Internetværk og Web-programmering
Security in computer networks

Lecture 13

Michele Albano

Distributed, Embedded, Intelligent Systems

DEIS

Agenda

8.1 What is network security?

8.2 Principles of cryptography

8.4 Authentication

8.3 Message integrity

8.6 Securing TCP connections: SSL

8.7 Network layer security: IPsec

What is Network Security?

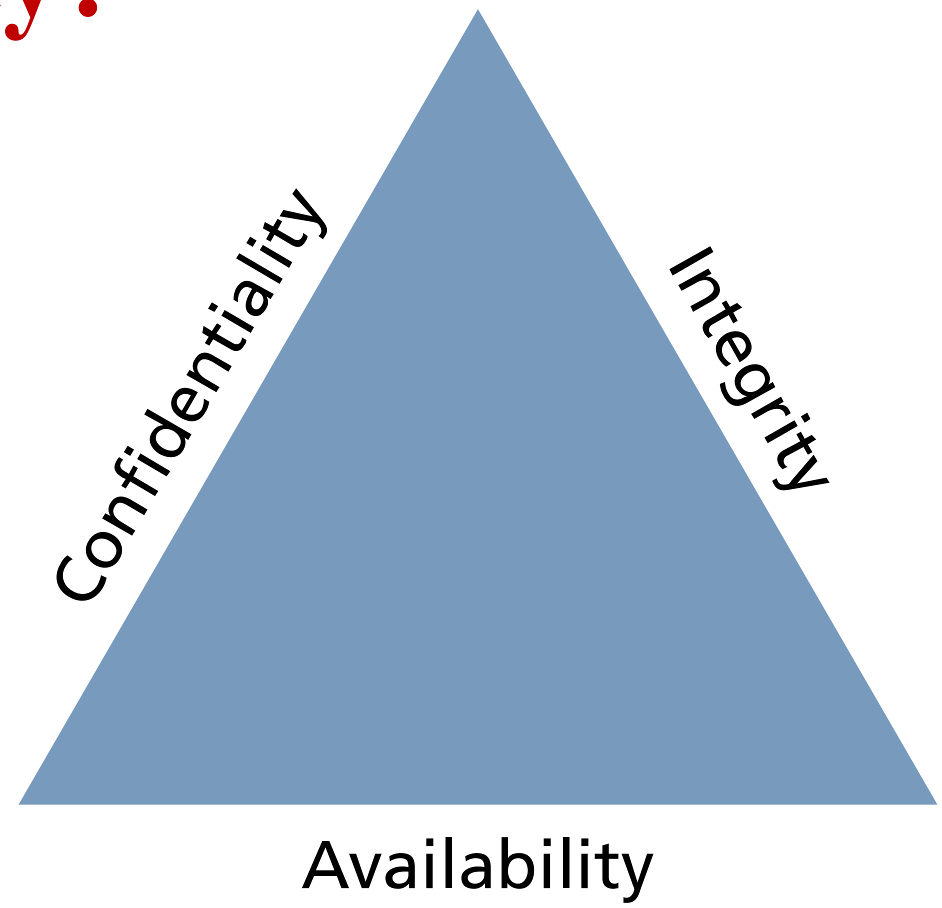
confidentiality: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

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

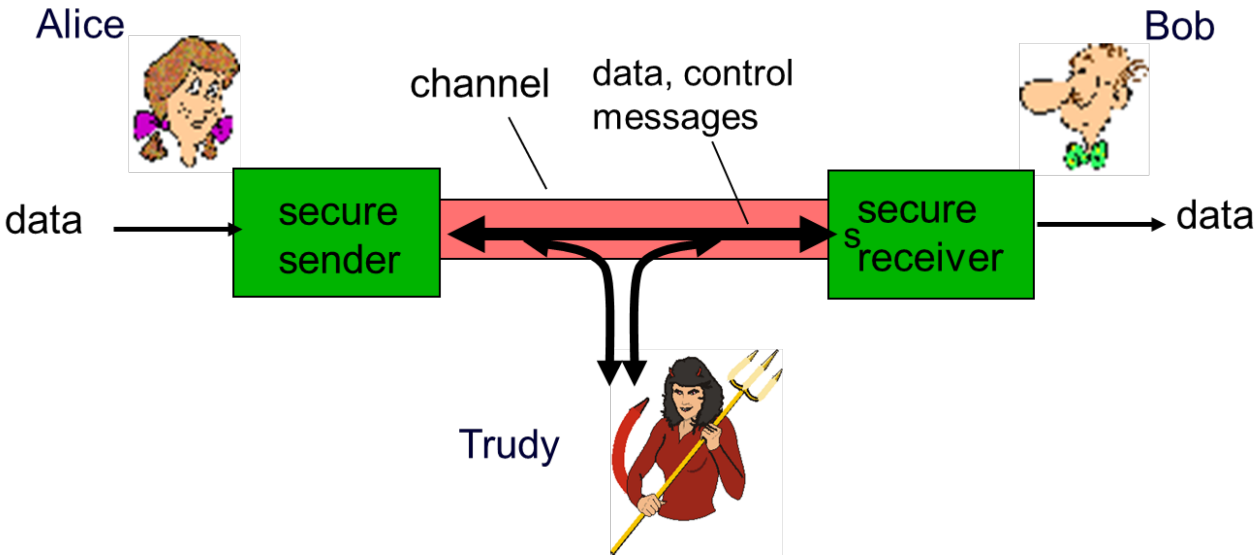
message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

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



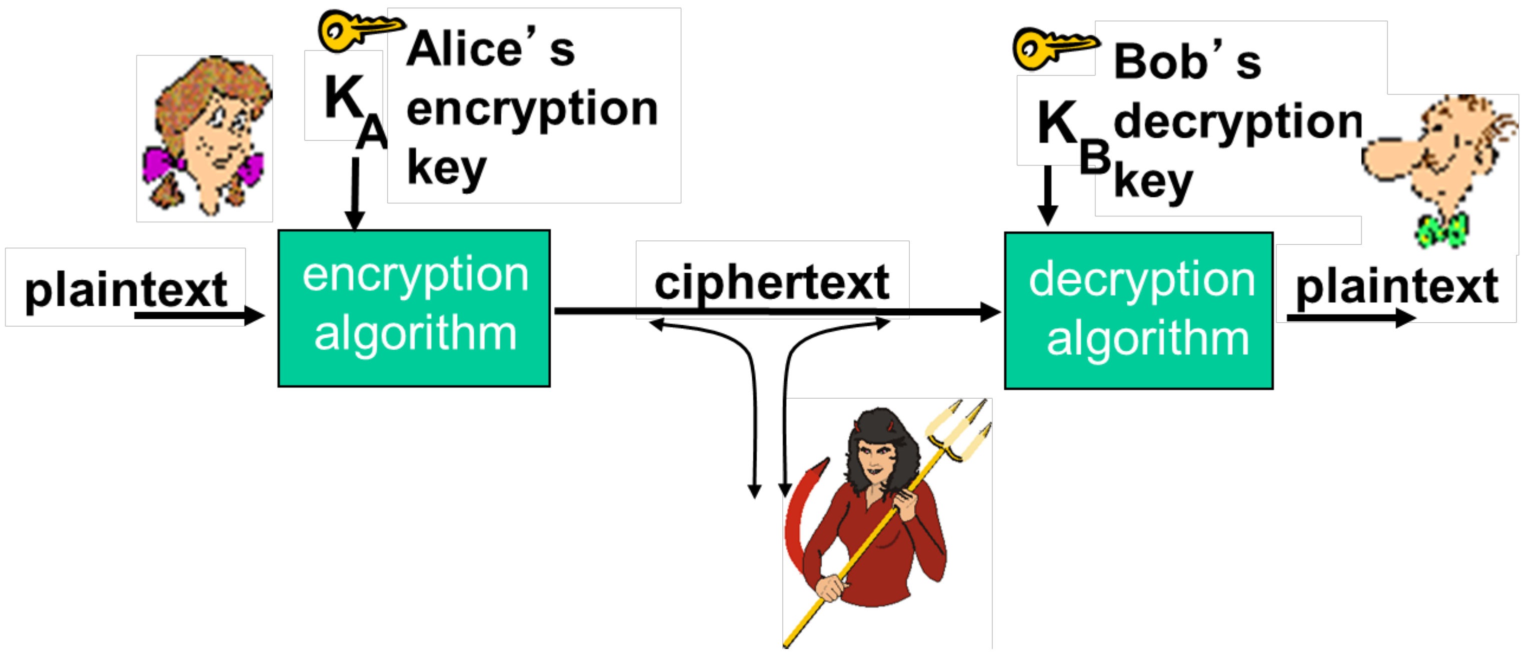
What can a “bad person” do?

Broadly, threats can be categorized using the STRIDE mnemonic

- **Spoofing**—The attacker uses someone else’s information to access the system.
- **Tampering**—The attacker modifies some data in nonauthorized ways.
- **Repudiation**—The attacker removes all trace of their attack, so that they cannot be held accountable for other damages done.
- **Information disclosure**—The attacker accesses data they should not be able to.
- **Denial of service**—The attacker prevents real users from accessing the systems.
- **Elevation of privilege**—The attacker increases their privileges on the system thereby getting access to things they are not authorized to do.

