SC250
Computer Networking I

Network Security

Prof. Matthias Grossglauser

School of Computer and Communication Sciences
EPFL

http://lcawww.epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

What is Network Security?

Confidentiality: only sender, intended receiver should "understand" message contents
- sender encrypts message
- receiver decrypts message

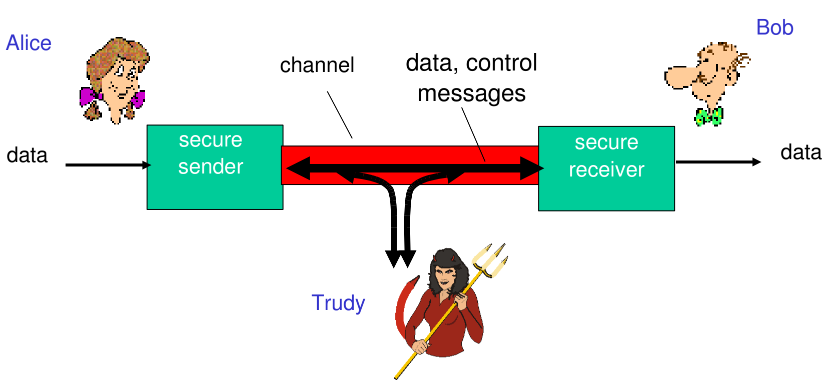
Authentication: sender, receiver want to confirm identity of each other

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



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?

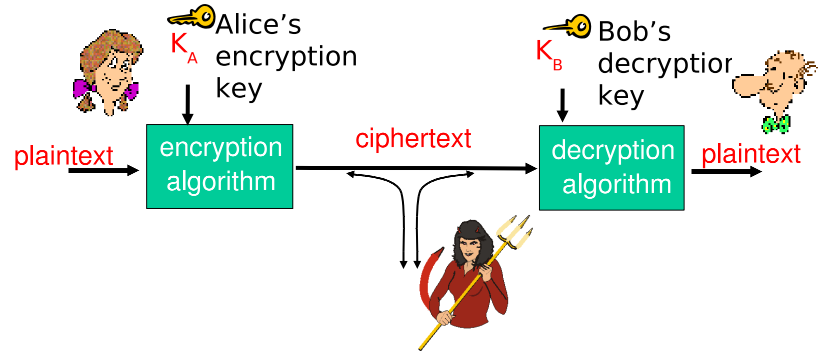
Many Threats

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)

