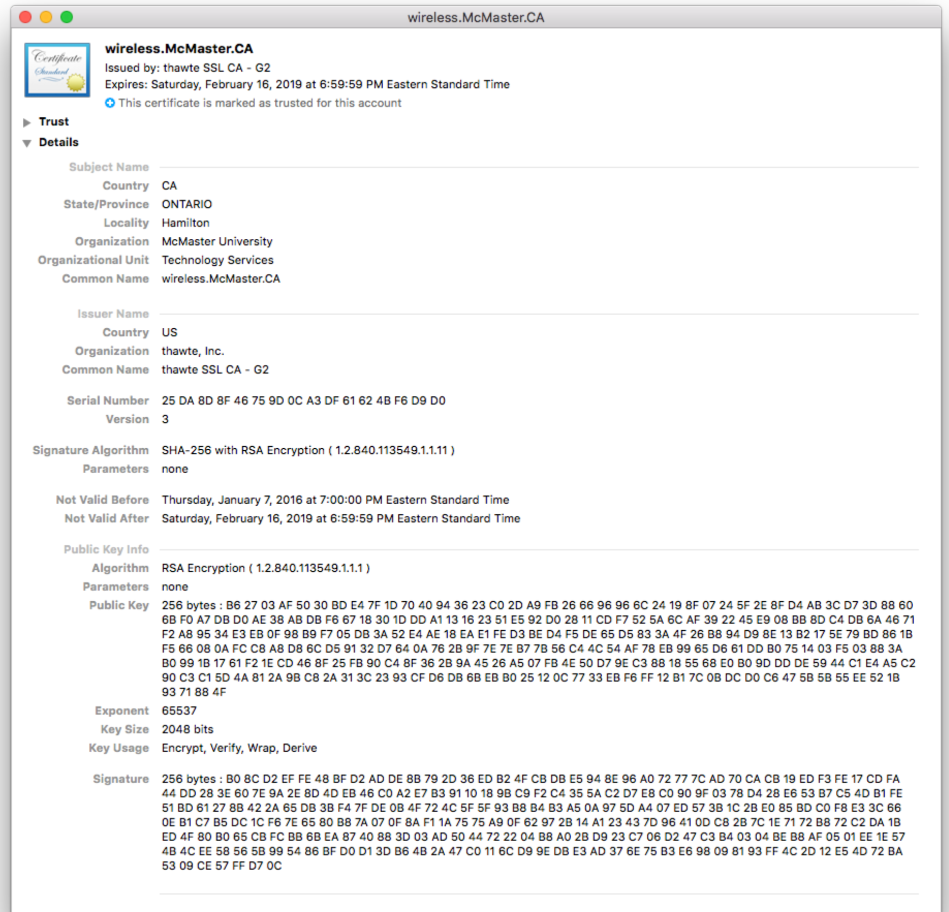
Secure sockets layer (SSL)

- transport layer security to any TCP-based application using SSL services.
- used between Web browsers, servers for e-commerce (https).
  - SSL can be used for non-Web applications, e.g., IMAP.
- SSL: basis of IETF Transport Layer Security (TLS).
- security services:
  - server authentication
  - data encryption
  - data integrity
  - client authentication (optional)



SSL (cont'd)

- server authentication:
  - SSL-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to extract server's public key from certificate.
- check your browser's security menu to see its trusted CAs.



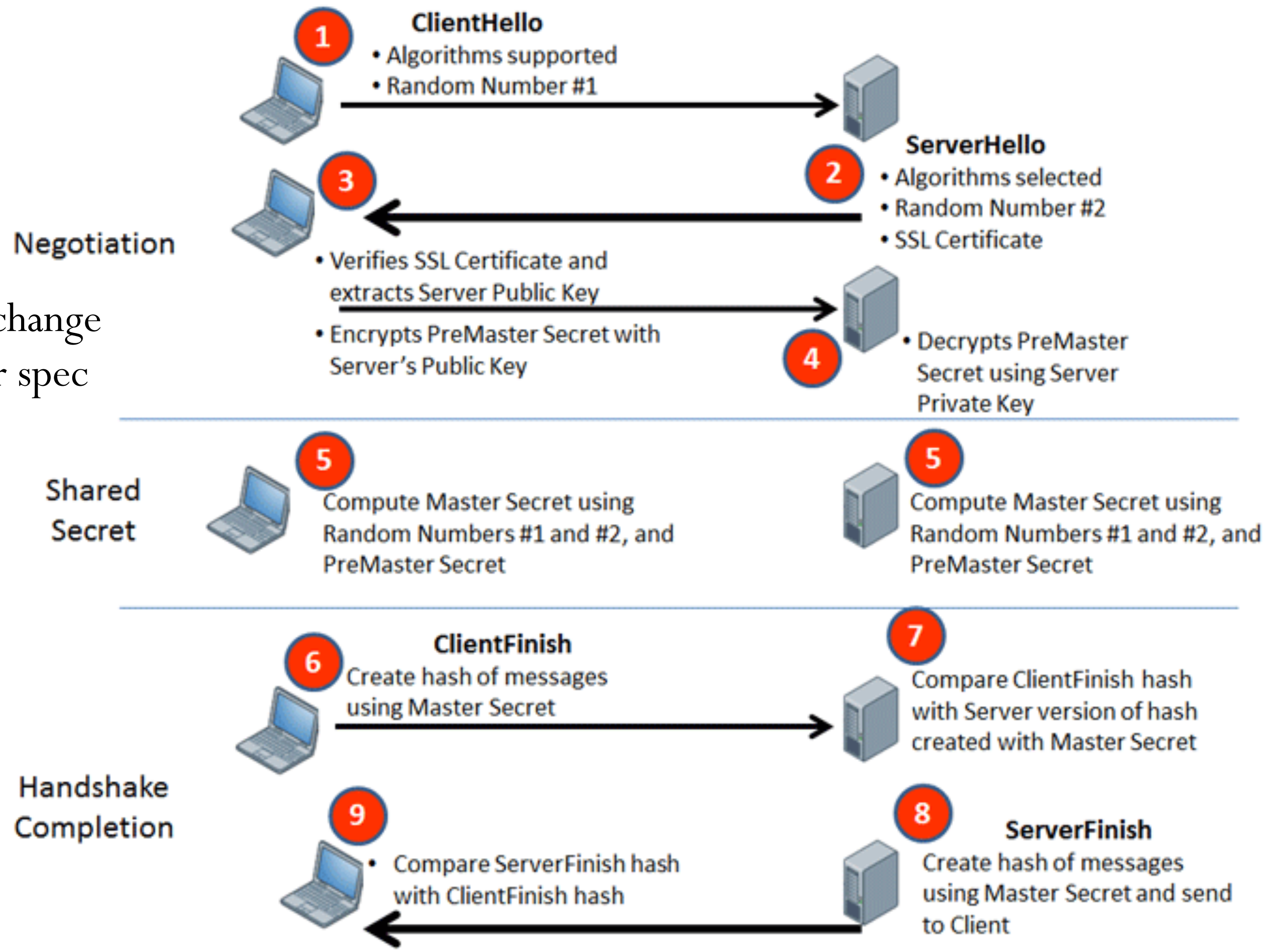
SSL (continued)

Encrypted SSL session:

- Browser generates *symmetric session key*, encrypts it with server's public key, sends encrypted key to server.
- Using private key, server decrypts session key.
- Browser, server know session key
  - All data sent into TCP socket (by client or server) encrypted with session key.
- Client authentication can be done with client certificates.