Agenda

The Language of Cryptography



m plaintext message

$\mathbf{K_A(m)}$ ciphertext, encrypted with key K_A

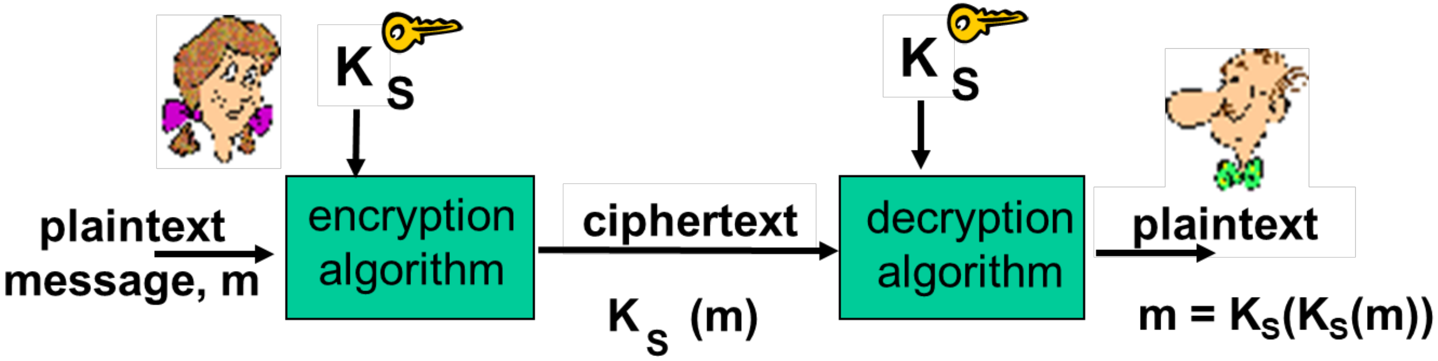
$$\mathbf{m = K_B(K_A(m))}$$

Breaking an Encryption Scheme

- **cipher-text only attack:** Trudy has ciphertext she can analyze
- **two approaches:**
  - brute force: search through all keys
  - statistical analysis

- **known-plaintext attack:** Trudy has plaintext corresponding to ciphertext
  - Trudy knows both m and $K_A(m)$
- **chosen-plaintext attack:** Trudy can get ciphertext for chosen plaintext

Symmetric Key Cryptography



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

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher, DES, AES

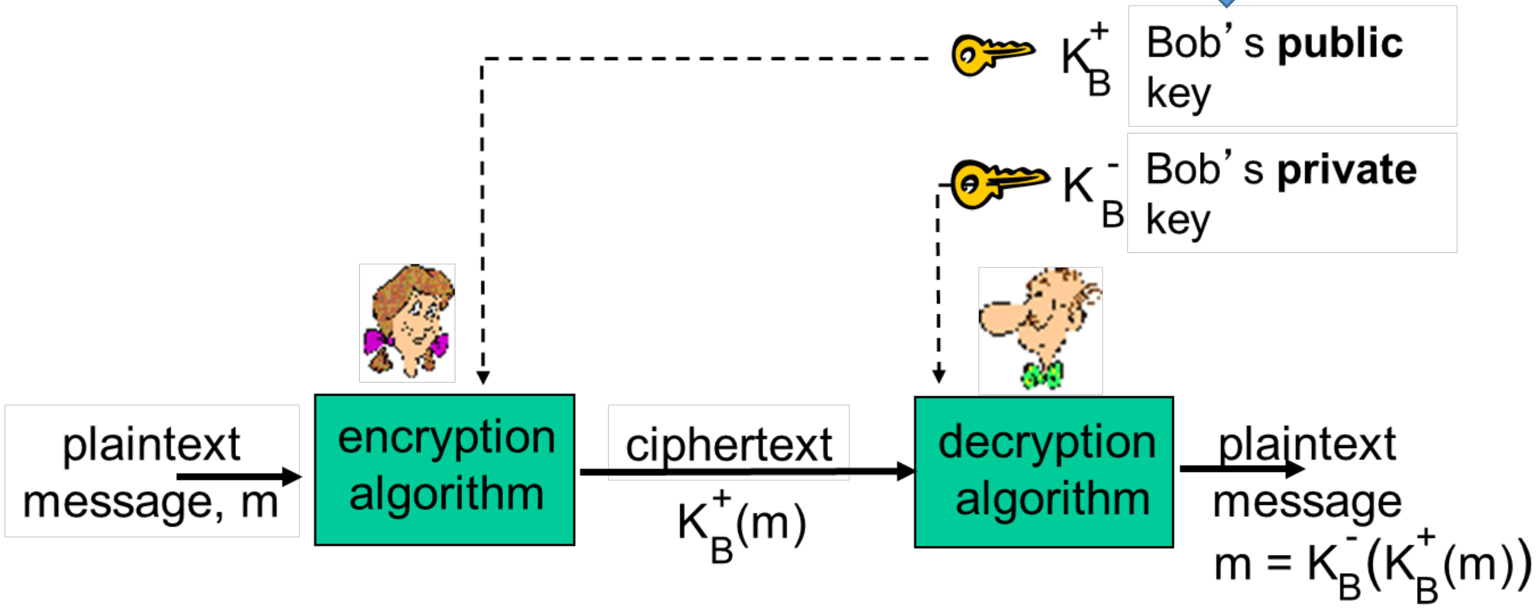
Q: how do Bob and Alice agree on key value?

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”)?

- 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 Encryption Algorithms

requirements:

1. need $k_B^+(.)$ and $k_B^-(.)$ such that

$$\mathbf{k_B^-\left(k_B^+(m)\right) = m}$$

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

Important example:

- **RSA:** Rivest, Shamir, Adelson algorithm

RSA: Another Important Property

The following property will be **very** useful later:

$$\underbrace{K_B^-\left(K_B^+(m)\right)}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+\left(K_B^-(m)\right)}_{\text{use private key first, followed by public key}}$$

result is the same!

RSA in Practice: Session Keys

- mathematical operations in RSA are computationally intensive
- DES is at least 100 times faster than RSA
- 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 key K_S
- once both have K_S, they use symmetric key cryptography

Agenda

8.1 What is network security?

8.2 Principles of cryptography

8.4 Authentication

8.3 Message integrity

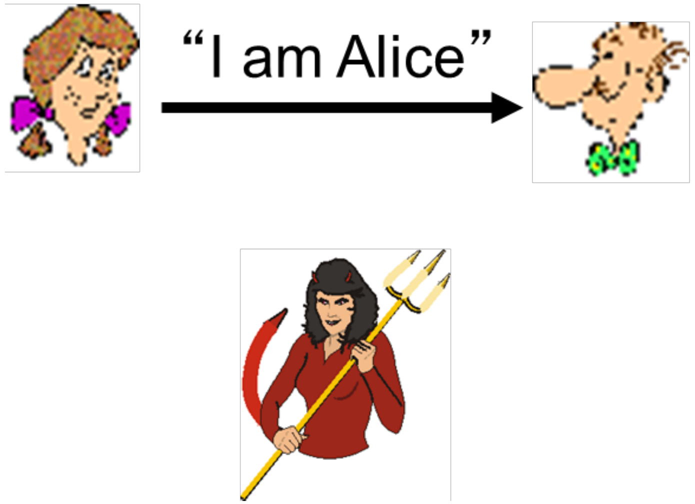
8.6 Securing TCP connections: SSL

8.7 Network layer security: IPsec

Authentication (1 of 2)

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

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

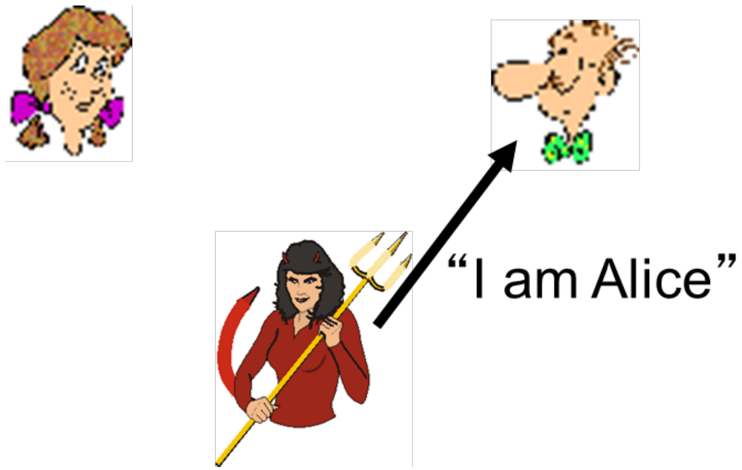


Failure scenario??

Authentication (2 of 2)

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

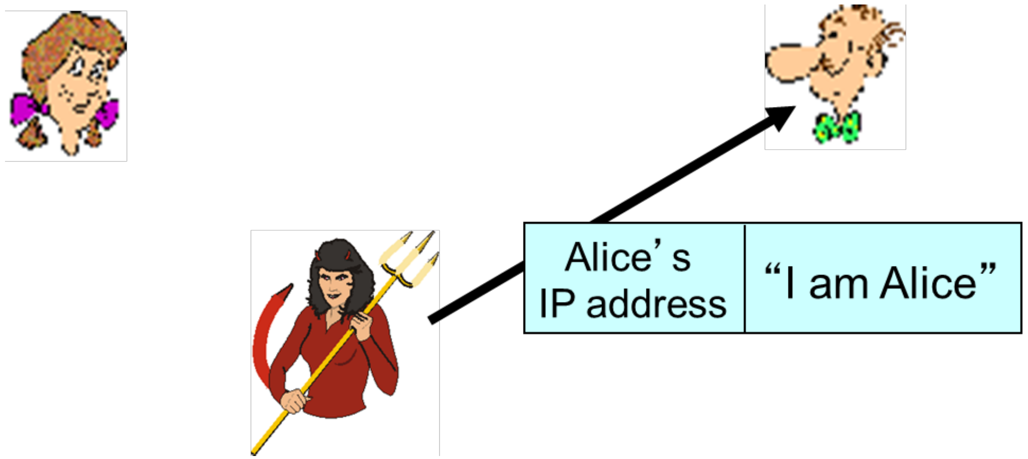
Authentication: using IP address (1 of 2)

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

Failure scenario??

Authentication: using IP address (2 of 2)

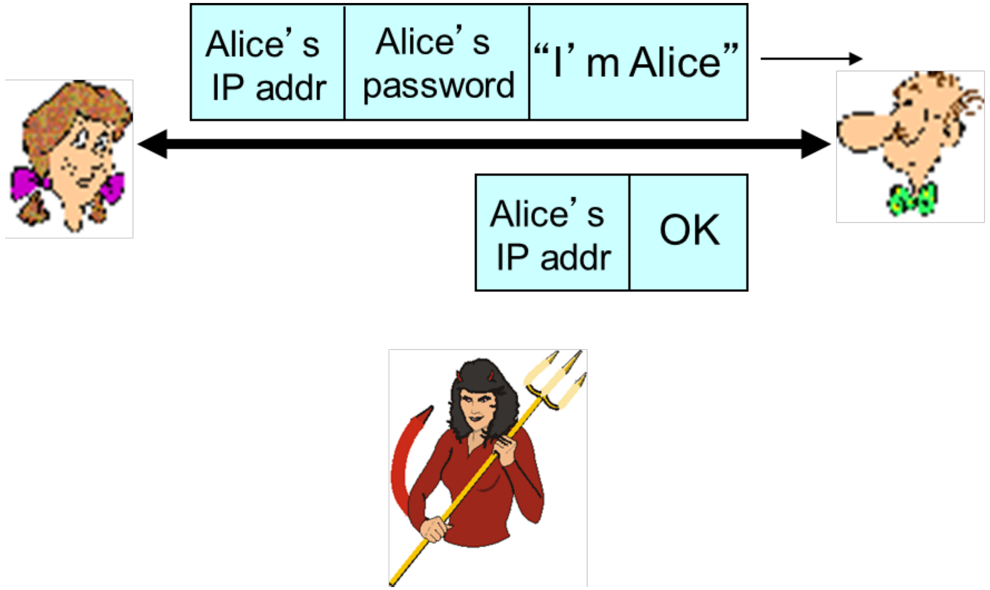
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