The Language of Cryptography



symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

Symmetric Key Cryptography

Substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

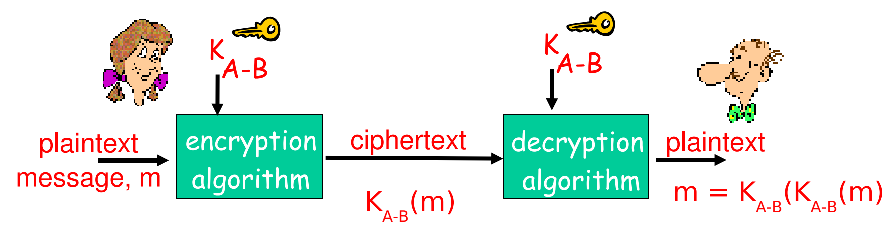
E.g.:

```
Plaintext:  bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Q: How hard to break this simple cipher?:

- brute force (how hard?)
- other?

Symmetric Key Cryptography



Symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{A-B}

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: how do Bob and Alice agree on key value?

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
- Making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

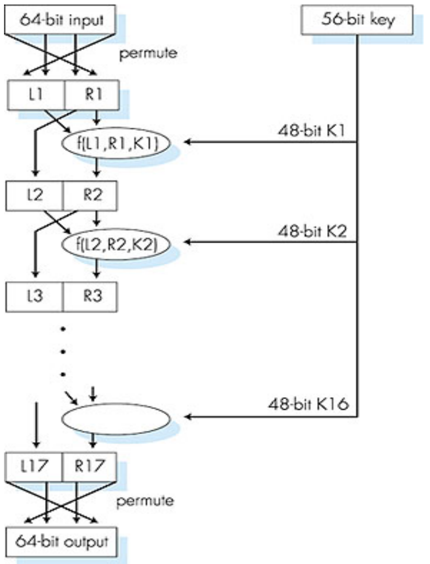
Symmetric Key Cryptography: DES

DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



AES: Advanced Encryption Standard

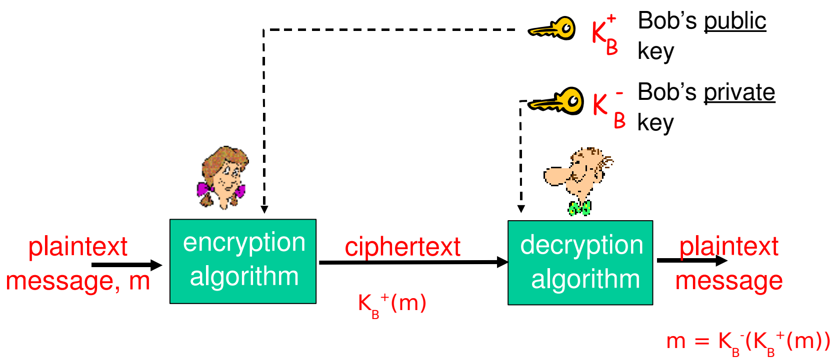
- New (Nov. 2001) symmetric-key NIST standard, replacing DES
- Processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- Brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

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 $K_B^+(\ )$ and $K_B^-(\ )$ 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, Adleman algorithm

RSA: Choosing Keys

- 1. Choose two large prime numbers p, q (e.g., 1024 bits each)
- 2. Compute n = pq, z = (p-1)(q-1)
- 3. Choose e (with e<n) that has no common factors with z. (e, z are "relatively prime").
- 4. Choose d such that ed-1 is exactly divisible by z (in other words: ed mod z = 1).
