

COMP 243

Distributed Systems

Privacy & Authentication

Kevin Jeffay

Department of Computer Science
University of North Carolina at Chapel Hill

jeffay@cs.unc.edu

October 16, 2000

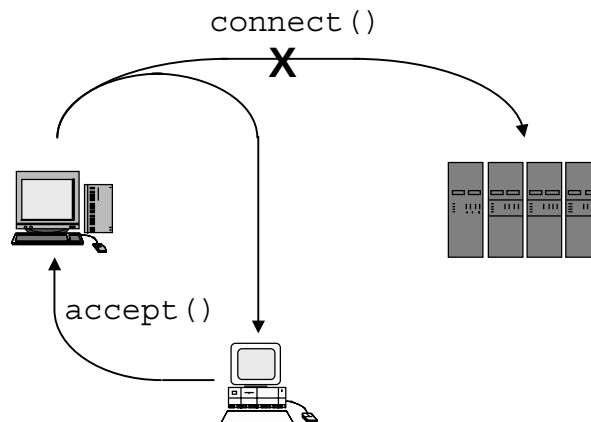
<http://www.cs.unc.edu/~jeffay/courses/comp243f00>

1

Security, Privacy, & Authentication

Introduction

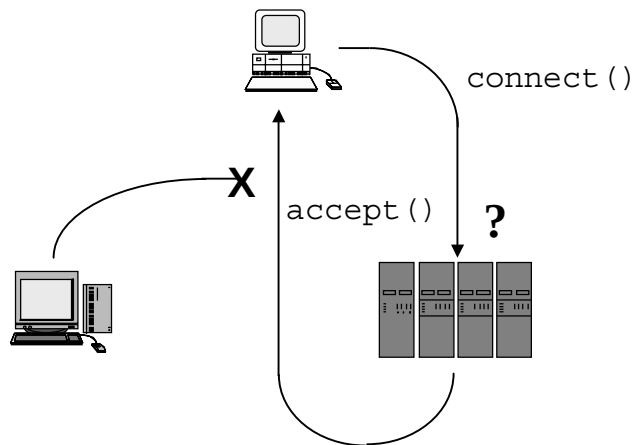
- ◆ How does a client really know which server it is connected to?



2

Security, Privacy, & Authentication Introduction

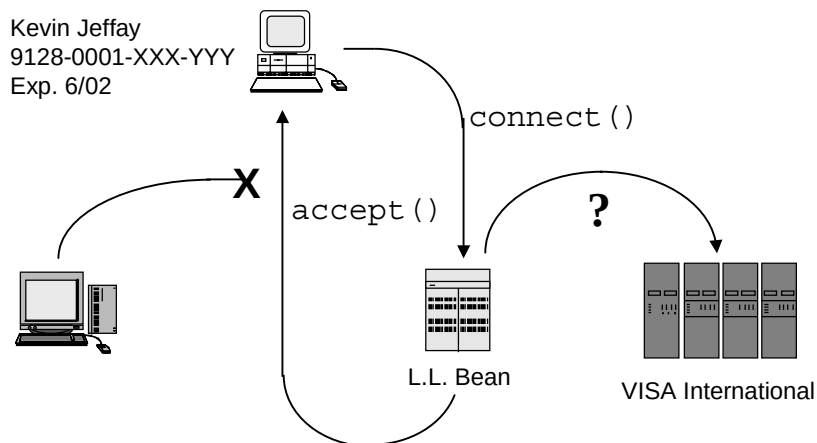
- ◆ How does a server really know to which clients it is connected?



3

Security, Privacy, & Authentication Introduction

- ◆ How can a server prove a client made a specific request?
- ◆ How can a client prove a server gave a specific response?

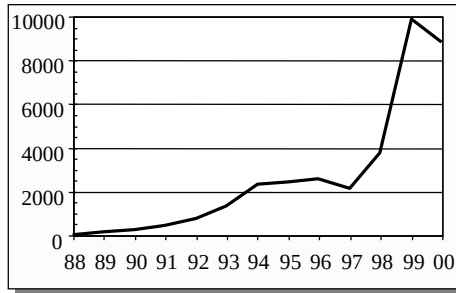


4

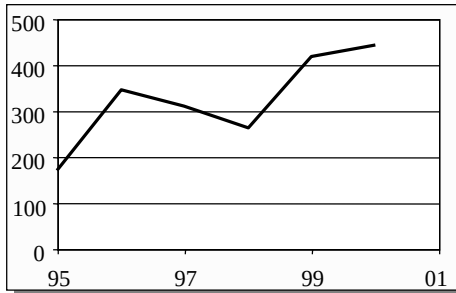
Security in a distributed system

How common are attacks?