Authentication: use a key

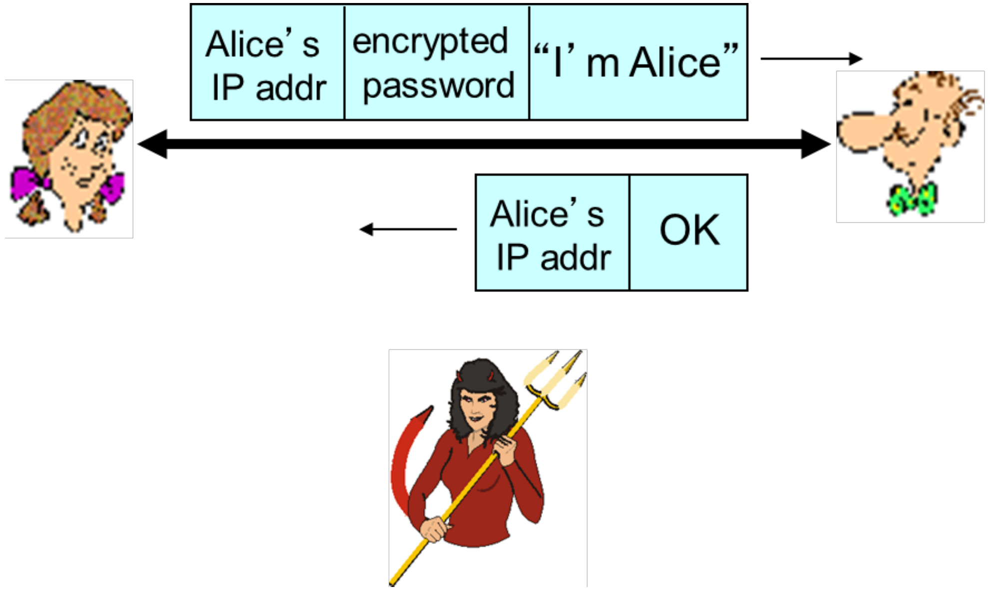
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: encrypt the password (1 of 2)

Protocol ap3.1: Alice says “I am Alice” and sends her **encrypted** secret password to “prove” it.

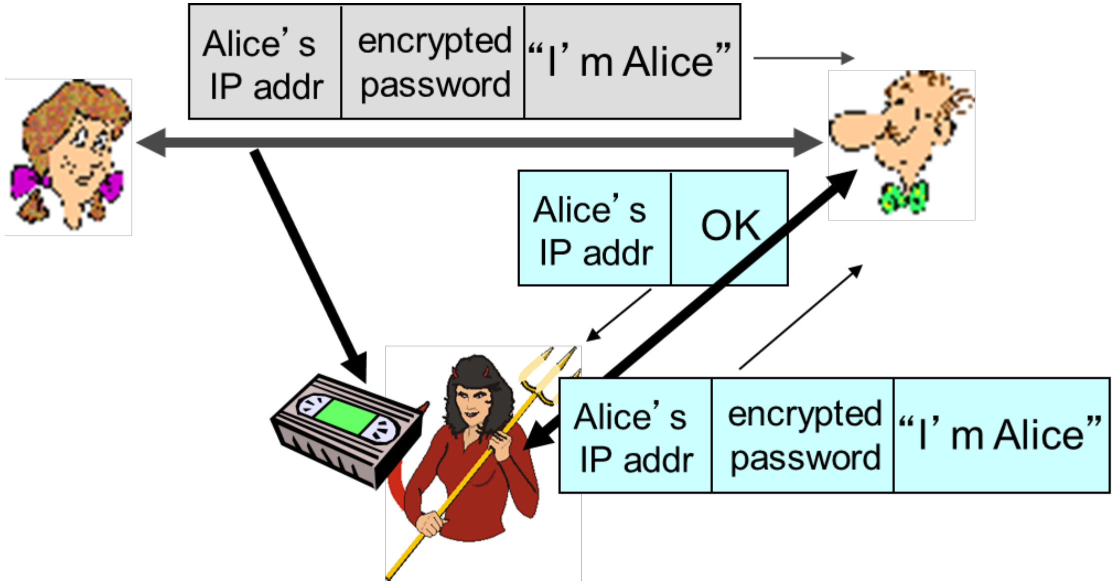


Failure scenario??

Authentication: encrypt the password (2 of 2)

Protocol ap3.1: Alice says "I am Alice" and sends her **encrypted** secret password to "prove" it.

The attacker doesn't learn Alice's password, but it does not need it.



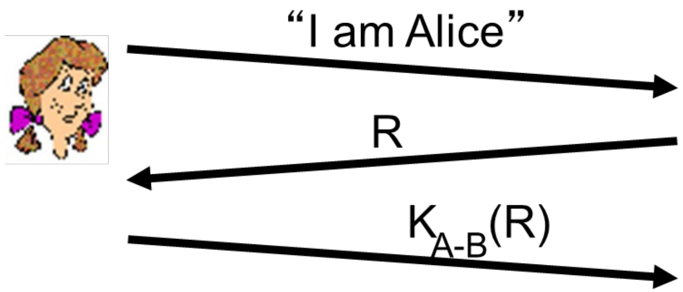
record and playback **still** works!

Authentication: use a nonce

Goal: avoid playback attack

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

ap4.0: 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?

Hint for failure: Trudy waits that Bob tries to authenticate with "Alice"

Hint for drawbacks: key management and dissemination

Authentication: nonce + public key

ap4.0 required a shared symmetric key

- can we authenticate using public key techniques?

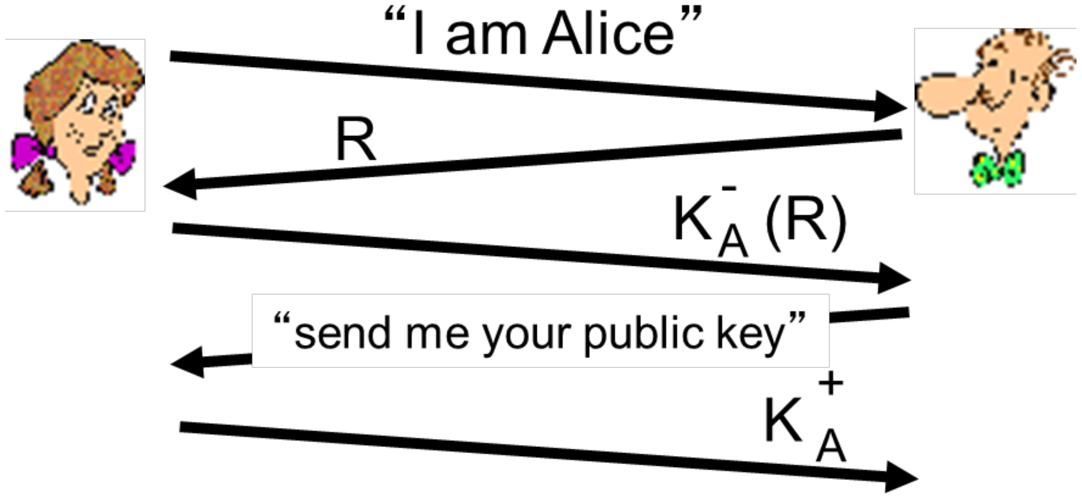
ap5.0: use nonce, public key cryptography

Bob computes

$$k_A^+\left(k_A^-(R)\right) = R$$

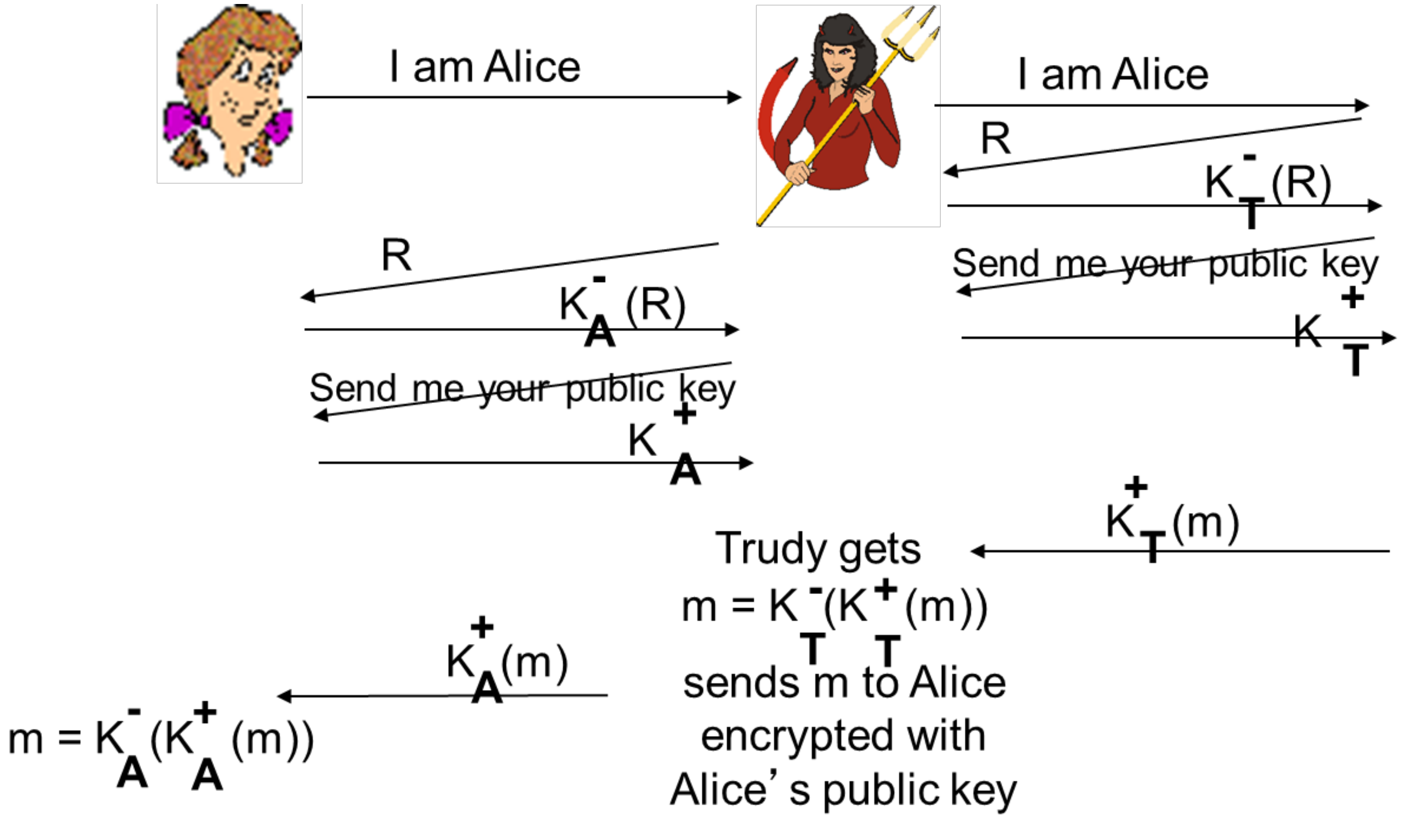
and knows only Alice could have the private key, that encrypted R such that