- 5. Public key is (n,e). Private key is (n,d).

(n,e) = K_B^+ ; (n,d) = K_B^-

RSA: Encryption, Decryption

- 0. Given (n,e) and (n,d) as computed above
- 1. To encrypt bit pattern, m, compute
  $c = m^e \bmod n$
  (i.e., remainder when m^e is divided by n)
- 2. To decrypt received bit pattern, c, compute
  $m = c^d \bmod n$
  (i.e., remainder when c^d is divided by n)

Magic happens! $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

RSA: Another Important Property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) \;=\; m \;=\; K_B^+(K_B^-(m))$$

use public key first, followed by private key

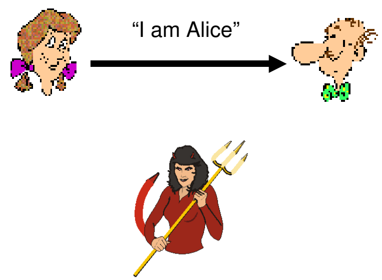
use private key first, followed by public key

Result is the same!

Authentication

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

Protocol ap1.0: Alice says "I am Alice"

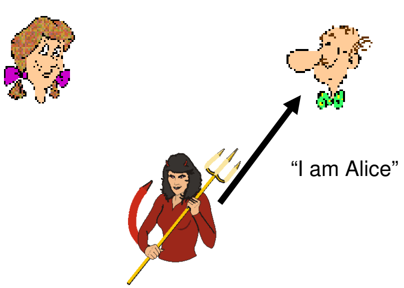


Failure scenario??

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

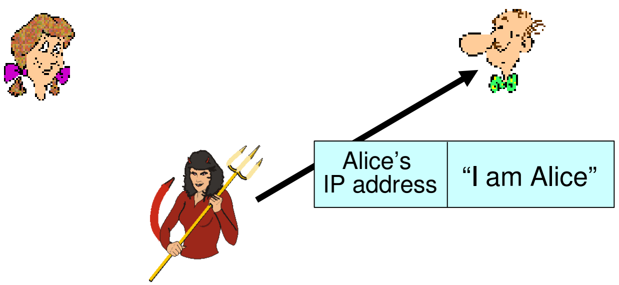
Authentication: Another Try

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

Failure scenario??

Authentication: Another Try

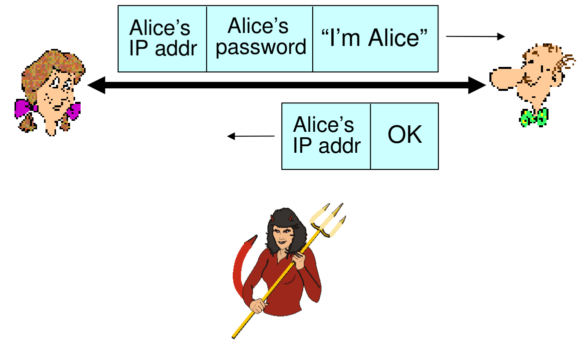
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



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

Authentication: Another Try

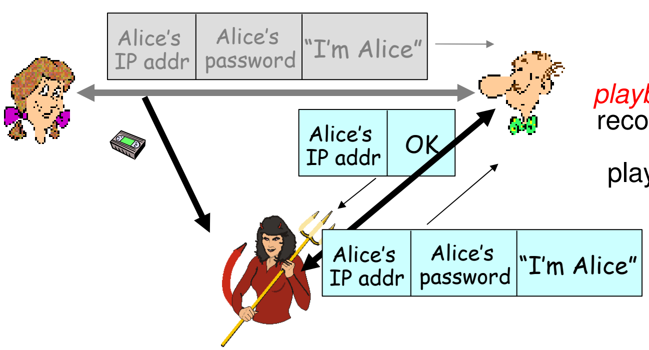
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



Failure scenario??

Authentication: Another Try

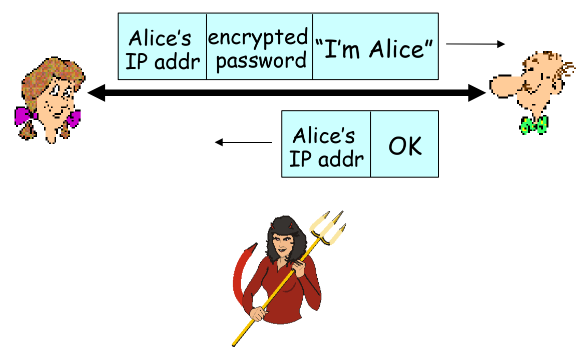
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