SSL + RSA

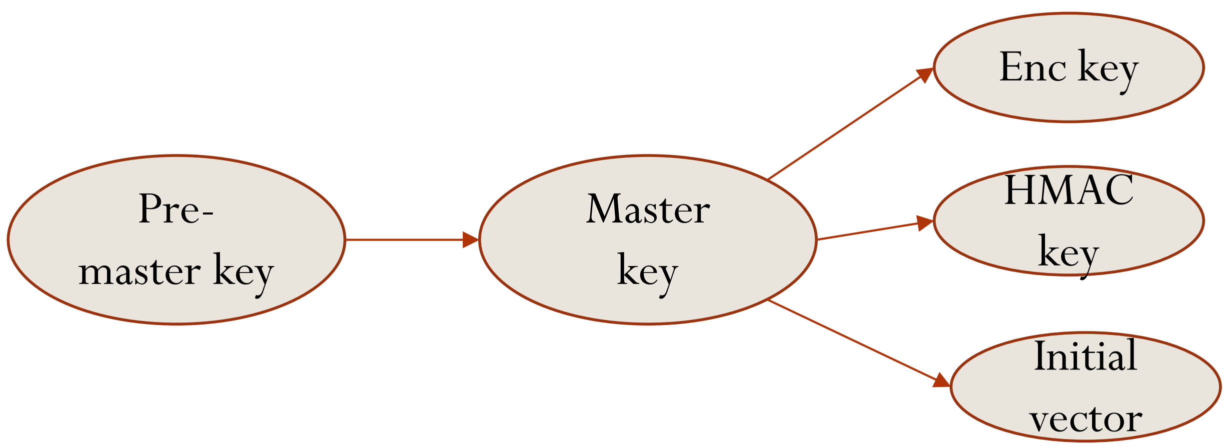
Client key exchange
Change cipher spec



SSL Cipher Suite

Key Exchange	Authentication	Cipher	Hash
RSA Diffie-Hellman …	RSA DSA	3DES AES …	MD5 SHA …

ECDHE_RSA_WITH_AES_256_GCM_SHA385
RSA_WITH_AES_128_CBC_SHA256



IEEE 802.11 security

- *War-driving:* drive around Bay area, see what 802.11 networks available?
  - More than 9000 accessible from public roadways
  - 85% use no encryption/authentication
  - packet-sniffing and various attacks easy!
- Securing 802.11
  - encryption, authentication
  - first attempt at 802.11 security: Wired Equivalent Privacy (WEP): a failure
  - current attempt: 802.11i

802.11 Security Overview

Open Authentication

Preshared-key Authentication

Small business/home

802.1x Authentication

Large enterprise

Authentication and access control

Wired Equivalent Privacy (WEP)

WPA/ WPA2 Personal

WPA2

Data encryption and MIC

Catastrophic failure!