$$k_A^+\left(k_A^-(R)\right) = R$$



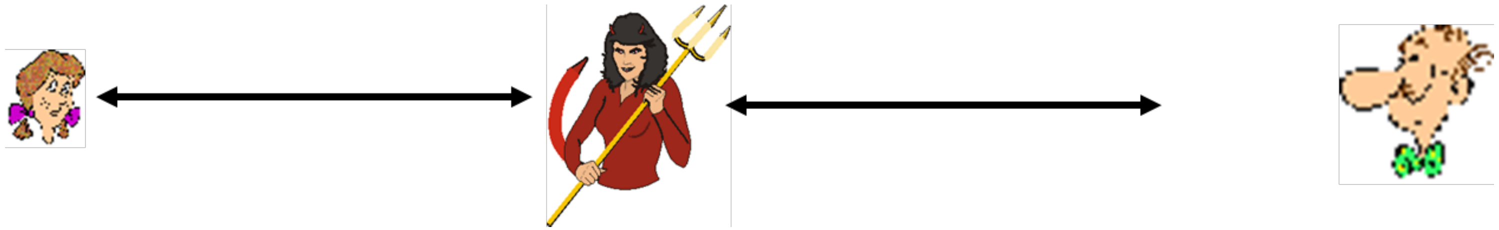
ap5.0: Security Hole (1 of 2)

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



ap5.0: Security Hole (2 of 2)

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



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!

Agenda

8.1 What is network security?

8.2 Principles of cryptography

8.4 Authentication

8.3 **Message integrity**

8.6 Securing TCP connections: SSL

8.7 Network layer security: IPsec

Digital Signatures (1 of 3)

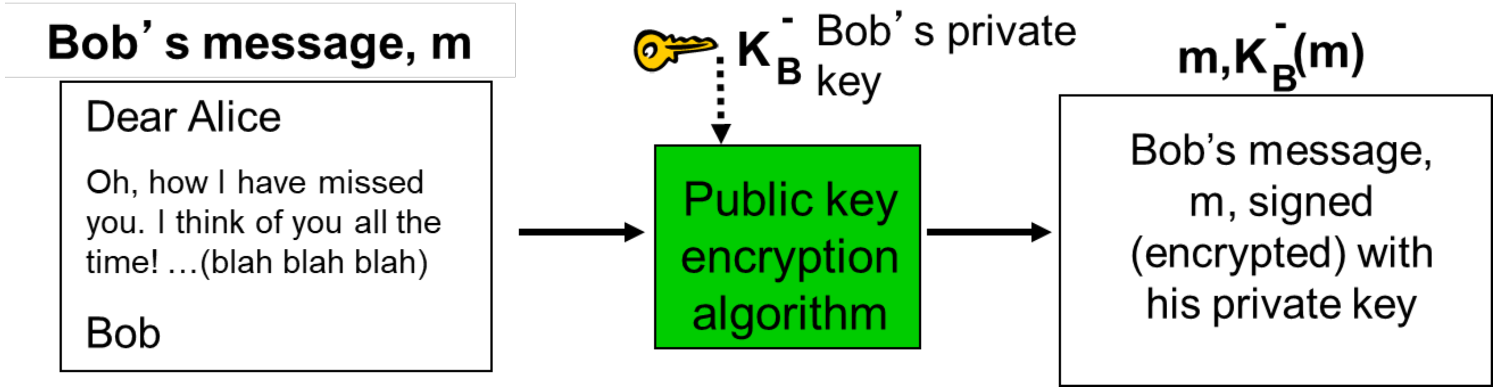
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing 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

Digital Signatures (2 of 3)

simple digital signature for message m:

- Bob signs m by encrypting with his private key $K_B^-(m)$, creating "signed" message, $K_B^-(m)$



Digital Signatures (3 of 3)

- suppose Alice receives msg m, with signature: $m, K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key

  k_B to $^+k_B(m)^-$ then checks $K_B(K_B^+(m)^-) = m$.
- If $k_B^+(k_B^-(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

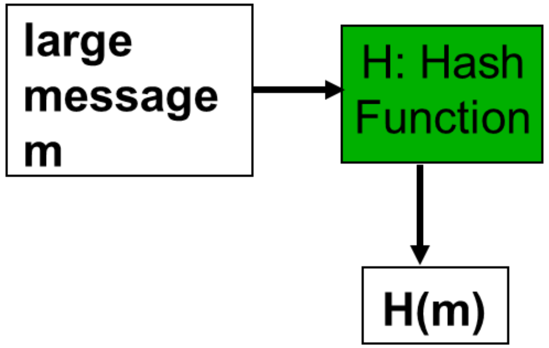
- It was Bob who signed m
- Bob signed m and not m'
  - By the way, Bob cannot repudiate m

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)
- then, encrypt (sign) the hash H(m)

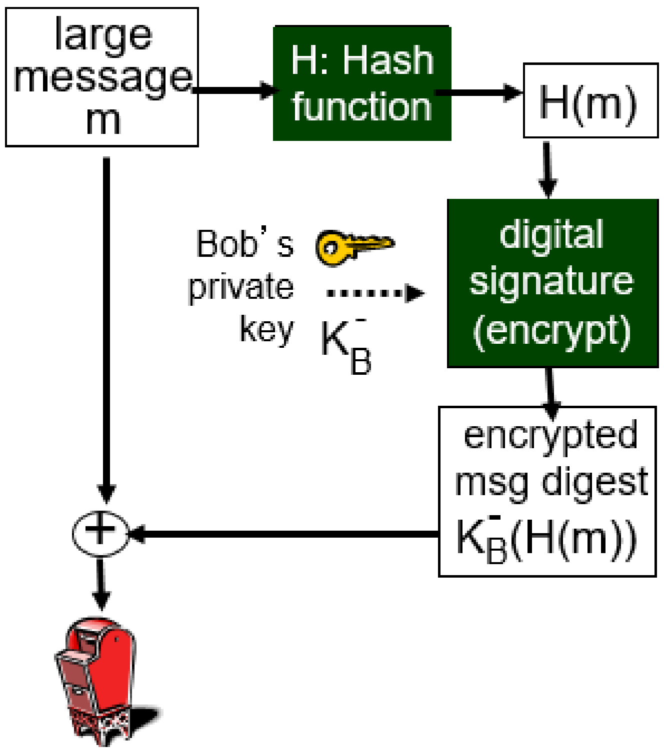


Hash function properties:

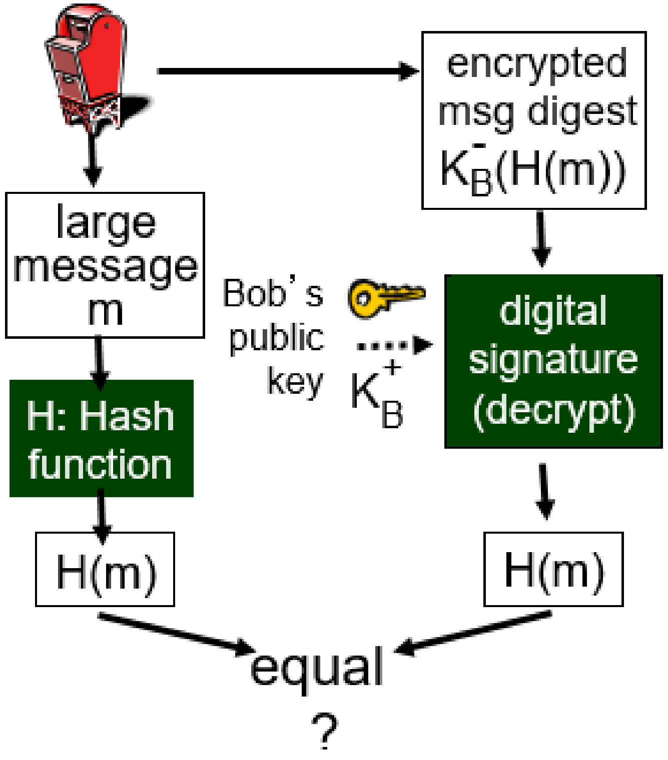
- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

Digital Signature = Signed Message Digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:

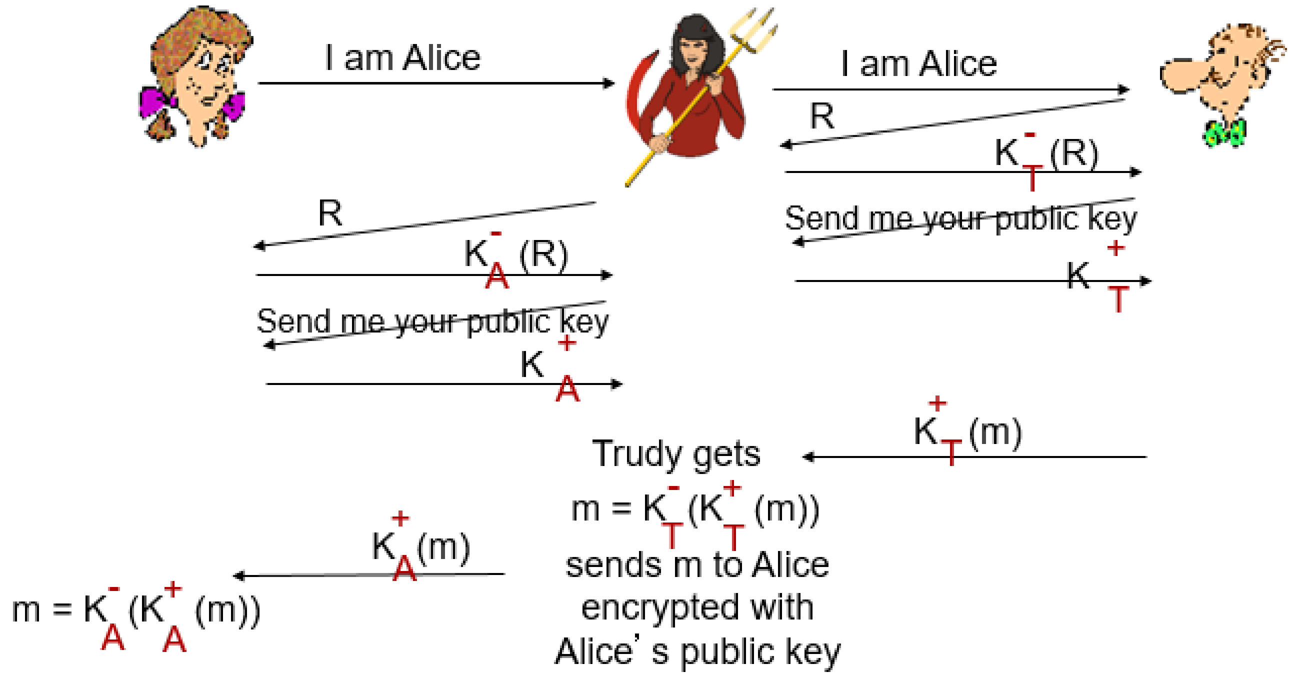


Hash Function Algorithms

- 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

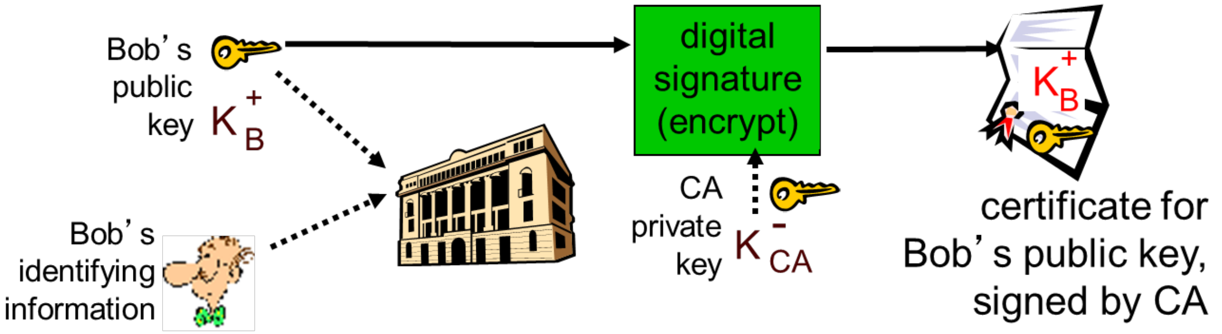
Recall: ap5.0 Security Hole

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



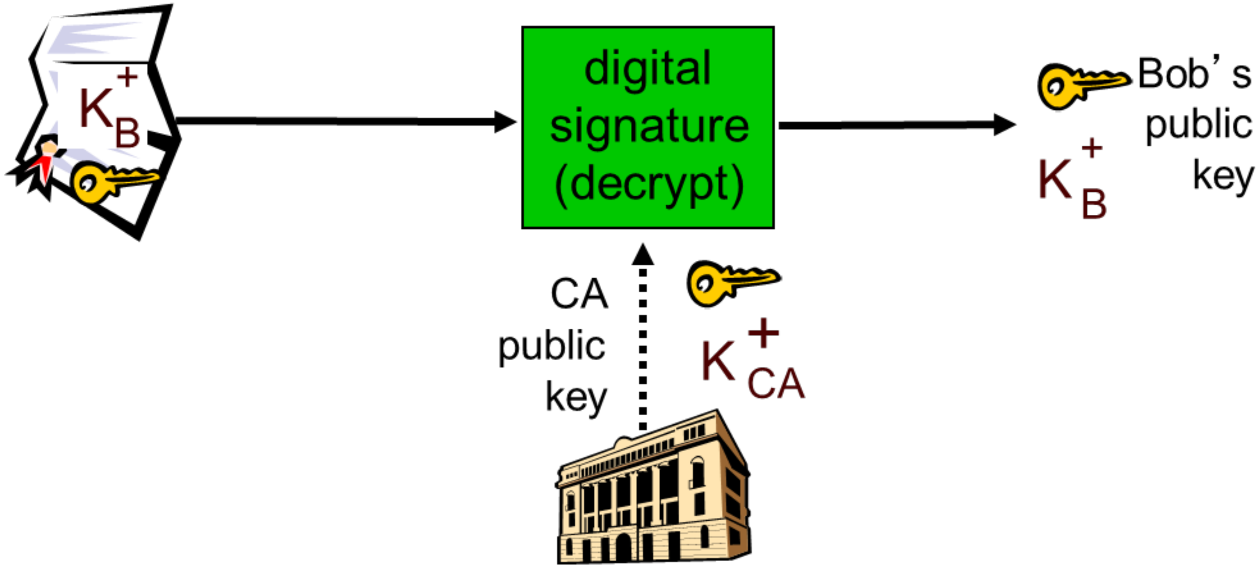
Certification Authorities (1 of 2)

- **certification authority (CA):** binds public key to particular entity, E.
- E(person, router) 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 (2 of 2)

- 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
- Requirements for it to work:
  - Everybody must know K^{+}_{CA} (e.g.:pre-shipping CA's certificates with the browser)
  - Nobody can know K^{-}_{CA} (except the CA)



Agenda

8.1 What is network security?

8.2 Principles of cryptography

8.4 Authentication

8.3 Message integrity

8.6 Securing TCP connections: SSL

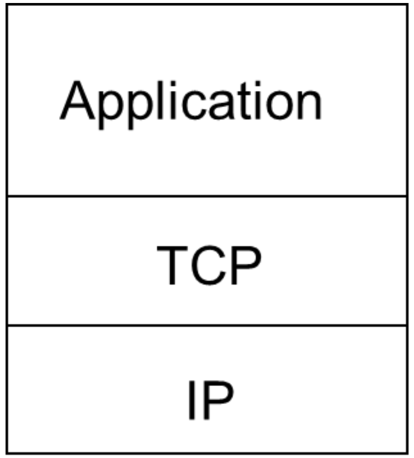
8.7 Network layer security: IPsec

SSL: Secure Sockets Layer

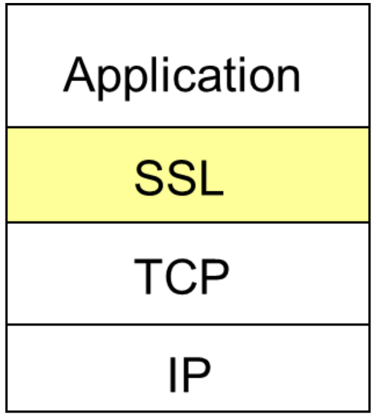
- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation TLS: transport layer security, RFC 2246
- provides
  - **confidentiality**
  - **integrity**
  - **authentication**

- original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- available to all TCP applications
  - secure socket interface

SSL and TCP/IP



normal application



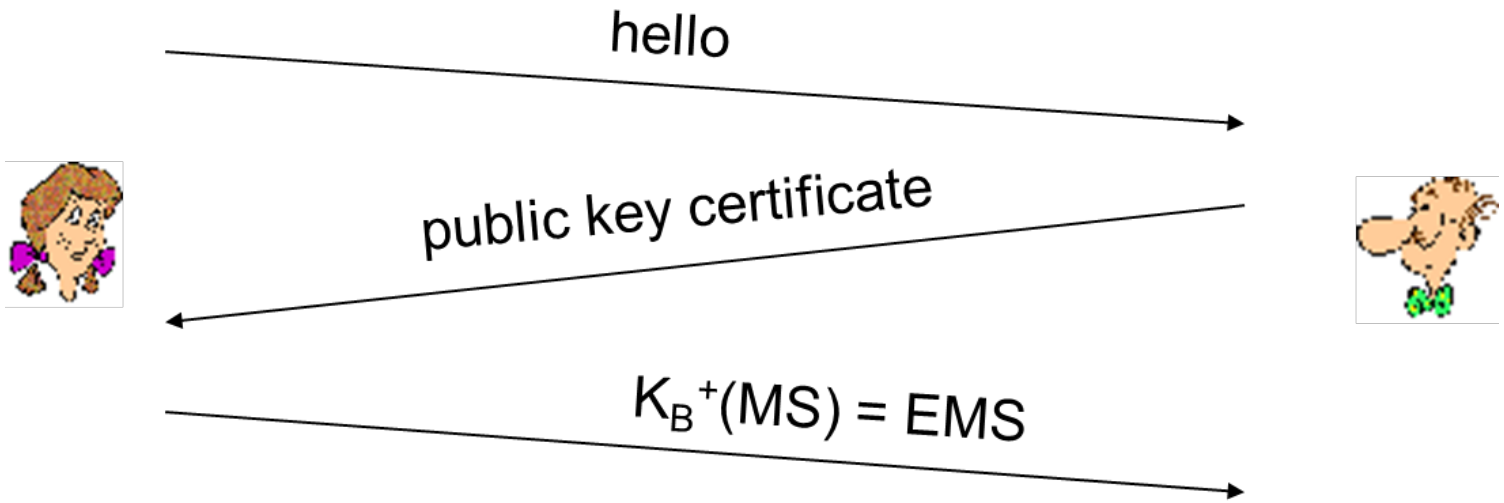
application with SSL

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

Toy SSL: A Simple Secure Channel

- **handshake:** Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- **key derivation:** Alice and Bob use shared secret to derive set of keys
- **data transfer:** data to be transferred is broken up into series of records
- **connection closure:** special messages to securely close connection

Toy: A Simple Handshake



MS: master secret

EMS: encrypted master secret

Toy: Key Derivation

- considered bad to use same key for more than one cryptographic operation
  - use 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 and creates the keys

Toy: Data Records

- why not encrypt data in constant stream as we write it to TCP?
  - where would we put the MAC? If at end, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
  - each record carries a MAC
  - receiver can act on each record as it arrives
- issue: in record, receiver needs to distinguish MAC from data
  - want to use variable-length records

Toy: Sequence Numbers

- **problem:** attacker can capture and replay record or re-order records
- **solution:** put sequence number into MAC:
  - $MAC = MAC(M_x, \text{sequence} \mid\mid \text{data})$
  - note: no sequence number field