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

Authentication: Yet another Try

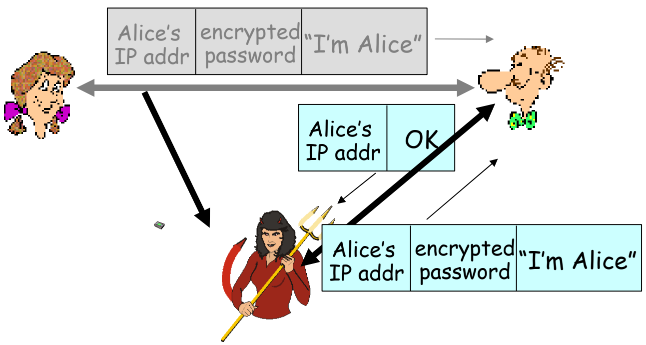
Protocol ap3.1: Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.



Failure scenario??

Authentication: Another Try

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



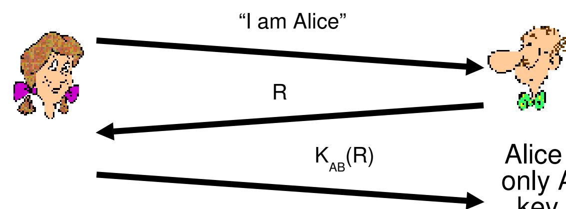
record and playback still works!

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



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

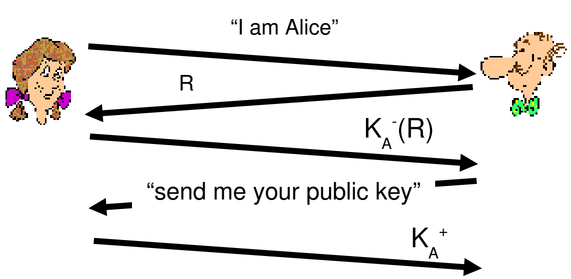
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

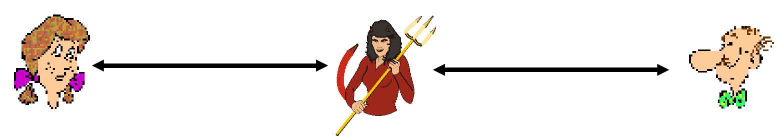
- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



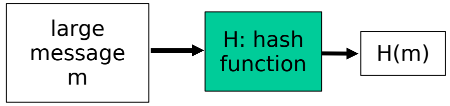
ap5.0: Security Hole

Man (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!

Integrity: Message Digests



Computationally expensive to encrypt long messages

Goal: fixed-length, easy-to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m)*.

Hash function properties:

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

Warning: Hash Function not Equal to Cryptographic Hash Function

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of message
- is many-to-one
- Good for random errors, bad against attacker

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

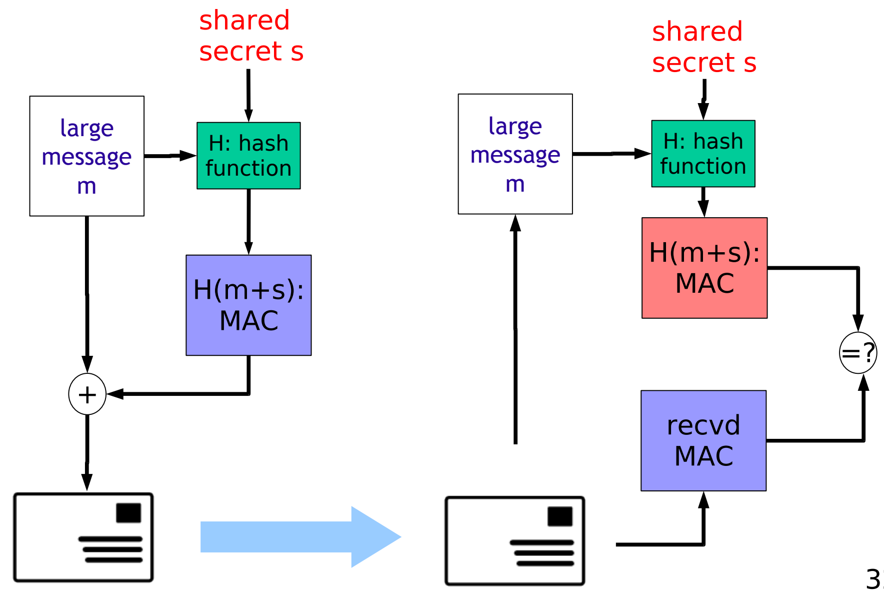
message	ASCII format	message	ASCII format
I O U 1	49 4F 55 31	I O U 9	49 4F 55 39
0 0 . 9	30 30 2E 39	0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
	B2 C1 D2 AC		B2 C1 D2 AC

different messages but identical checksums!