Dominating protocol

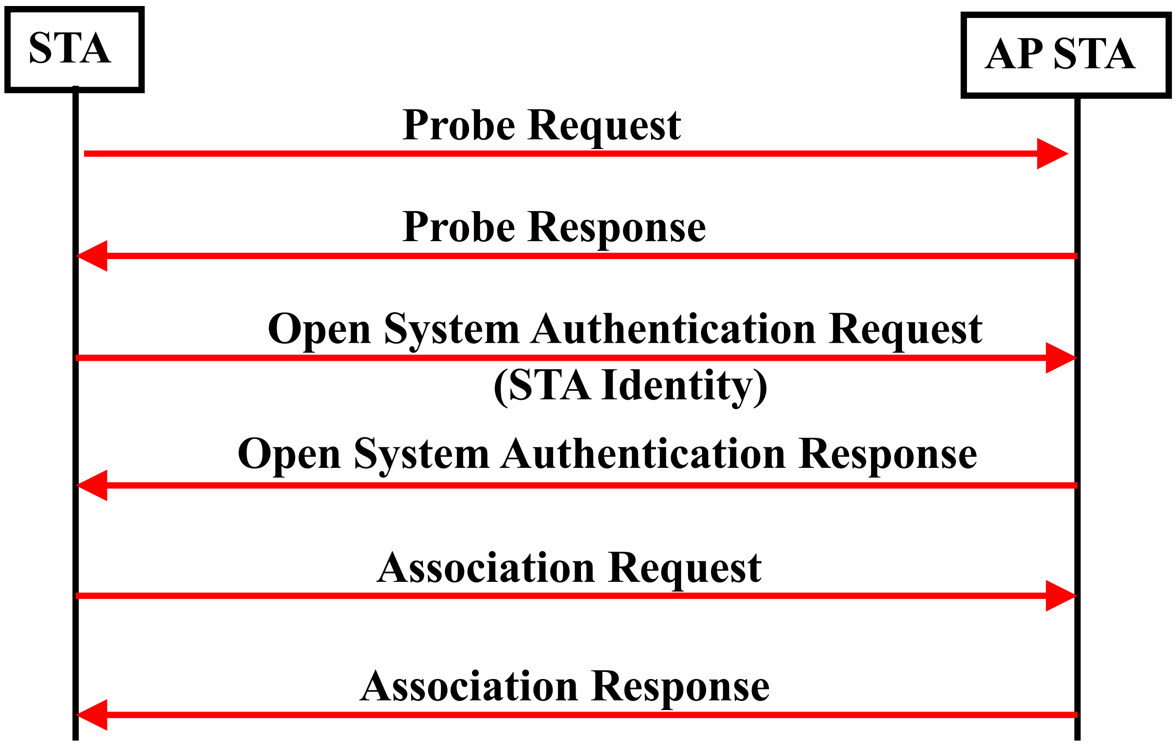
RC4

TKIP

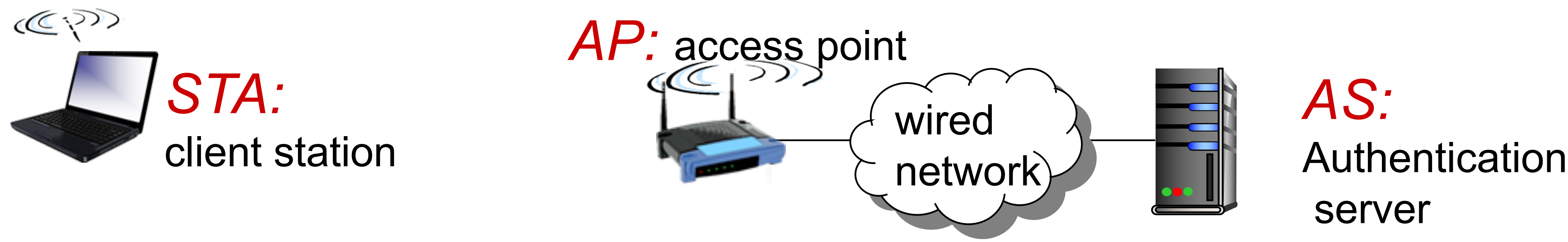
AES

Open System Authentication

- Establishing the IEEE 802.11 association with no authentication



802.11i: four phases of operation



1 Discovery of security capabilities

2 STA and AS mutually authenticate, together generate Master Key (MK). *AP serves as "pass through"*

3 STA derives Pairwise Master Key (PMK)

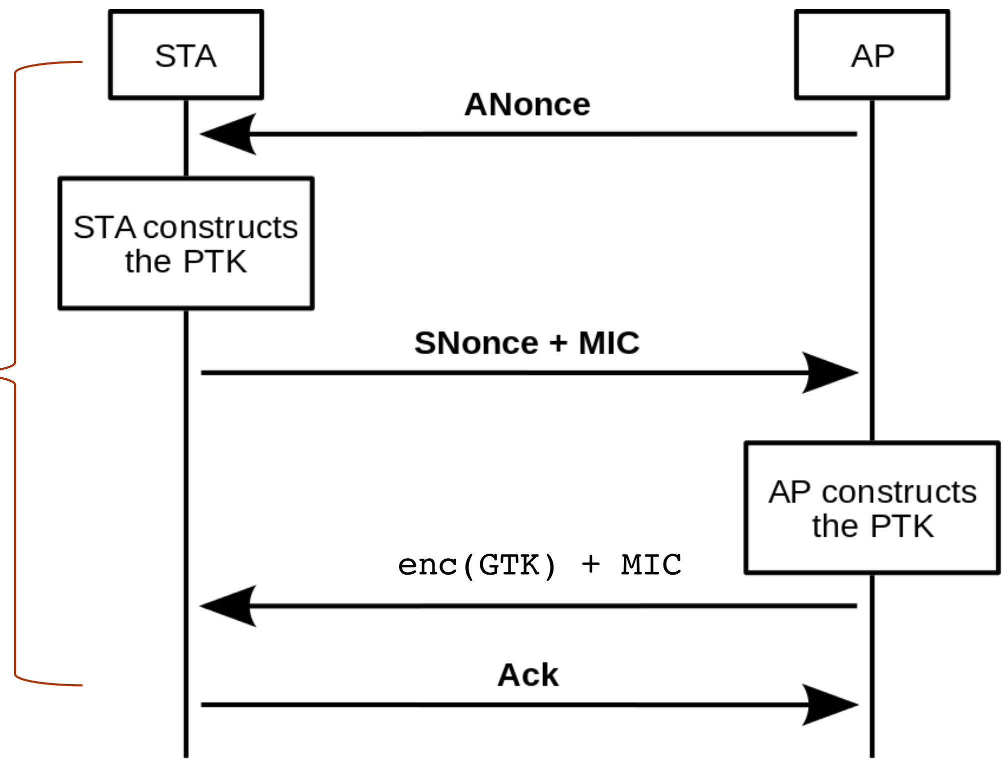
3 AS derives same PMK, sends to AP

4 STA, AP use PMK to derive Temporal Key (PTK) used for message encryption, integrity

Pre-shared Key (PSK) Authentication

- Uses a passphase to generate encryption key
- *PMK = PBKDF2(PassPhrase, ssid, ssidLength, 4096, 256)*
- *PTK = PRF512(*PMK, AMAC, SMAC, ANonce, SNonce)

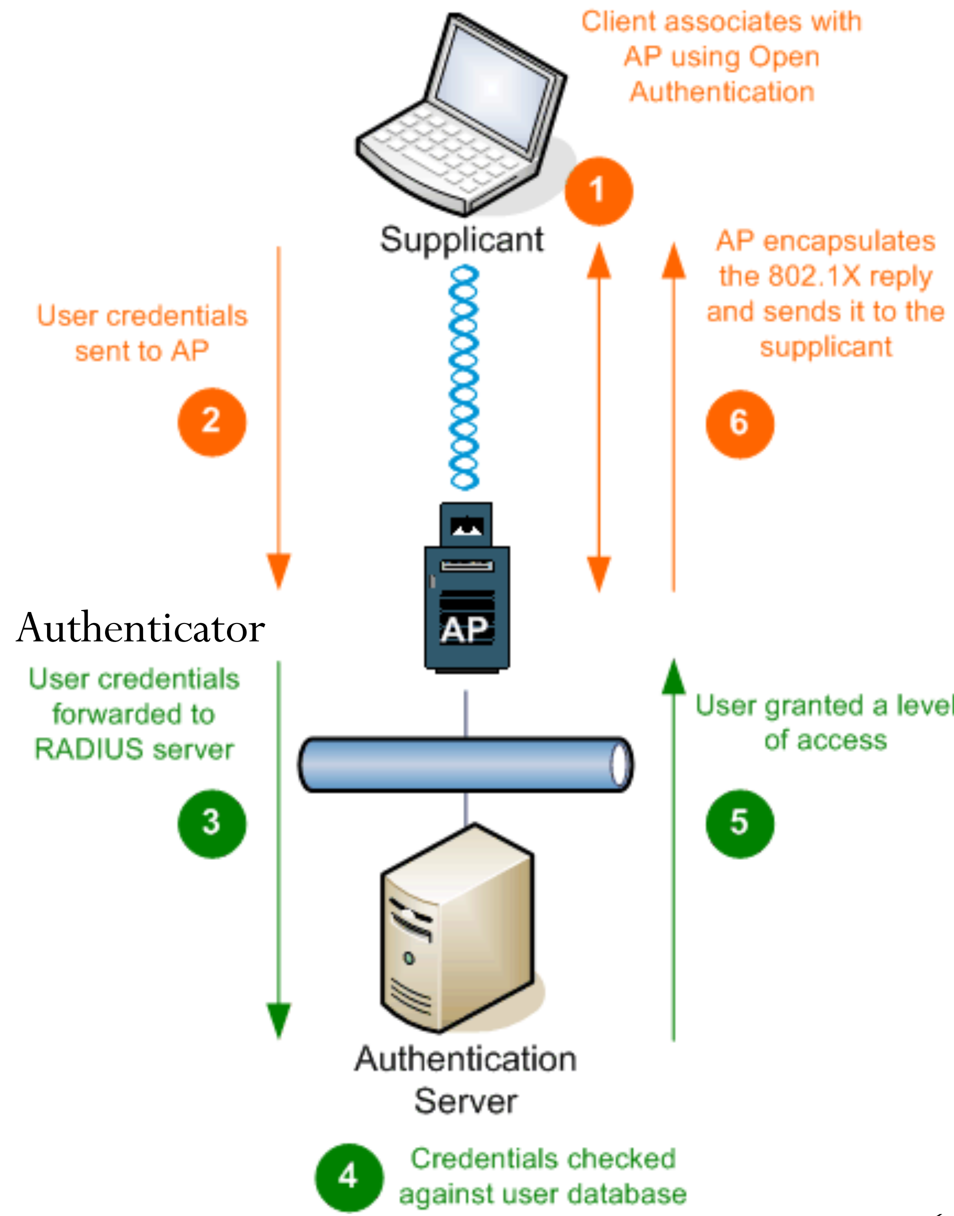
4-way
Handshake



PTK – Pairwise temporary key
MIC -- Message integrity check
GTK – Group template key

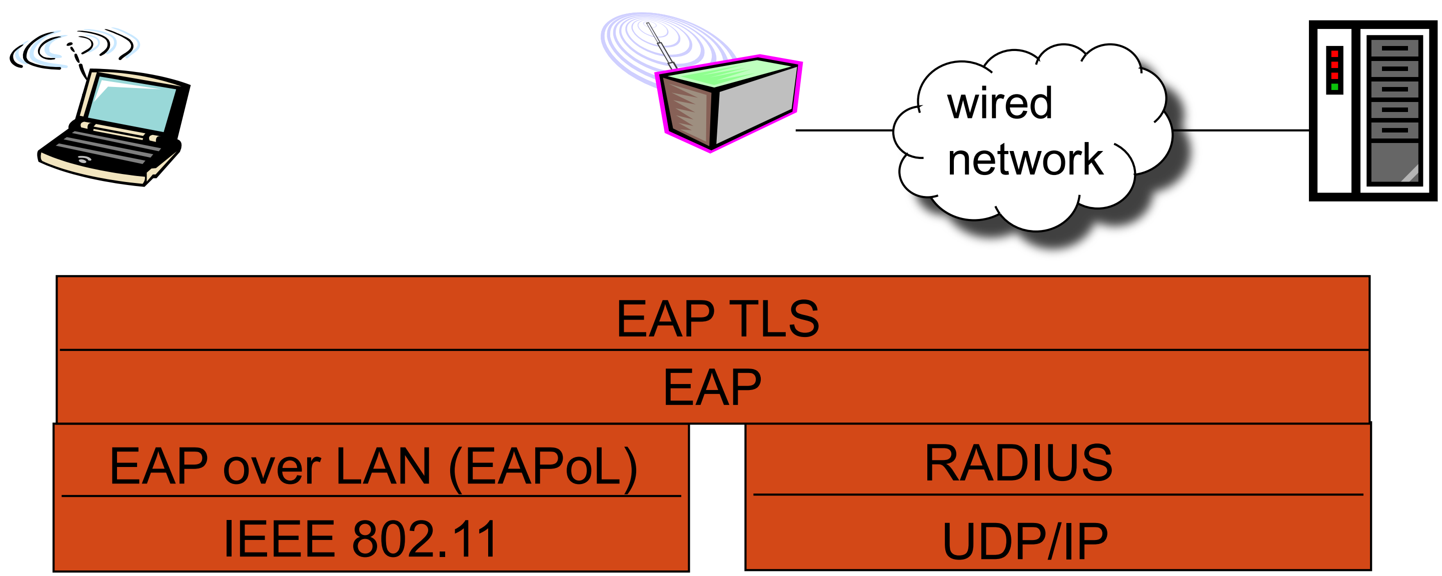
802.1x Authentication

- An IEEE standard for port-based network access control
- Provide authentication for devices connected via LAN or WLAN
- RADIUS (Remote Authentication Dial-In User Service)

