

## Telematics Course (Winter Semester 2003/04)

### FAQs about final exam

Telematics group  
University of Göttingen, Germany

Telematics group  
University of Göttingen, Germany

### F&Qs

- When is the final exam for WS03/04?
  - It will take place on March 1, 2004, from 10:00am-11:30am
- How difficult will it be?
  - It depends. If you are not very acquainted with the content
- How should I prepare for it, then?
  - Review the content of the lecture. **Try to complete the homework at your own effort.** If you have any question, please come to me.
- When should I register for it?
  - If you study a bachelor's program, fine – go to Munopag website and register online
  - If you do not have a Munopag entry, please send an email to Munopag: [krull@cs.uni-goettingen.de](mailto:krull@cs.uni-goettingen.de)
  - The deadline is 06.02.2004, the last lecture

WS 2003/04

2

Telematics group  
University of Göttingen, Germany

### Security Concerns

- 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

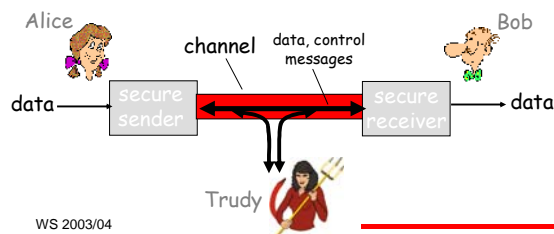
WS 2003/04

3

Telematics group  
University of Göttingen, Germany

### Friends and enemies: Alice, Bob, Trudy

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



WS 2003/04

4

## Who might Bob, Alice be?

- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

## There are bad guys out there!

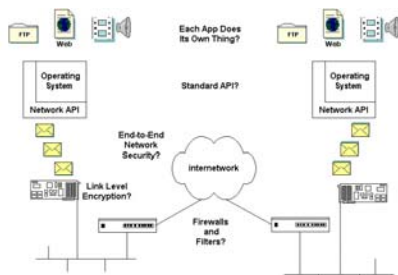
**Q:** What can a "bad guy" do?

**A:** a lot!

- *eavesdrop*: intercept messages
- actively *insert* messages into connection
- *impersonation*: can fake (spoof) source address in packet (or any field in packet)
- *hijacking*: "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

*more on this later .....*

## Where to Put the Protection?



## Host/OS Based Security

- Key idea: protect the *DATA*
  - End hosts are in control of data
  - Users are in control of end hosts
  - Users can and often will do dumb things, + hijackers!
  - Result: very difficult to protect all hosts
- Approaches:
  - OS software integrity (most attacks on non-patched OS)
  - user-level access control (AAA, tokens)
  - block unneeded services (finger, ftp, DNS)
  - path encryption via IPsec
  - device-level access control (MAC, IP, DNS) in servers, routers, Ethernet switches
  - e.g., host firewalling (such as TCP wrappers, IP chains)



### Symmetric key crypto: DES

#### DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase (“Strong cryptography makes the world a safer place”) decrypted (brute force) in 4 months
  - no known “backdoor” decryption approach
- making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

### Public Key Cryptography

#### symmetric key crypto

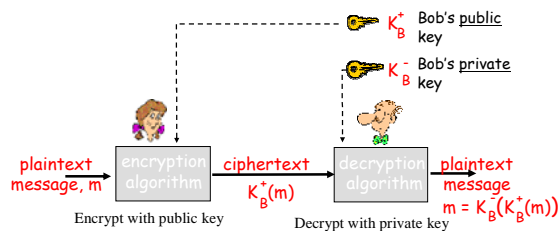
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never “met”)?

#### public key cryptography

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



### Public key cryptography



### Public key encryption algorithms

#### Requirements:

- 1 need  $K_B^+(\cdot)$  and  $K_B^-(\cdot)$  such that
 
$$K_B^-(K_B^+(m)) = m$$
- 2 given public key  $K_B^+$  it should be impossible to compute private key  $K_B^-$

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

### Authentication

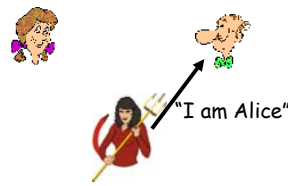
**Goal:** Bob wants Alice to “prove” her identity to him  
**Protocol ap1.0:** Alice says “I am Alice”



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

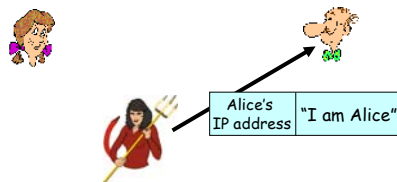
**Protocol ap2.0:** Alice says “I am Alice” in an IP packet containing her source IP address



Failure scenario??