Hash Functions

- 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

Integrity: Message Authentication Code



Message Integrity 2: Digital Signatures

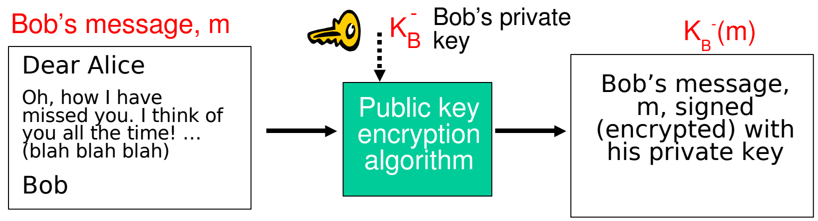
Cryptographic technique analogous to hand-written signatures.

- 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
- Note: MAC not sufficient, as Alice could "falsify" MAC of a message received from Bob
  - Impossible to prove if Bob really signed the message, or if Alice forged it
  - Reason: shared secret! Both are able to compute H(m+s)

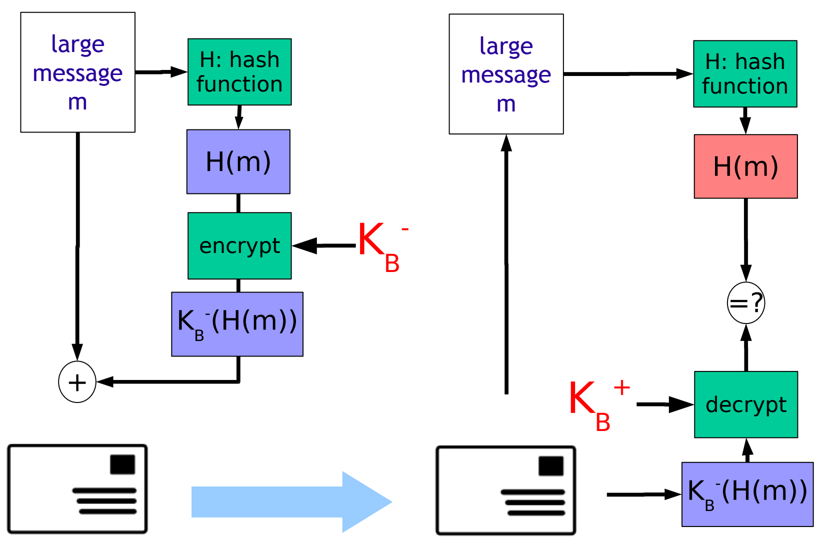
Digital Signatures

Simple digital signature for message m:

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



Integrity: Message Signature



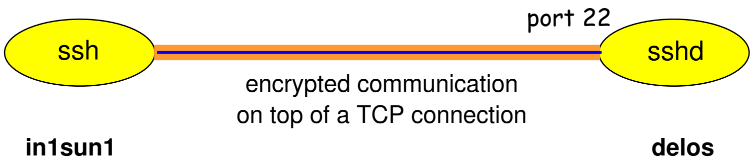
MAC and Signature

- MAC
  - Impossible to tamper with message
    - If message m' sent, H(m'+s) \= H(m+s)
  - Requires that shared secret s is established
- Signature
  - Impossible to tamper with message, and verifiable/non-forgeable
  - Requires private-public key
- Note:
  - Replay attack still possible -> can include sequence number into MAC/signature

Case Stuy: SSH (Secure Shell)

- Secure remote session
  - Encrypted connection, secret per session key
  - Port 22
  - Use instead of telnet
- Authentication options
  - Encrypted password
  - Preinstalled public key
- Tunnels and port redirection
  - Redirect the connections of other applications
  - Automatic redirection of X connections -> secure access of remote GUI

Basic SSH Connection