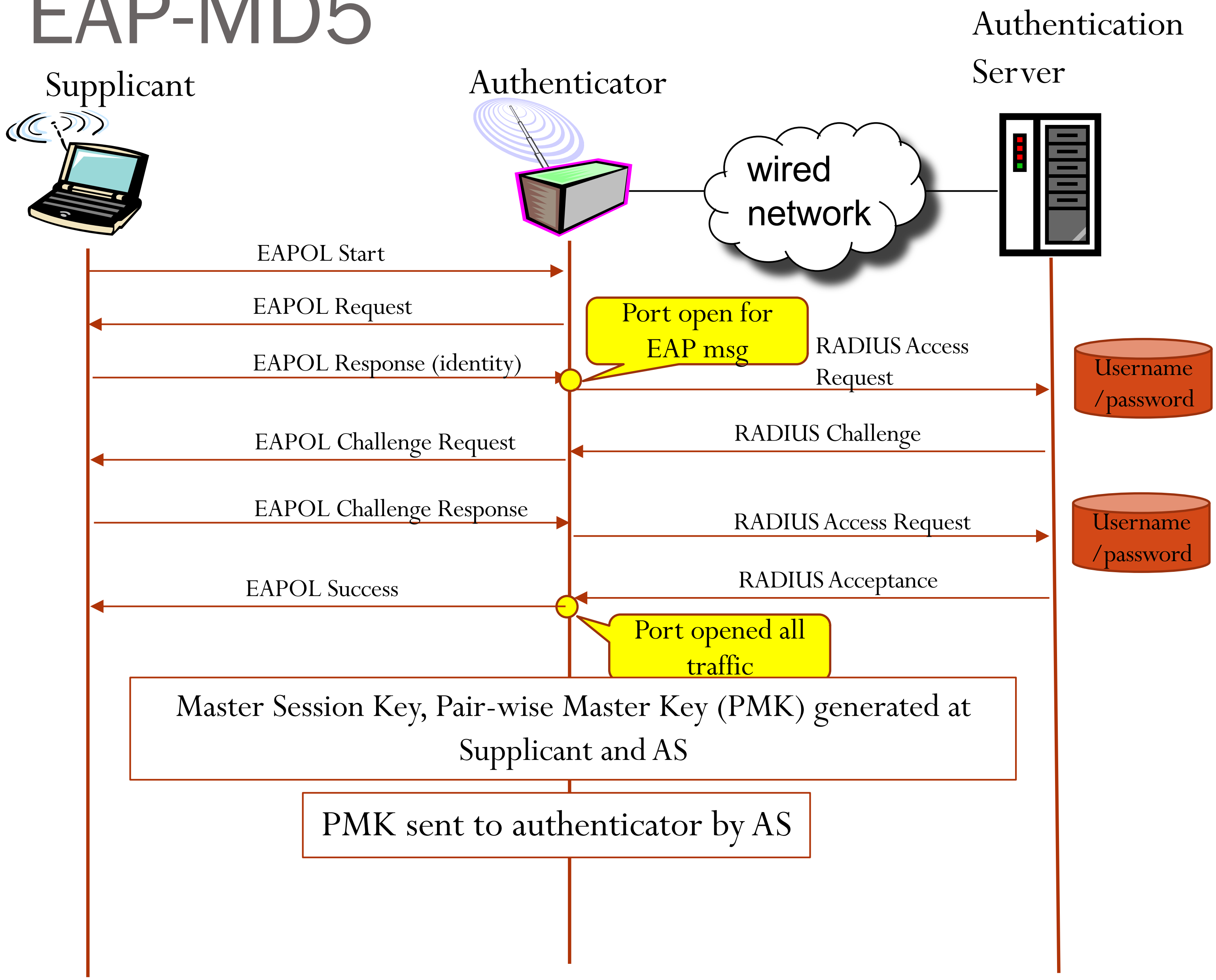


EAP: extensible authentication protocol

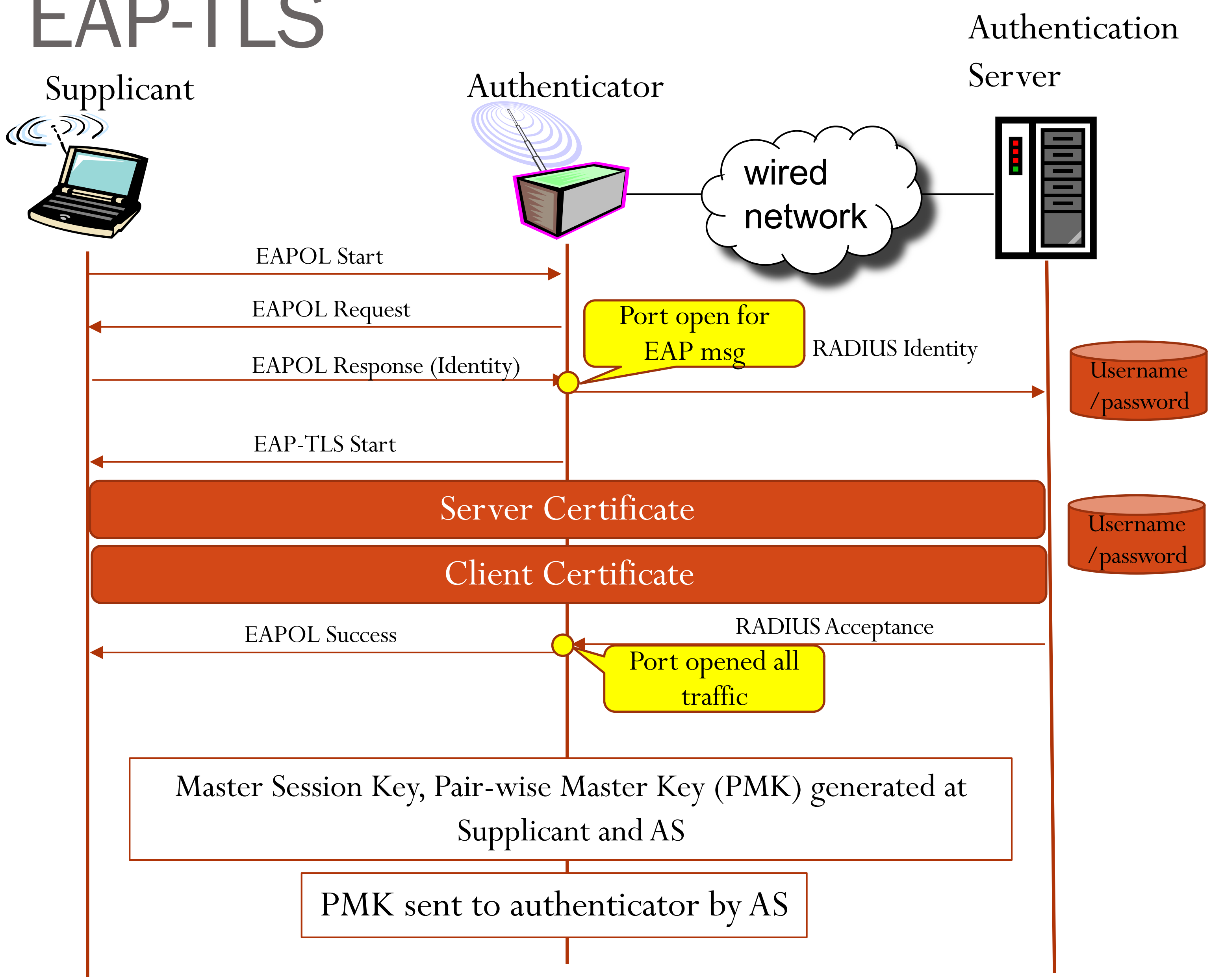
- EAP: end-end client (mobile) to authentication server protocol
  - Originally an extension of point-to-point protocol for dial-ups
- EAP sent over separate "links"
  - mobile-to-AP (EAP over LAN)
  - AP to authentication server (RADIUS over UDP)
- Support different authentication methods: MD5, TLS, PEAP ..