### Authentication: another try

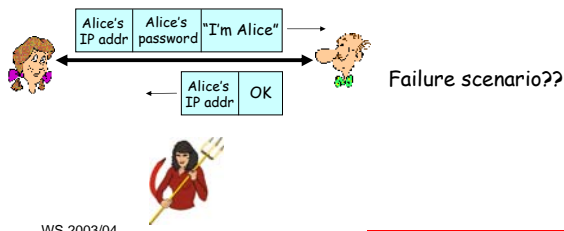
**Protocol ap2.0:** Alice says “I am Alice” in an IP packet containing her source IP address



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

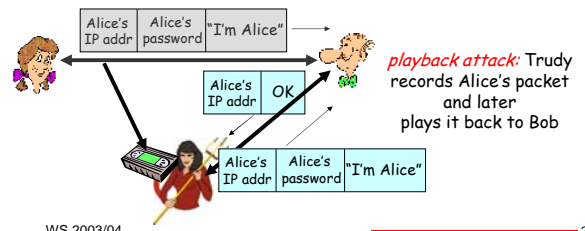
### Authentication: another try

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



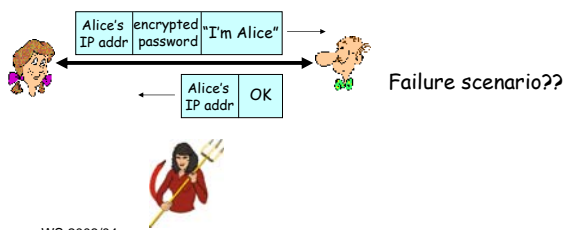
### Authentication: another try

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



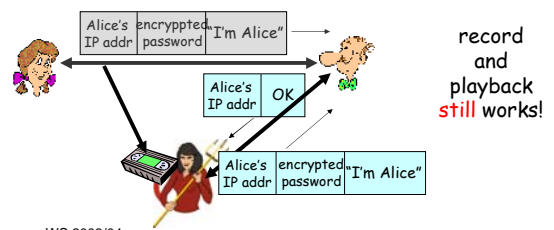
### Authentication: yet another try

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



### Authentication: another try

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

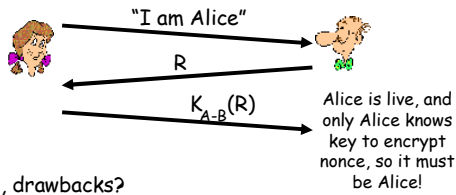


### Authentication: yet another try

**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



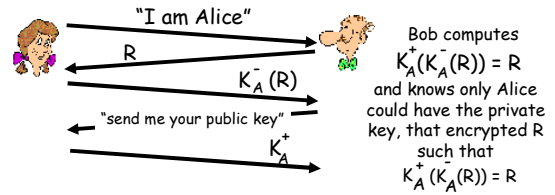
Failures, drawbacks?

### Authentication: ap5.0

ap4.0 requires shared symmetric key

• can we authenticate using public key techniques?

**ap5.0:** use nonce, public key cryptography



### ap5.0: security hole

**Man-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!

### Message Integrity Approaches

- Digital signature using RSA
  - special case of a message integrity where the code can only have been generated by one participant
  - compute signature with private key and verify with public key
- Keyed MD5
  - sender:  $m + \text{MD5}(m + k) + E(k, \text{private})$
  - receiver
    - recovers random key using the sender's public key
    - applies MD5 to the concatenation of this random key message
- MD5 with RSA signature
  - sender:  $m + E(\text{MD5}(m), \text{private})$
  - receiver
    - decrypts signature with sender's public key
    - compares result with MD5 checksum sent with message

## Digital Signatures

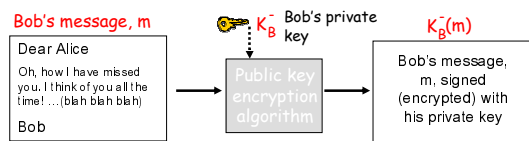
Cryptographic technique analogous to hand-written signatures.