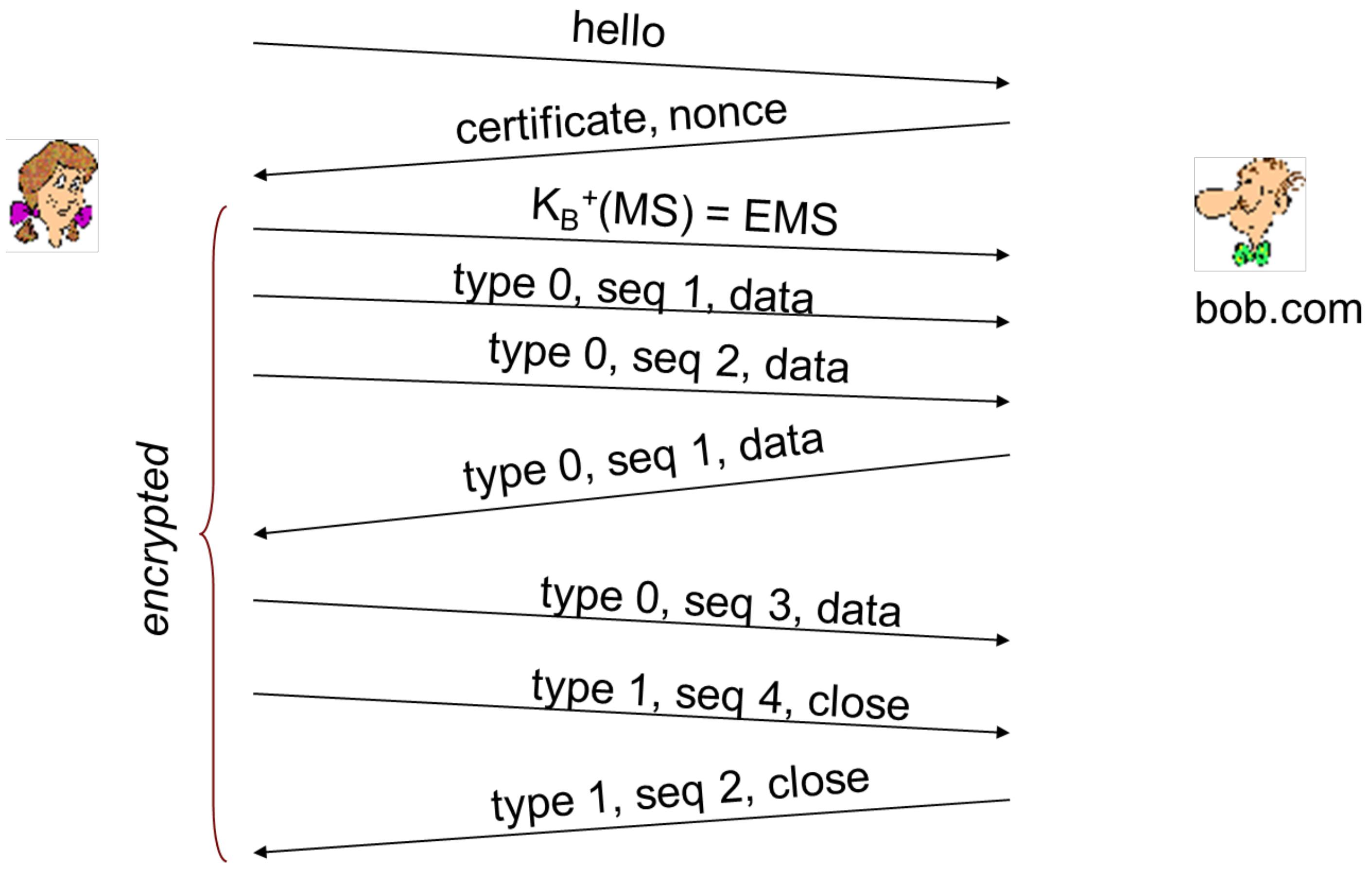
- **problem:** attacker could replay all records
- **solution:** use nonce

Toy: Control Information

- **problem:** 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 closure

$$MAC = MAC(M_x, sequence \,||\, type \,||\, data)$$

Toy SSL: Summary



Toy SSL is not complete

- how long are fields?
- which encryption protocols?
- want negotiation?
  - allow client and server to support different encryption algorithms
  - allow client and server to choose together specific algorithm before data transfer

SSL Cipher Suite

- cipher suite
  - public-key algorithm
  - symmetric encryption algorithm
  - MAC algorithm
- SSL supports several cipher suites
- negotiation: client, server agree on cipher suite
  - client offers choice
  - server picks one

- common SSL symmetric ciphers
  - DES – Data Encryption Standard: block
  - 3DES – Triple strength: block
  - RC2 – Rivest Cipher 2: block
  - RC4 – Rivest Cipher 4: stream
- SSL Public key encryption
  - RSA

Real SSL: Handshake (1 of 4)

Purpose

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

Real SSL: Handshake (2 of 4)

1. client sends list of algorithms it supports, along with client nonce
2. server chooses algorithms from list; sends back: choice + certificate + server nonce
3. client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. client sends a MAC of all the handshake messages
6. server sends a MAC of all the handshake messages

Real SSL: Handshake (3 of 4)

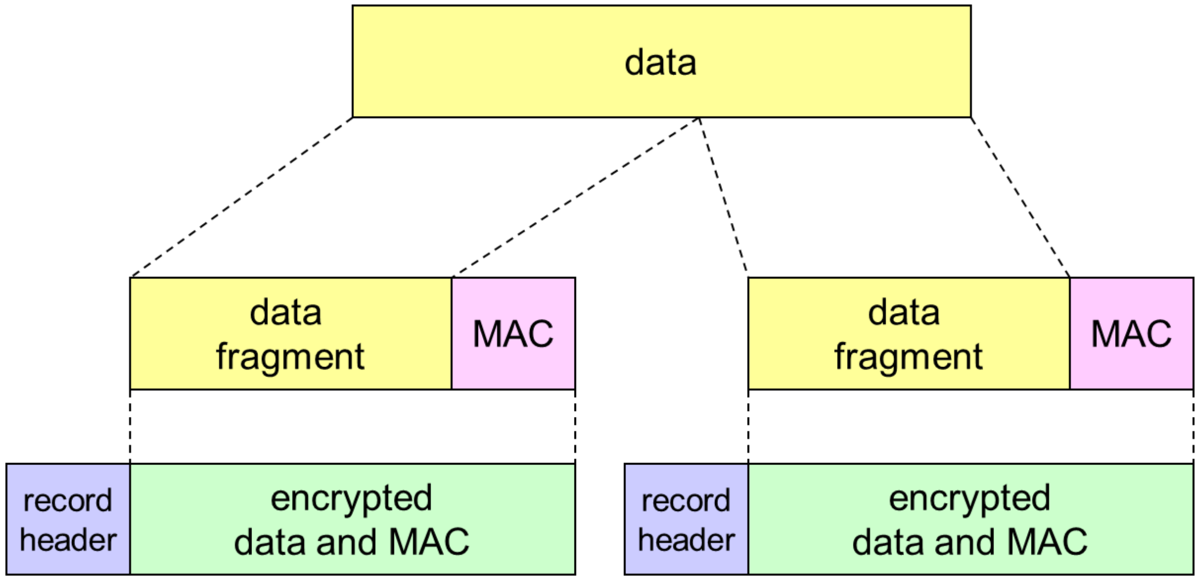
last 2 steps protect handshake from tampering

- client typically offers range of algorithms, some strong, some weak
- man-in-the middle could delete stronger algorithms from list
- last 2 steps prevent this
  - last two messages are encrypted

Real SSL: Handshake (4 of 4)

- why two random nonces?
- suppose Trudy sniffs all messages between Alice & Bob
- next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