EAP-MD5

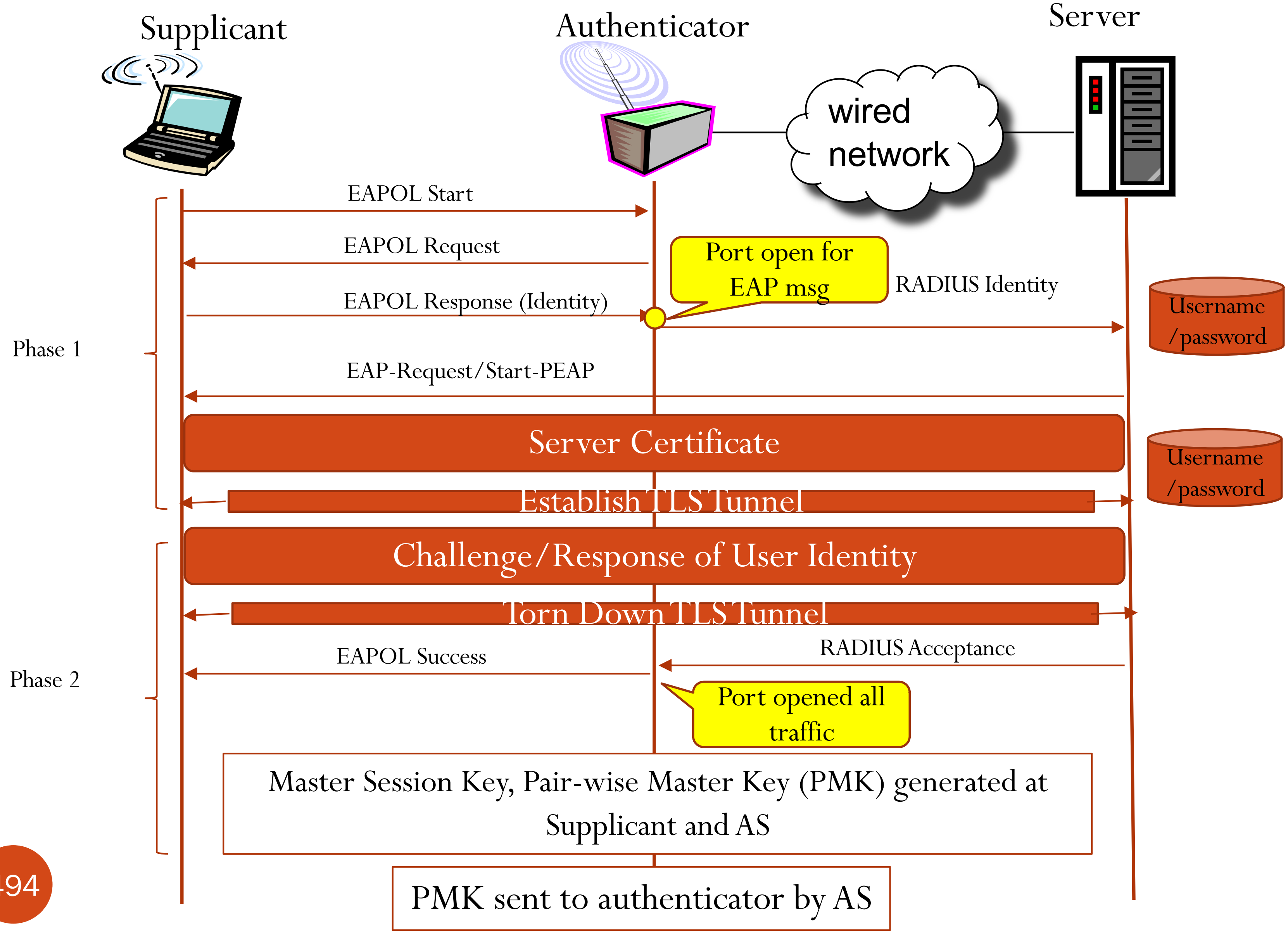

Supplicant
Authenticator
Authentication Server
wired network
EAPOL Start
EAPOL Request
Port open for EAP msg
EAPOL Response (identity)
RADIUS Access Request
Username /password
EAPOL Challenge Request
RADIUS Challenge
EAPOL Challenge Response
RADIUS Access Request
Username /password
EAPOL Success
RADIUS Acceptance
Port opened all traffic
Master Session Key, Pair-wise Master Key (PMK) generated at Supplicant and AS
PMK sent to authenticator by AS

EAP-TLS



More at https://tools.ietf.org/html/rfc5216

EAP-PEAP


Supplicant
Authenticator
Authentication Server
wired network
EAPOL Start
EAPOL Request
EAPOL Response (Identity)
Port open for EAP msg
RADIUS Identity
Username /password
EAP-Request/Start-PEAP
Phase 1
Server Certificate
Username /password
Establish TLS Tunnel
Challenge/Response of User Identity
Torn Down TLS Tunnel
Phase 2
EAPOL Success
RADIUS Acceptance
Port opened all traffic
Master Session Key, Pair-wise Master Key (PMK) generated at Supplicant and AS
PMK sent to authenticator by AS