◆ Some stats from CERT



Security incidents handled by CERT



Vulnerabilities reported to CERT

5

Security, Privacy, & Authentication Requirements

- ◆ We require technology to realize *secure, private* communication *channels*
 - » And the ability to send *signed* messages
- ◆ Solution: “A *judicious combination of mathematics and muddle*”
 - » Cryptography
 - ❖ The mathematics of secrets and authentication
 - » Protocols
 - ❖ End-to-end protocols for authentication and privacy

6

Security, Authentication, and Privacy

There ain't no silver bullet...

“Whenever anyone says that a problem is easily solved by cryptography, it shows that he doesn't understand it.”

— Roger M. Needham

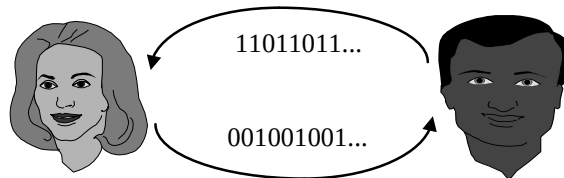


7

Requirements/Goals

The secure channel abstraction

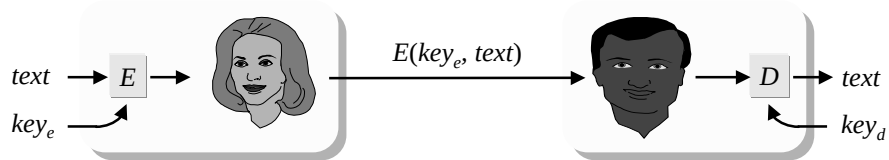
- ◆ The *secure channel* abstraction —A communications channel between two principals *Alice* and *Bob*, wherein:
 - » Data can't be altered without detection
 - ❖ Cryptography & protocols
 - » Data can't be read by unintended recipients
 - ❖ Cryptography
 - » Data is attributed to the correct originator, who can't disown it
 - ❖ Cryptography & system administration



8

Cryptography Basics

Message encryption & decryption



- ◆ Assume two functions of two variables
 key — an encryption “key”
 $text$ — a message
 $E(key, text)$ — encryption function
 $D(key, text)$ — decryption function
- ◆ $E(key_e, text)$ encrypts a message
- ◆ $D(key_d, E(key_e, text))$ decrypts an encrypted message
- ◆ If $key_d = key_e$, the cryptosystem is called *symmetric*, otherwise it is *asymmetric*

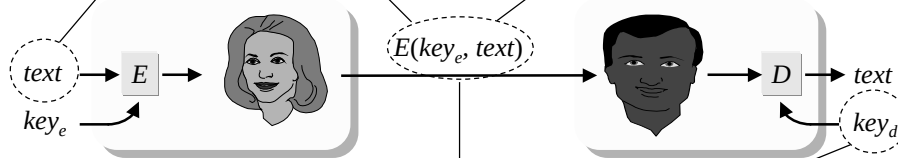
9

Properties of functions E and D

“One way” functions

Given the $text$ of the message and $E(key_e, text)$, it must be difficult to recover key_e

Given $E(key_e, text)$, it must be *computationally infeasible* to recover the $text$ of the message without knowledge of key_d



If $key_d \neq key_e$, then given one key, it must be difficult to recover the other