```
in1sun1% ssh delos.imag.fr
```


Back to SSH: Architecture

- ssh-trans
  - server authentication, confidentiality, integrity
- ssh-userauth
  - authenticates the client-side user
- ssh-connect
  - multiplexes the encrypted tunnel into several logical channels (enables port redirection)

ssh-trans

- Server authentication
  - each server host must have a host key
  - server host key is used during key exchange to verify that the client is really communicating with the correct server.
  - the client must have prior knowledge of the server's public host key:
    - client has a local database that associates each host name (as typed by the user) with the corresponding public host key (file known_hosts)
    - host name - key association can be certified by a trusted certification authority.
- Danger if the client talks to an unknown host
  - man-in-the-middle attack

ssh-trans

- Confidentiality
  - data encrypted using a one-time secret session key
- Key exchange phase
  - Diffie-Hellman method to create a secret key K
- Encryption
  - symmetric encryption using K
  - several ciphers can be used
- Integrity
  - MAC (Message Authentication Code) included with each packet
  - computed from the shared secret key, packet sequence number, the contents of the packet

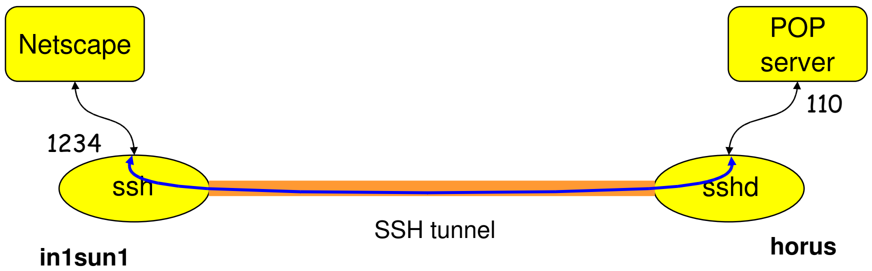
ssh-userauth

- Password
  - username, password on the remote system
- Public key authentication
  - user generates a pair of keys: public + secret
  - public key stored on the remote system (file authorized_keys)
  - authentication request
    - signature by the secret key over (session-id, username)
    - the signature verifed on the server by the public key

ssh-connect

- Multiple channels multiplexed into a single connection at the ssh-trans level
- Channels identified by numbers on each end
- Channels are individually flow-controlled
  - window size - amount of data to send

SSH Local Port Redirection

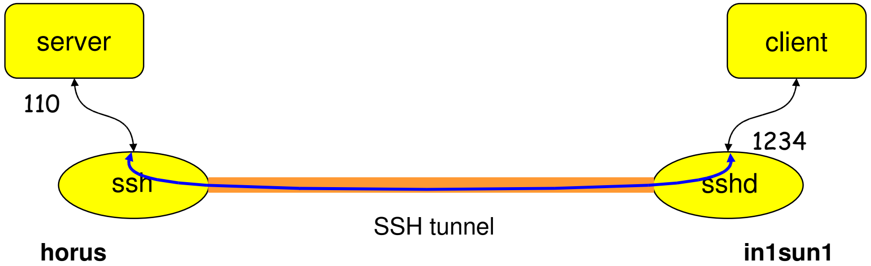


in1sun1% ssh -L 1234:horus.imag.fr:110 horus.imag.fr

config Netscape on in1sun1 - read e-mail by POP on: **localhost**, port 1234

e-mail will be read on **horus** through the ssh tunnel

SSH Remote Port Redirection