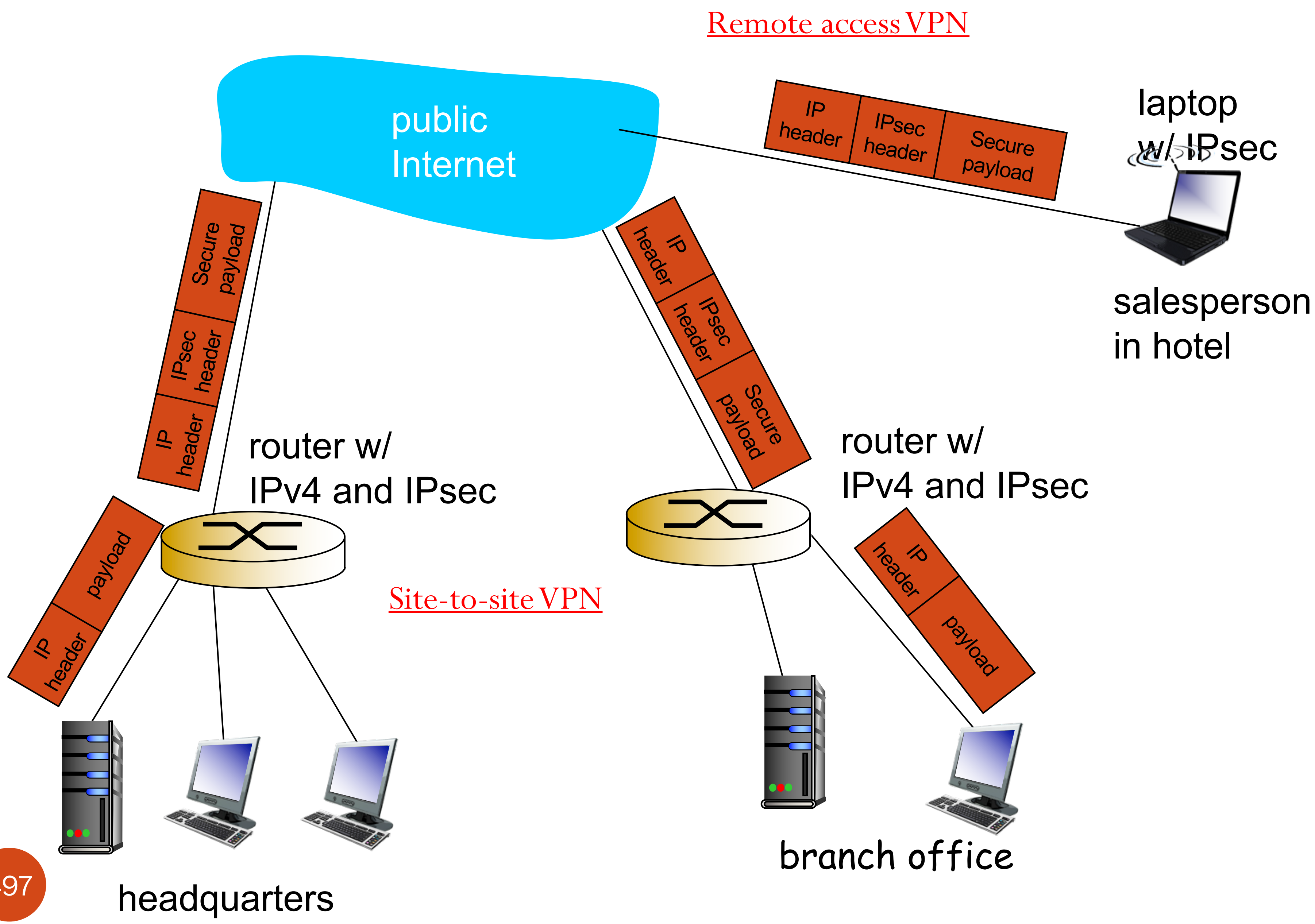
What is network-layer confidentiality ?

- between two network entities: sending entity encrypts datagram payload, payload could be:
  - TCP or UDP segment, ICMP message, OSPF message ….
- all data sent from one entity to other would be hidden:
  - web pages, e-mail, P2P file transfers, TCP SYN packets …
- "blanket coverage"

Virtual Private Networks (VPNs)

- institutions often want private networks for security.
  - costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet instead
  - encrypted before entering public Internet
  - logically separate from other traffic

Virtual Private Networks (VPNs)

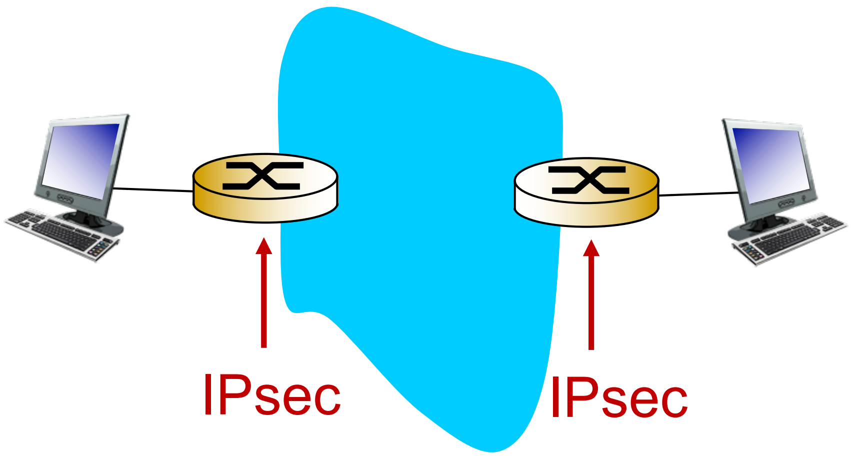


IPsec services

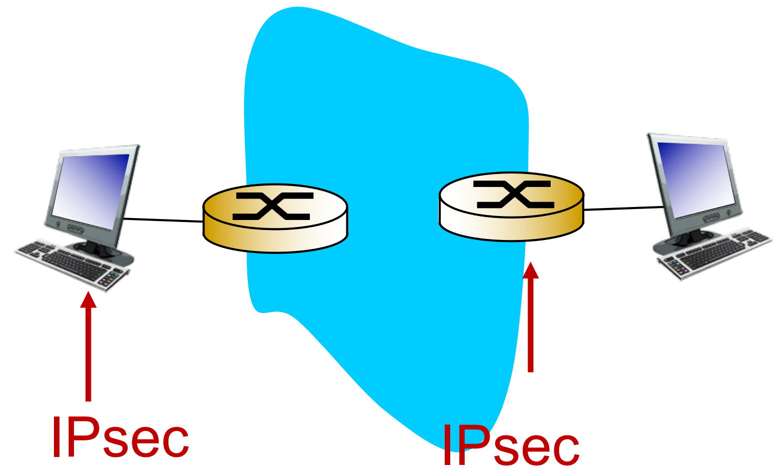
- data integrity
- origin authentication
- replay attack prevention
- confidentiality

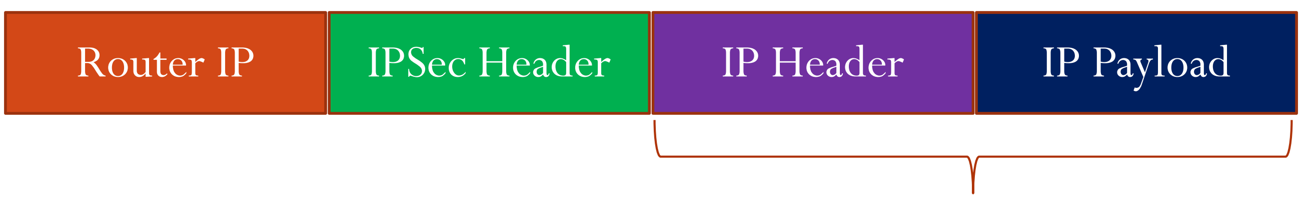
IPsec – tunneling mode

- edge routers IPsec-aware



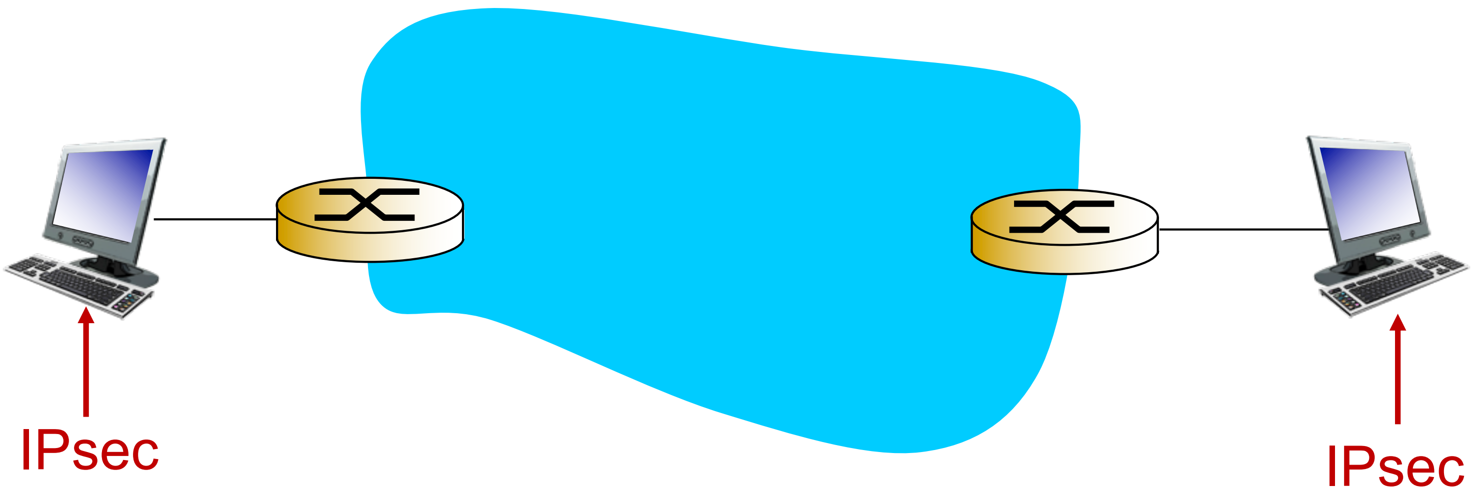
- hosts IPsec-aware





IPsec transport mode

- IPsec datagram emitted and received by end-system
- protects upper level protocols