SSL Record Protocol

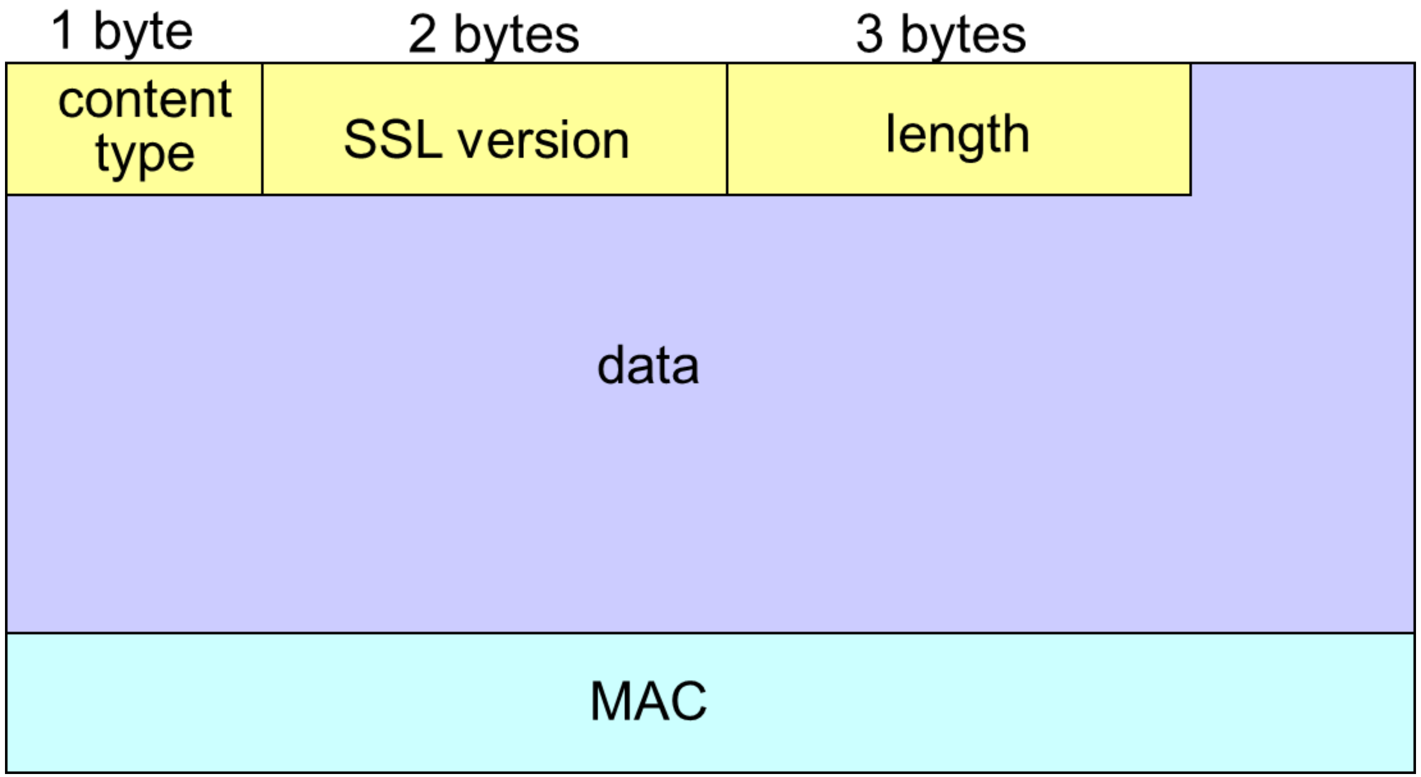


record header: content type; version; length

MAC: includes sequence number, MAC key M_x

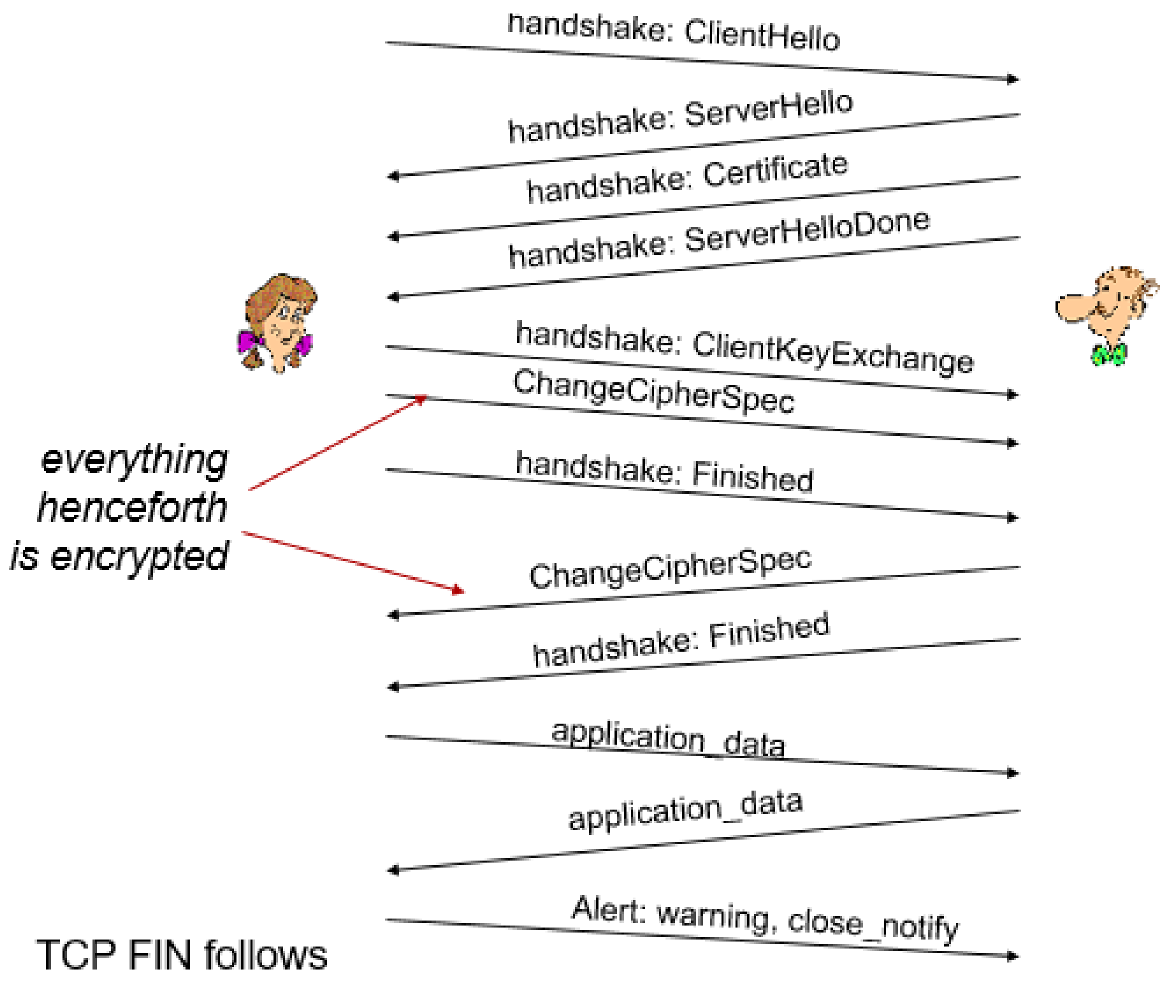
fragment: each SSL fragment: 2^{14} bytes (~16 Kbytes)

SSL Record Format



data and MAC encrypted (symmetric algorithm)

Real SSL Connection



Key Derivation

- client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - produces master secret
- master secret and new nonces input into another random-number generator: “key block”
  - because of resumption: TBD
- key block sliced and diced:
  - client MAC key
  - server MAC key
  - client encryption key
  - server encryption key
  - client initialization vector (IV)
  - server initialization vector (IV)

Agenda

8.1 What is network security?

8.2 Principles of cryptography

8.4 Authentication

8.3 Message integrity

8.6 Securing TCP connections: SSL

8.7 Network layer security: IPsec

What is Network-Layer Confidentiality ?

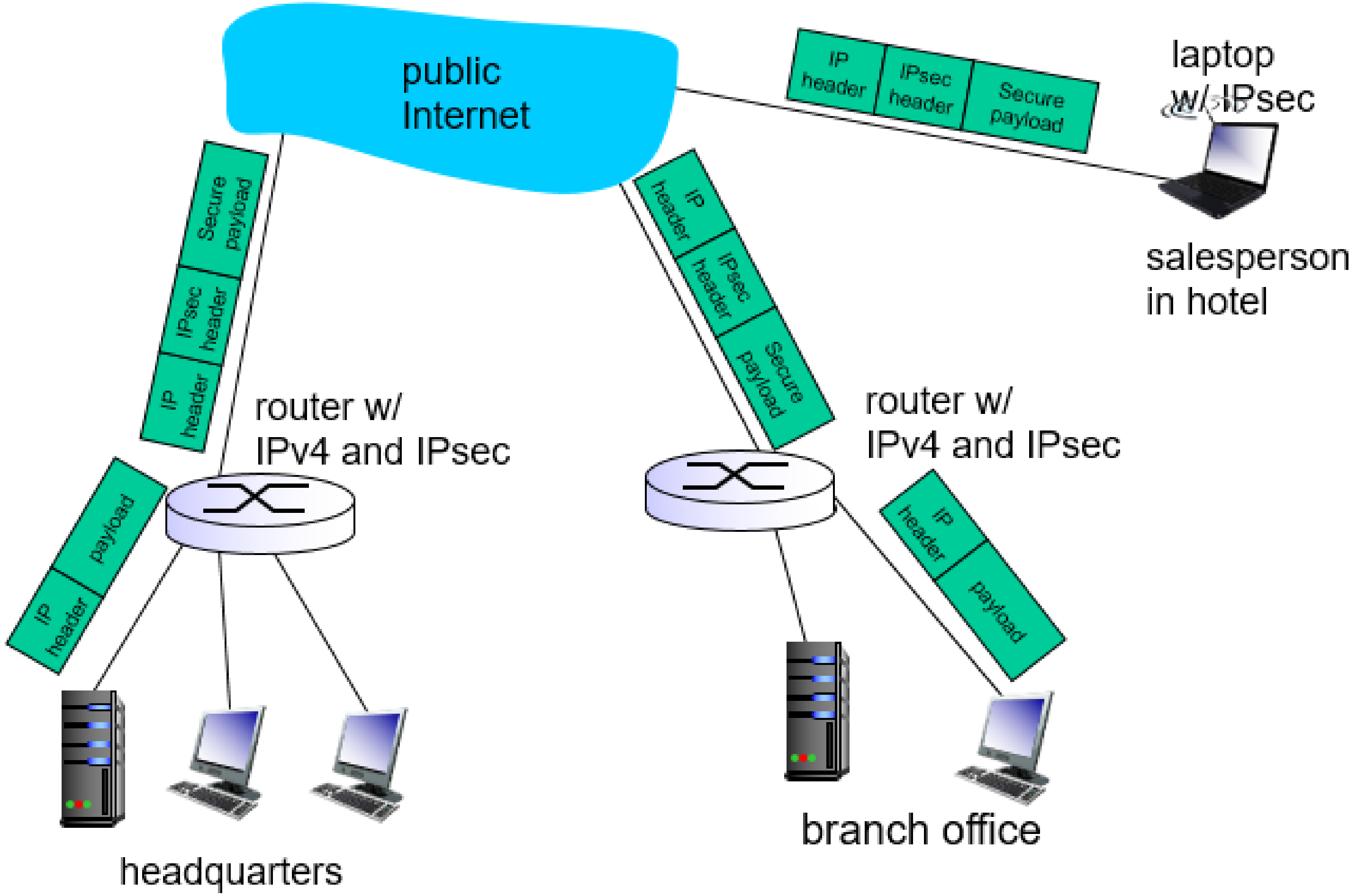
between two network entities:

- 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)

motivation:

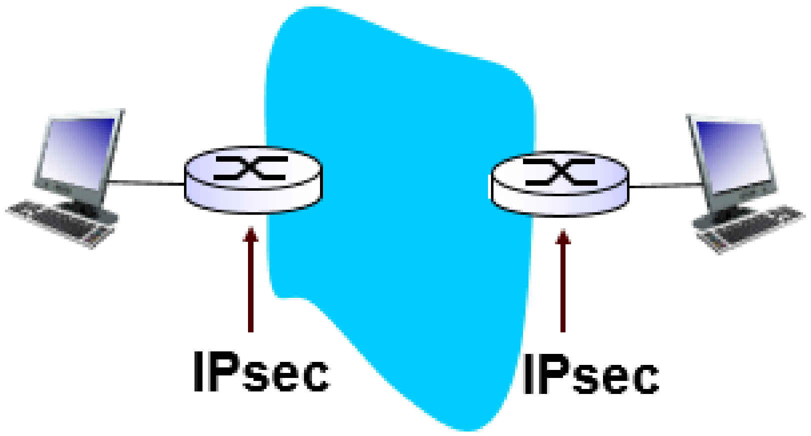
- institutions often want private networks for security
  - costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet
  - encrypted before entering public Internet
  - logically separate from other traffic

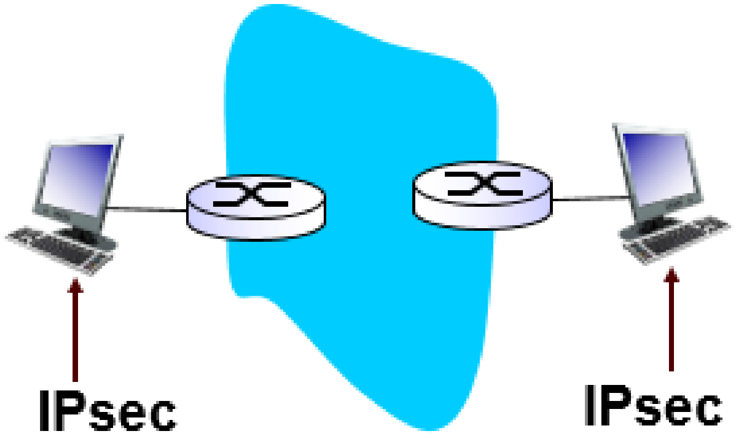


IPsec Services

- data integrity
- origin authentication
- replay attack prevention
- confidentiality
- two protocols providing different service models:
  - AH
  - ESP

IPsec – Transport Mode





- Tunnel mode:
  edge routers IPsec-aware

- Host mode:
  hosts IPsec-aware

Two IPsec Protocols

- 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

Four Combinations are Possible!