- 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

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 (more)

- Suppose Alice receives msg  $m$ , digital signature  $K_B^-(m)$
- 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:

- Bob signed  $m$ .
- No one else signed  $m$ .
- Bob signed  $m$  and not  $m'$ .

Non-repudiation:

- ✓ Alice can take  $m$ , and signature  $K_B^-(m)$  to court and prove that Bob signed  $m$ .

## Trusted Intermediaries

Symmetric key problem:

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

Solution:

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

Public key problem:

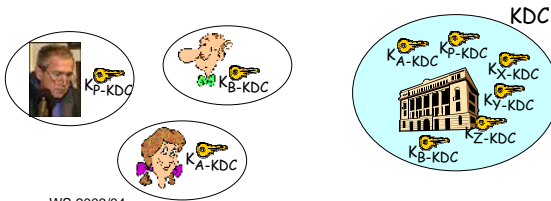
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- trusted certification authority (CA)

### Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- **KDC**: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys,  $K_{A-KDC}$   $K_{B-KDC}$ , for communicating with KDC.

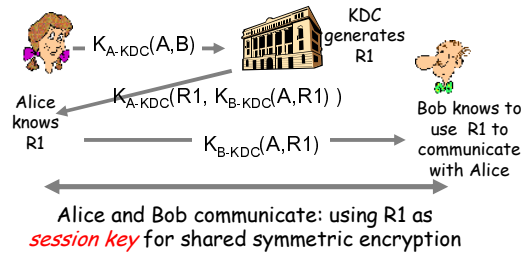


WS 2003/04

33

### Key Distribution Center (KDC)

**Q:** How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?

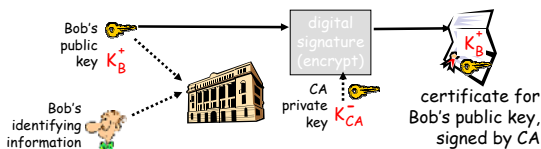


WS 2003/04

34

### Certification Authorities

- **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"

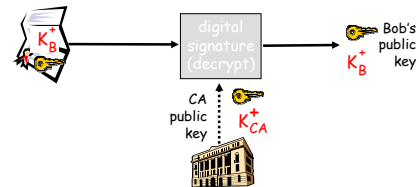


WS 2003/04

35

### 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

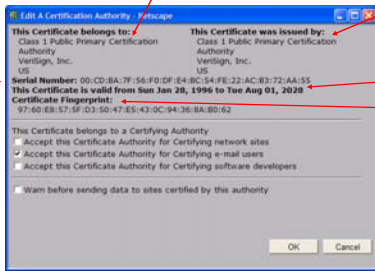


WS 2003/04

36

### A certificate contains:

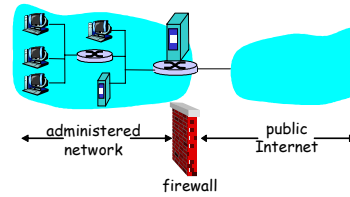
- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- info about certificate issuer
- valid dates
- digital signature by issuer

### Firewalls

**firewall**  
isolates organization's internal net 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.**

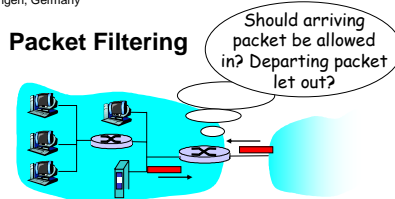
- e.g., attacker replaces CIA's homepage with something else

**allow only authorized access to inside network (set of authenticated users/hosts)**

**two types of firewalls:**

- application-level
- packet-filtering

### Packet Filtering



- internal network connected to Internet via **router firewall**
- router **filters packet-by-packet**, decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

### Packet Filtering

- **Example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.**
  - All incoming and outgoing UDP flows and telnet connections are blocked.
- **Example 2: Block inbound TCP segments with ACK=0.**
  - Prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