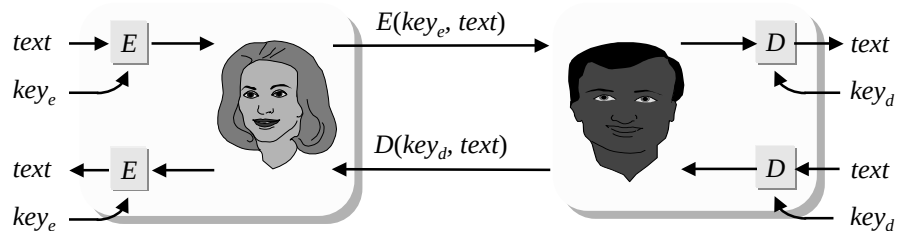
10

Properties of functions E and D

“Reversible” functions

◆ Functions E and D are interchangeable

- » D can be used to encrypt messages
- » E can be used to decrypt messages

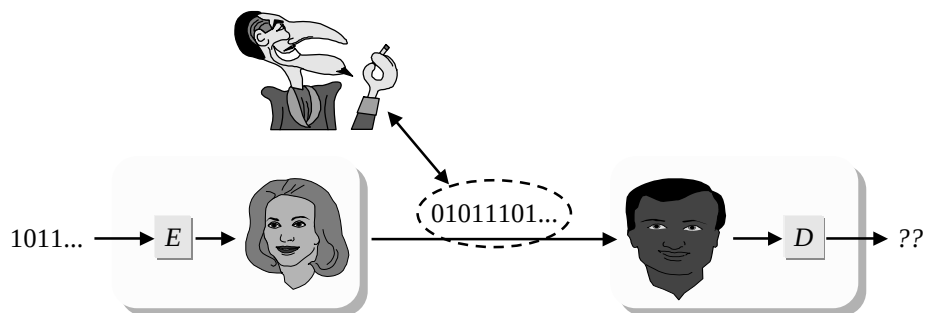


$$D(\text{key}_d, E(\text{key}_e, \text{text})) = E(\text{key}_e, D(\text{key}_d, \text{text})) = \text{text}$$

11

Issues in Cryptosystems

Common attacks



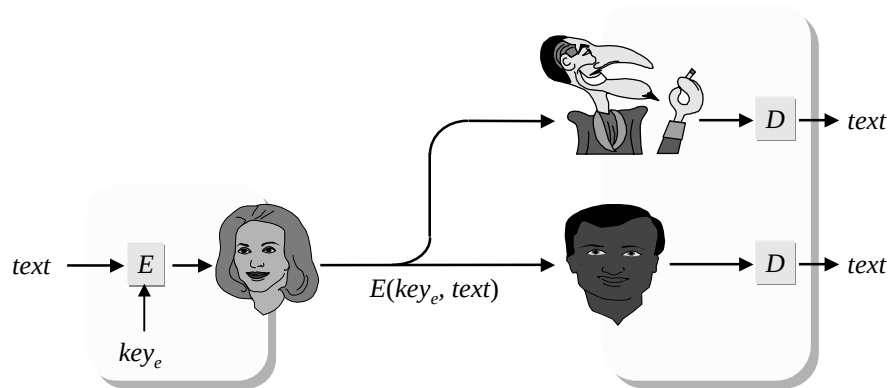
◆ What can a hacker do?

- » Read, copy, and replay messages (bit patterns)
- » Inject arbitrary bit patterns into the message stream
- » Erase arbitrary bit patterns from the message stream
- » Saturate either the transmission medium or a resource necessary for the recipient's operation

12

Issues in Cryptosystems

◆ Is the message private?

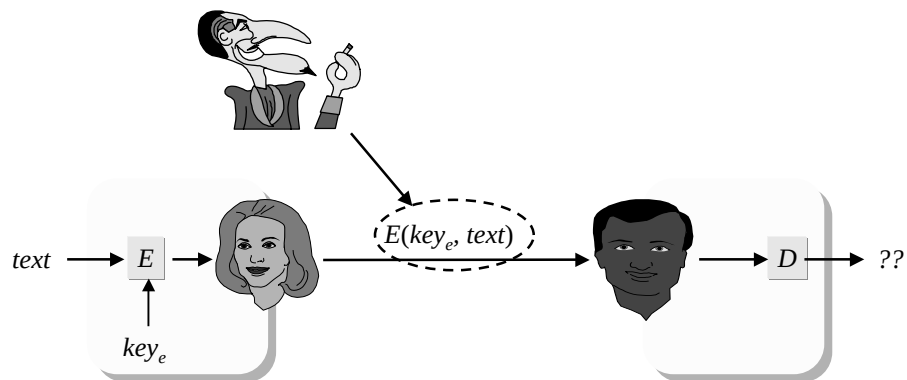


13

Issues in Cryptosystems

◆ How does Bob know that the message received is the message Alice sent?