```
horus% su root
horus% ssh -R 1234:in1sun1.imag.fr:110
     in1sun1.imag.fr
```

Netscape on in1sun1: read e-mail by POP on **`localhost`** port 1234 (read in fact on **`horus`**)

SSH: Summary

- Excellent security
  - Encryption
  - Two-way authentication
  - Should be used instead of telnet/rlogin
- Integration with other applications
  - Through tunneling
  - E-mail, X-windows,…

Trusted Intermediaries

<u>Symmetric key problem:</u>

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

Solution:

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

<u>Public key problem:</u>

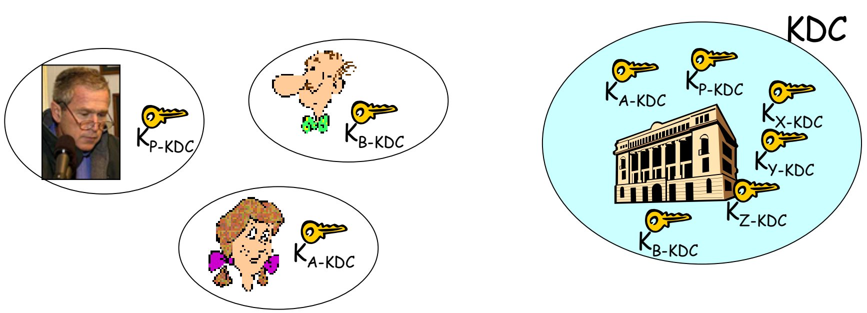
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- Trusted certification authority (CA)

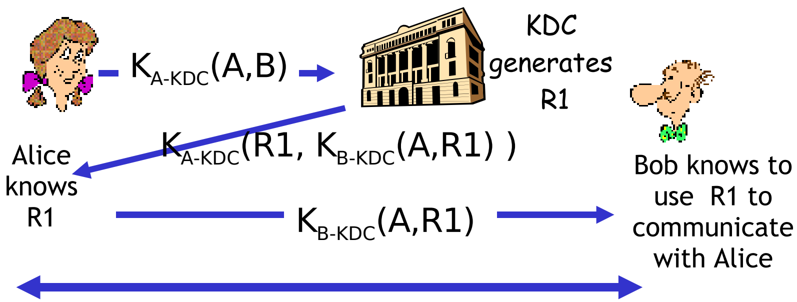
Key Distribution Center (KDC)

- 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.



Key Distribution Center (KDC)

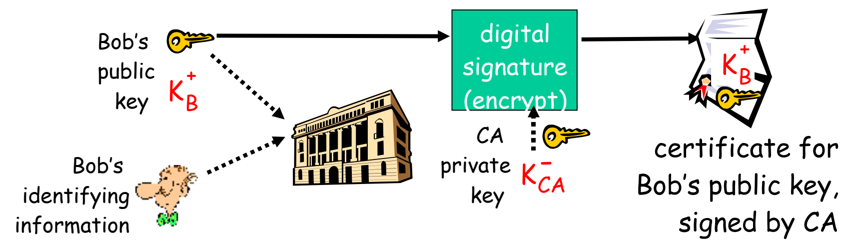
Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

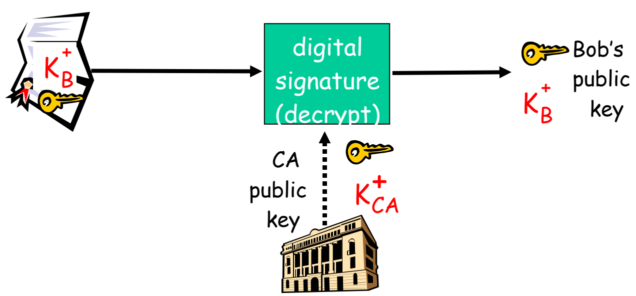
Certification Authorities

- 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"



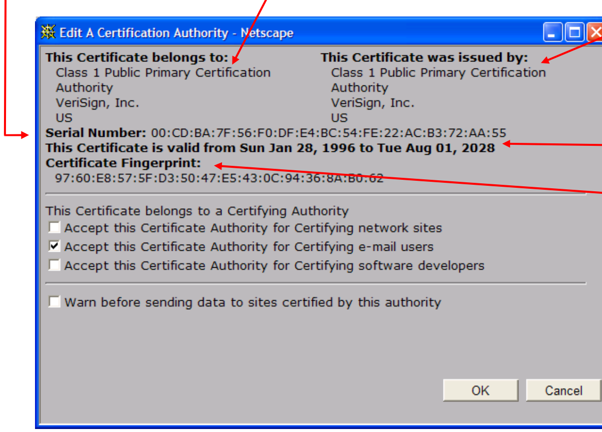
Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (from Bob or elsewhere).
  - Applies CA's public key to Bob's certificate, get Bob's public key



Certificate Elements

- 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

Network Security: Summary

- Key concepts:
  - Confidentiality: keeping it secret
  - Authentication: ensuring the origin
  - Integrity: making it tamper-proof
  - Availability
- Symmetric vs public keys
  - Symmetric: fast; requires shared secret
  - Public: computationally expensive; no shared secret
- Cross-layer issue:
  - Application layer: secure e-commerce transactions, remote login, etc. (SSL “https”)
  - Network layer: ensure validity of routing updates, etc. (IPSEC)
  - Physical layer: protect your wireless home network, etc.