### Example Firewall: ipchains

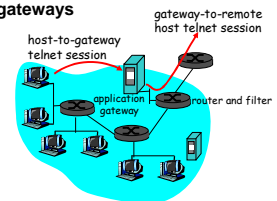
```
-A input -s 192.168.0.0/255.255.0.0 -d 0.0.0.0/0.0.0.0 -j DENY
-A input -s 172.0.0.0/255.240.0.0 -d 0.0.0.0/0.0.0.0 -j DENY
-A input -s 10.0.0.0/255.0.0.0 -d 0.0.0.0/0.0.0.0 -j DENY
-A input -s 224.0.0.0/224.0.0.0 -d 0.0.0.0/0.0.0.0 -j DENY
-A input -s 0.0.0.0/0.0.0.0 -d a.b.c.d/255.255.255.255 22:22 -p 6 -j ACCEPT
-A input -s 0.0.0.0/0.0.0.0 -d a.b.c.d/255.255.255.255 1024:65535 -p 6 ! -j ACCEPT
```

### Example Firewall: Cisco Router Filters

```
access-list 100 deny ip 192.168.0.0 0.0.255.255 any
access-list 100 deny ip 172.0.0.0 0.15.255.255 any
access-list 100 deny ip 10.0.0.0 0.255.255.255 any
access-list 100 deny ip 0.0.0.0 0.255.255.255 any
access-list 100 deny ip 127.0.0.0 0.255.255.255 any
access-list 100 deny ip 224.0.0.0 31.255.255.255 any
access-list 100 deny ip 1.2.0.0 0.0.255.255 any
access-list 100 permit tcp any host 1.2.3.4 eq domain
access-list 100 permit udp any host 1.2.3.4 eq domain
access-list 100 deny tcp any host 1.2.3.5 eq telnet log
access-list 100 deny tcp any host 1.2.3.6 eq syn log
access-list 100 deny ip any host 1.2.3.4
access-list 100 permit ip any 1.2.0.0 0.0.255.255
access-list 100 deny ip any any
```

### Application gateways

- Filters packets on application data as well as on IP/TCP/UDP fields.
- **Example:** allow select internal users to telnet outside.



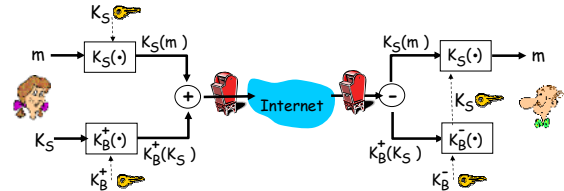
1. Require all telnet users to telnet through gateway.
2. For authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. Router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls and gateways

- **IP spoofing:** router can't know if data "really" comes from claimed source
- if multiple app's. 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.

Secure e-mail

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

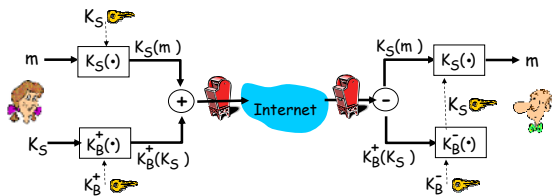


Alice:

- 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

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

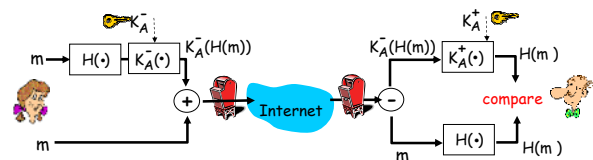


Bob:

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

Secure e-mail (continued)

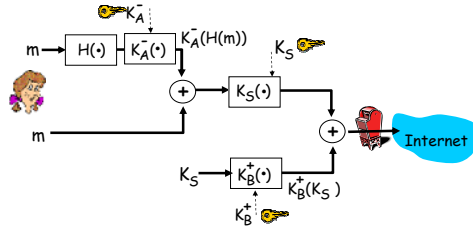
- Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

### Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



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

### Pretty good privacy (PGP)

- Internet e-mail encryption scheme, de-facto standard.
- uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described.
- provides secrecy, sender authentication, integrity.
- inventor, Phil Zimmerman, was target of 3-year federal investigation.