IP Header	IPSec Header	IP Payload

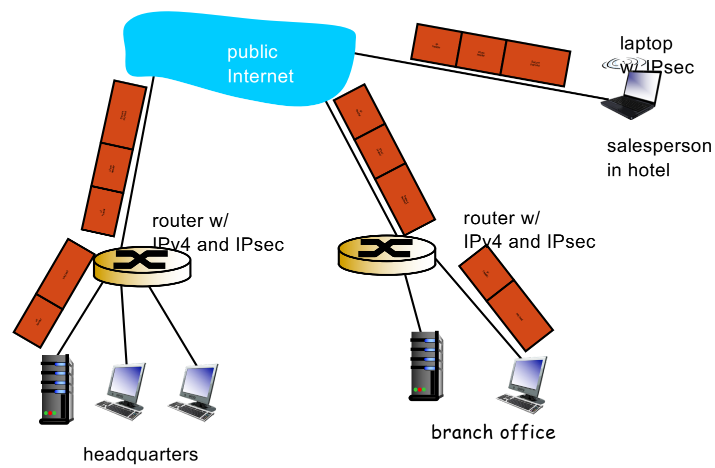
IPsec protocols

- IKE/IKEv2: provides a framework for policy negotiation and key management
  - Security associations (SAs)
- Authentication Header (AH) protocol
  - provides source authentication & data integrity but not confidentiality
- Encapsulation Security Protocol (ESP)
  - provides source authentication, data integrity, and confidentiality
  - more widely used than AH

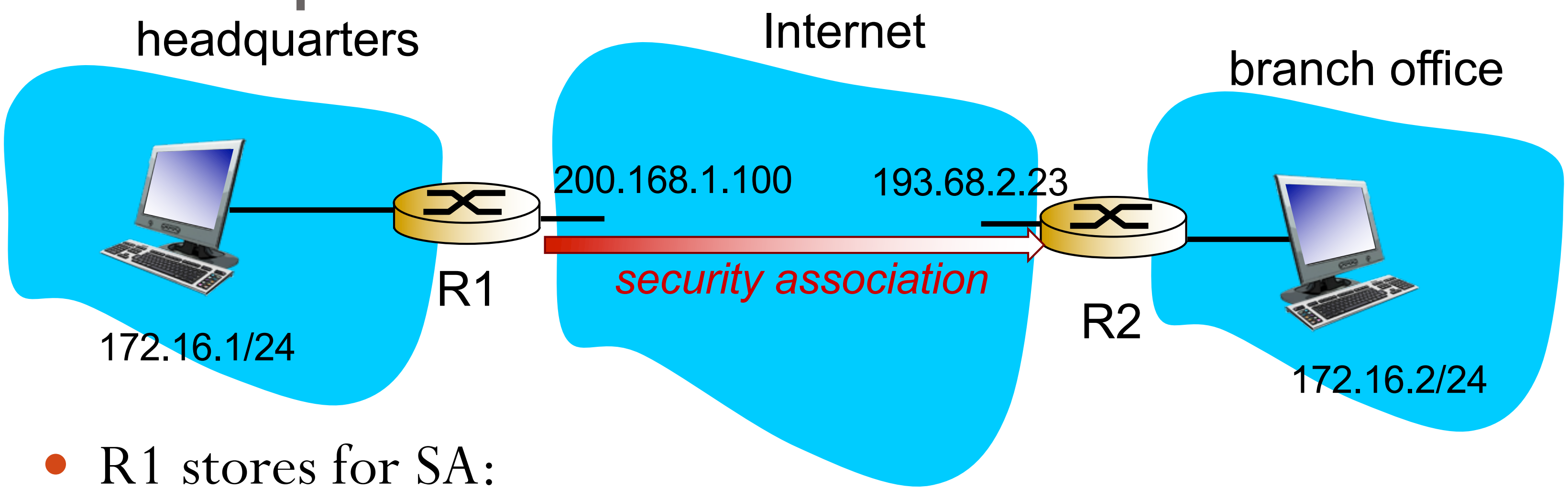
Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

Security associations (SAs)

- before sending data, "security association (SA)" established from sending to receiving entity
  - SAs are simplex: for only one direction
- ending, receiving entitles maintain state information about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!
- how many SAs in VPN w/ headquarters, branch office, and n traveling salespeople?

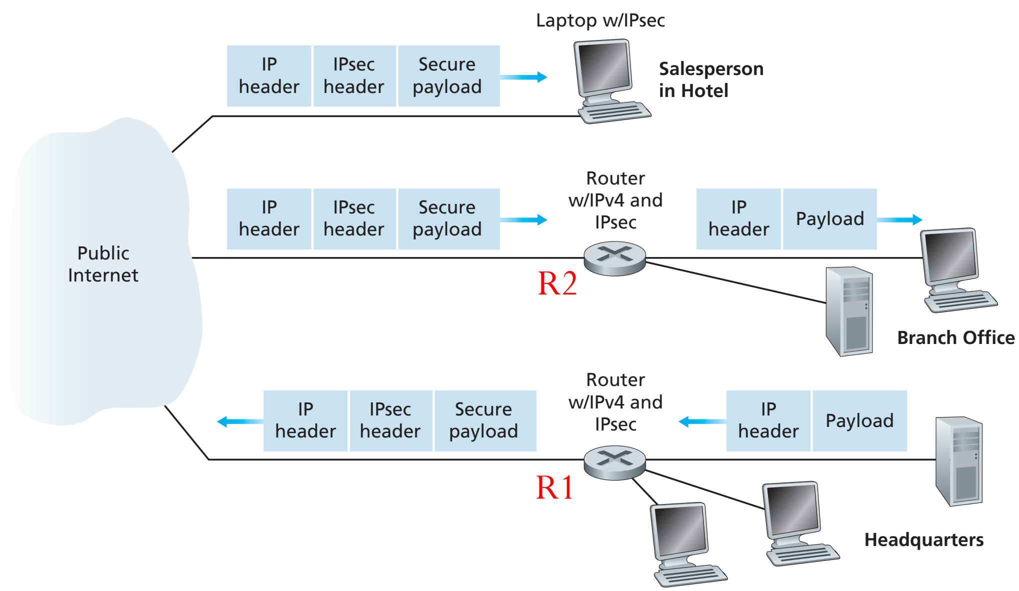


Example SA from R1 to R2



- R1 stores for SA:
  - 32-bit SA identifier: Security Parameter Index (SPI)
  - origin SA interface (200.168.1.100)
  - destination SA interface (193.68.2.23)
  - type of encryption used (e.g., 3DES with CBC)
  - encryption key
  - type of integrity check used (e.g., HMAC with MD5)
  - authentication key

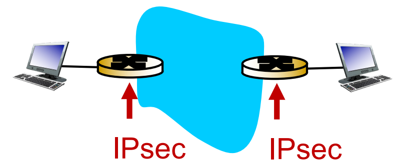
Security Association Database (SAD)