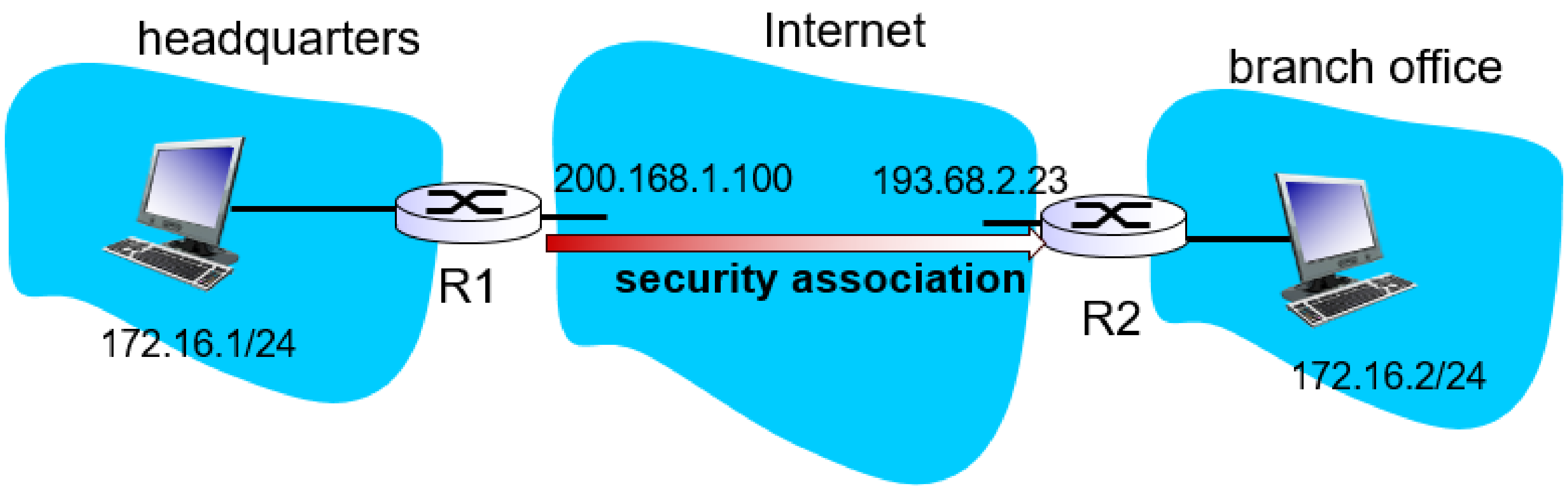
Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

most common and most important

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?

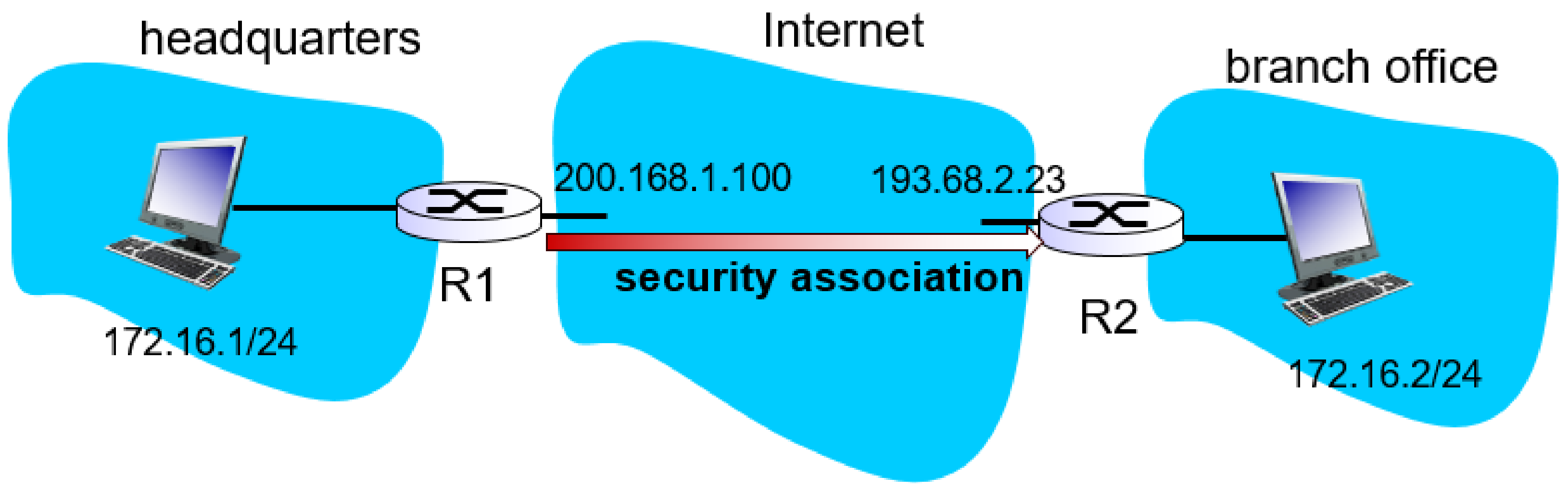
Example SA from R1 to R2 (1 of 2)



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)

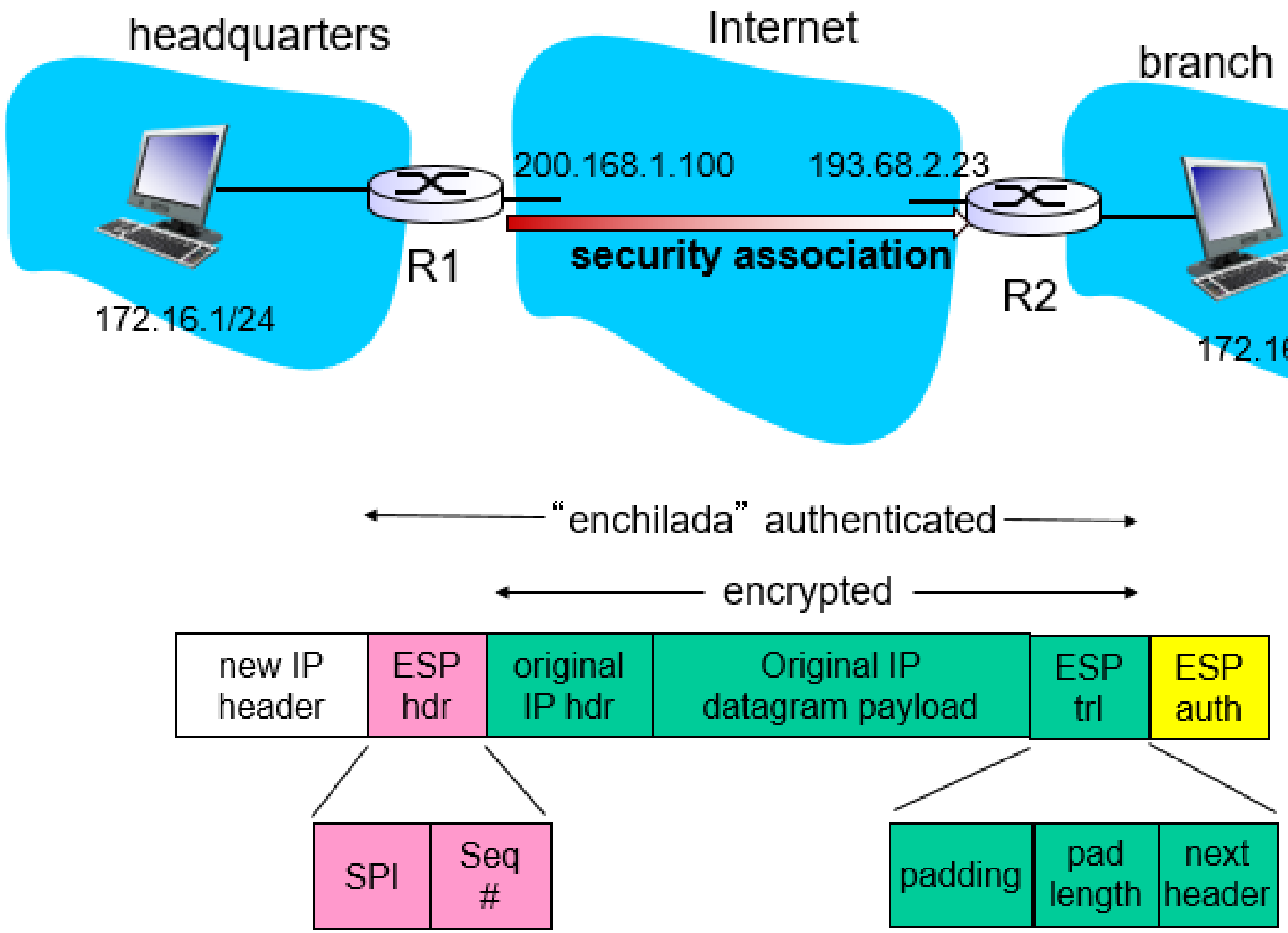
Example SA from R1 to R2 (2 of 2)



R1 stores for SA:

- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

What Happens?

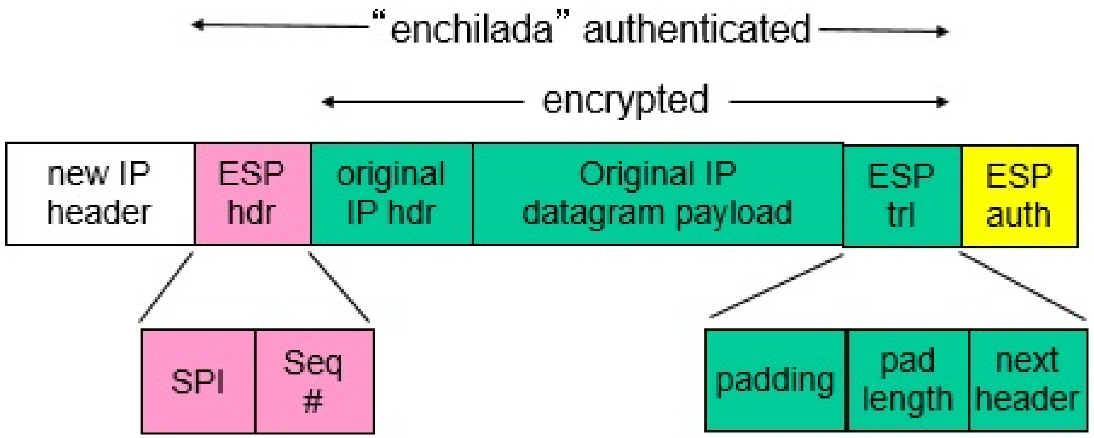


R1: Convert Original Datagram to IPsec Datagram

- 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



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.

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

SLUT