#### A PGP signed message:

```

---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Bob:My husband is out of town
tonight.Passionately yours,
Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHHgJGhgg/12EpJ+lo8gE4vB3mqJ
hFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

### Web Security

- authentication: basic, digest
- often supplemented by *cookies*
- access control via network addresses
- multi-layered:
  - SHTTP (secure HTTP) = just for HTTP (shhttp://)  
CommerceNet, Mosaic
  - SSL (→TLS) = generic for TCP (https://)  
implementation: SSLeay
  - IP security: host-to-host

### Web vulnerabilities

#### Risks:

1. revealing private information on server
2. intercept of client information (credit card records)
3. information about host → break in
4. execute programs, denial of service
5. server log privacy

#### Information Leakage:

- Altavista search for etc/passwd
- directory listings
- chroot
- soft links
- file ownership ↔ local protection
- web access
- cgi-bin

## HTTP access control - principles

- client doesn't know which method
- client attempts access (GET, PUT, ...) normally
- server returns  
HTTP/1.0 401 Unauthorized  
WWW-Authenticate: Basic realm="WallyWorld"
- realm: protection space
- client tries again with  
Authorization: Basic base64(user:password)
- passwords in the clear → not secure
- repeat cycle on each access

## Web Server Access Configuration

http://hoohoo.ncsa.uiuc.edu/docs/tutorials/user.html  
For NCSA httpd, Apache → .htaccess per directory or global:

```
AuthType Basic
AuthUserFile /etc/passwd
AuthName "Private information"
<Limit GET>
order deny,allow
require user hgs
deny from all
allow from .ncsa.uiuc.edu
</Limit>
```

Global configuration file access.conf:  
<Directory /full/path/to/protected/directory>  
AuthName name.of.your.server  
AuthType Basic  
AuthUserFile /usr/local/etc/httpd/conf/passwd  
<Limit GET POST>  
require user foo  
</Limit>  
</Directory>

## Secure sockets layer (SSL)

- **transport layer security to any TCP-based app using SSL services.**
- Standardized as TLS (Transport-layer Security)
- used between Web browsers, servers for e-commerce (**shttp**).
- security services:
  - server authentication
  - data encryption
  - client authentication (optional)
- **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:

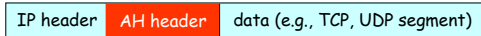
- Browser generates **symmetric session key**, encrypts it with server's public key, sends encrypted key to server.
- Using private key, server decrypts session key.
- Browser, server know session key
  - All data sent into TCP socket (by client or server) encrypted with session key.
- SSL: basis of IETF Transport Layer Security (TLS).
- SSL can be used for non-Web applications, e.g., IMAP.
- Client authentication can be done with client certificates.

### IPsec: Network Layer Security

- **Network-layer secrecy:**
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- **Network-layer authentication**
  - destination host can authenticate source IP address
- **Two principle protocols:**
  - authentication header (AH) protocol
  - encapsulation security payload (ESP) protocol
- **For both AH and ESP, source, destination handshake:**
  - create network-layer logical channel called a security association (SA)
- **Each SA unidirectional.**
- **Uniquely determined by:**
  - security protocol (AH or ESP)
  - source IP address
  - 32-bit connection ID

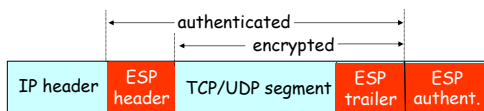
### Authentication Header (AH) Protocol

- provides source authentication, data integrity, no confidentiality
  - AH header inserted between IP header, data field.
  - protocol field: 51
  - intermediate routers process datagrams as usual
- AH header includes:**
- connection identifier
  - authentication data: source- signed message digest calculated over original IP datagram.
  - next header field: specifies type of data (e.g., TCP, UDP, ICMP)



### ESP Protocol

- provides secrecy, host authentication, data integrity.
- data, ESP trailer encrypted.
- next header field is in ESP trailer.
- ESP authentication field is similar to AH authentication field.
- Protocol = 50.



### Network Security (summary)

**Basic techniques.....**

- cryptography (symmetric and public)
- authentication
- message integrity
- key distribution
- access control and firewalls

**.... used in many different security scenarios**

- secure email and web
- secure transport (SSL/TLS)
- IPsec