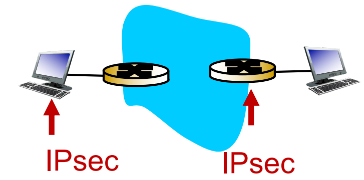


Assume

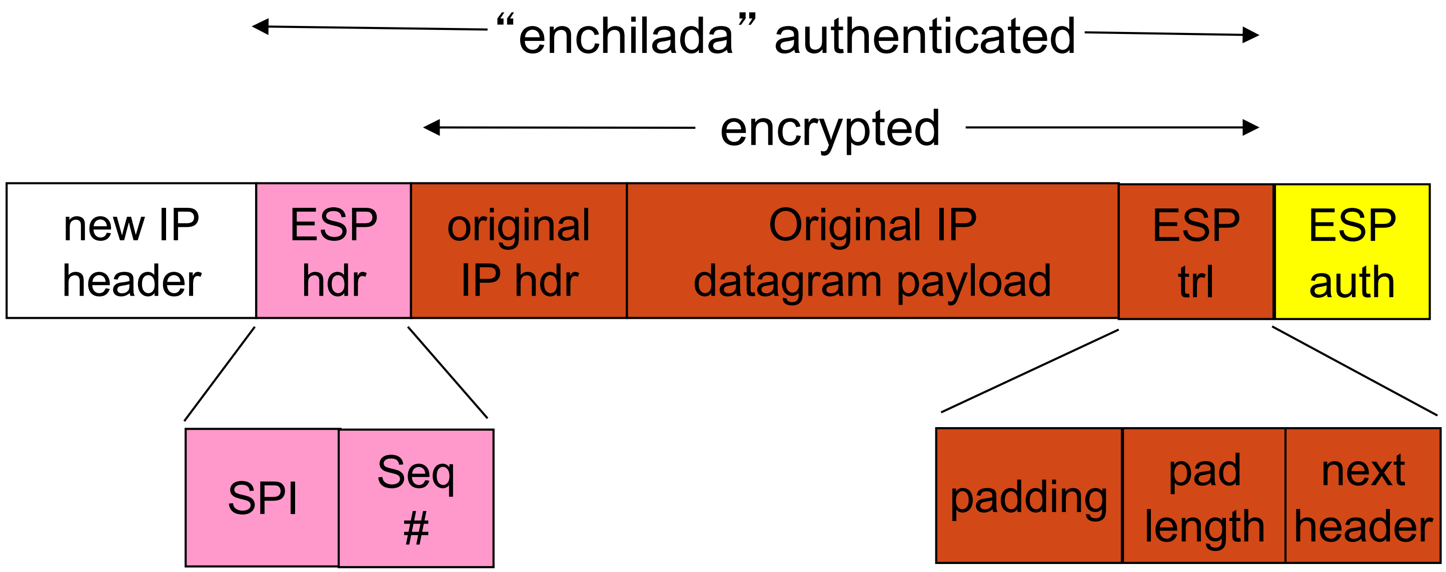
- endpoint holds SA state in security association database (SAD), where it can locate them during processing.
- One bi-directional IPsec traffic between headquarters and the branch office
  - 2 SAs
- One bi-directional IPsec traffic between headquarters and each salesperson
  - with n salespersons, 2n SAs in R1's 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, and processes datagram accordingly.

IPsec datagram

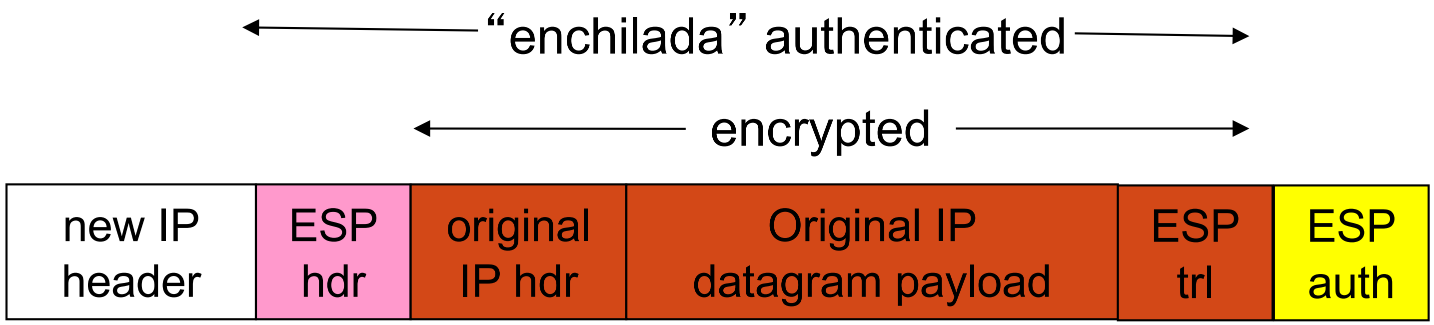




- focus for now on tunnel mode with ESP

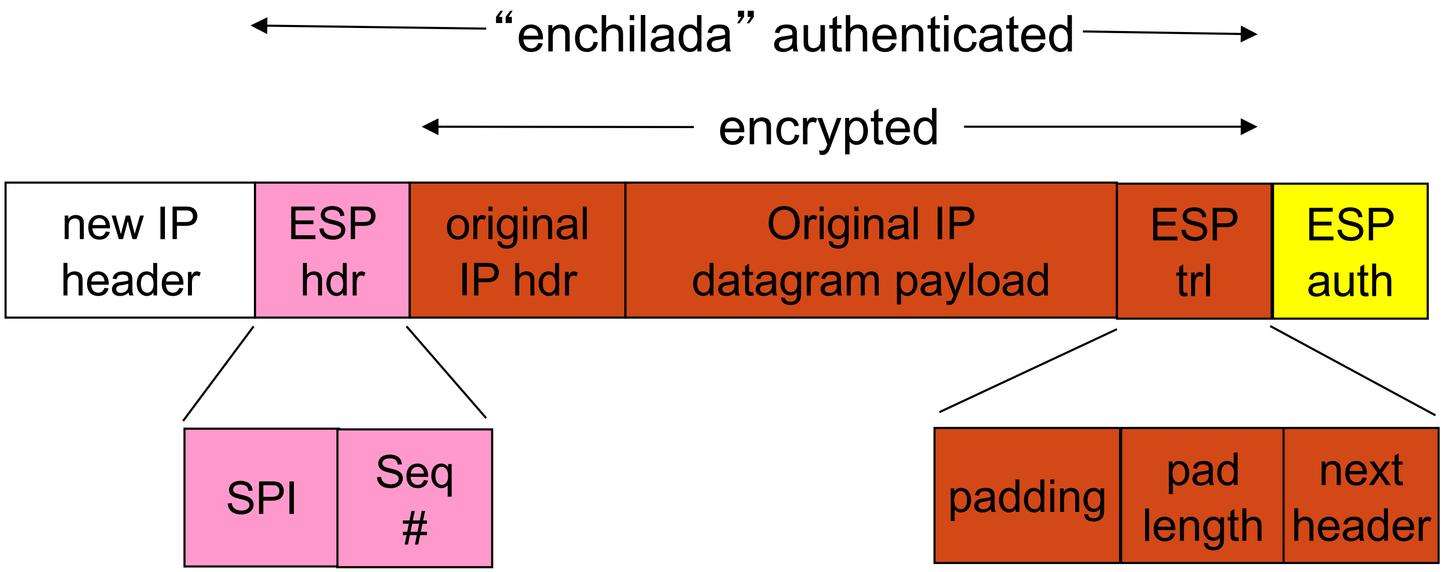


R1: convert original datagram to IPsec datagram



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

Inside the enchilada:



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

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

Security Policy Database (SPD)

- policy: For a given datagram, sending entity needs to know if it should use IPsec
- needs also to know which SA to use
  - may use: source and destination IP address; protocol number
- info in SPD indicates "what" to do with arriving datagram
- info in SAD indicates "how" to do it