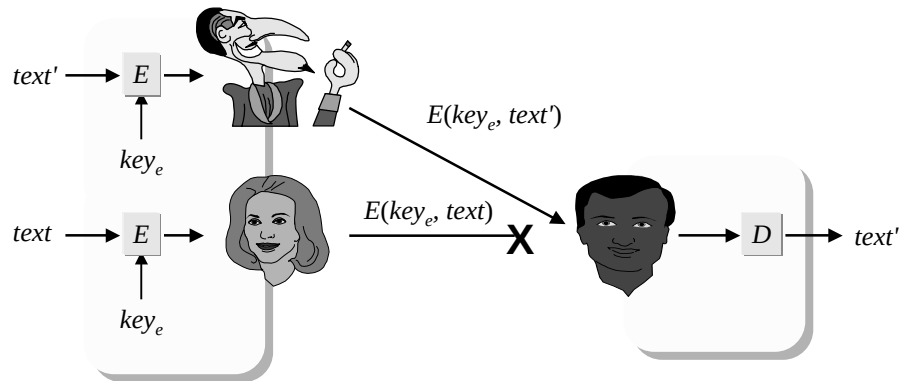
» Have the contents been modified?



14

Issues in Cryptosystems

- ◆ How does Bob know that Alice sent the message?

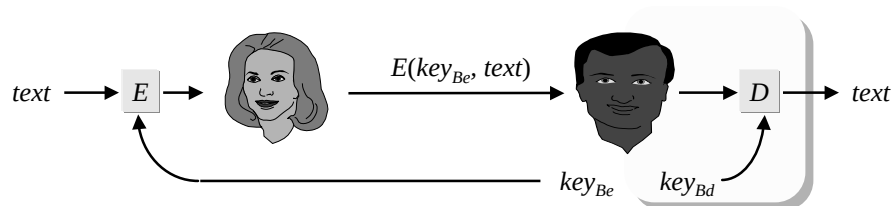


15

Asymmetric (Public Key) Cryptosystems

Sending private messages

- ◆ Each principal maintains a pair of unique keys (key_e, key_d)
 - » key_d is called the *private key*
 - » key_e is called the *public key*
- ◆ For Alice to send a private message to Bob...
 - » Alice looks up Bob's public key key_{Be}
 - » Alice sends Bob $E(key_{Be}, text)$
 - » Bob reads the message $D(key_{Bd}, E(key_{Be}, text)) = text$ using his private key key_{Bd}



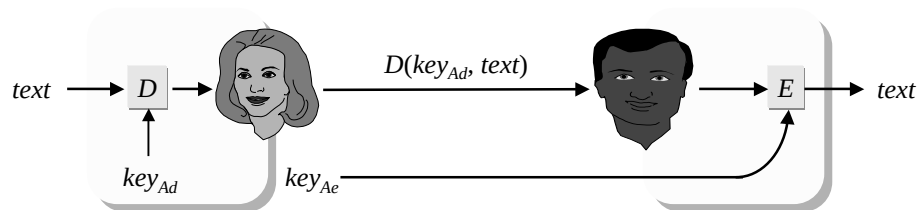
16

Public Key Cryptosystems

Sending “signed messages”

◆ For Alice to send a signed message to Bob...

- » Alice sends Bob $D(key_{Ad}, text)$ using her private key key_{Ad}
- » Bob reads the message $E(key_{Ae}, D(key_{Ad}, text)) = text$ using Alice’s public key key_{Ae}



17

Public Key Cryptosystems

Sending signed, private messages

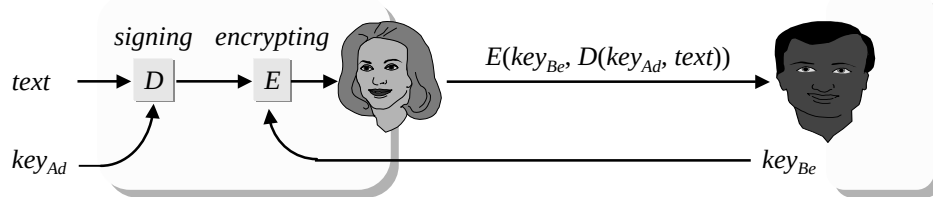
- ◆ But, the private message is “unsigned”,
 - » The intruder also has access to Bob’s public key and could spoof a message from Alice
- ◆ And the signed message is not private
 - » The intruder has access to Alice’s public key and could read the content

18

Public Key Cryptosystems

Sending signed, private messages

- ◆ For Alice to send a signed, private message to Bob...
 - » Alice signs her message to Bob using her private key
 - ❖ $D(key_{Ad}, text)$
 - » Alice looks up Bob's public key
 - » Alice encrypts her signed message
 - ❖ $E(key_{Be}, D(key_{Ad}, text))$
 - » Alice sends the signed, encrypted message to Bob

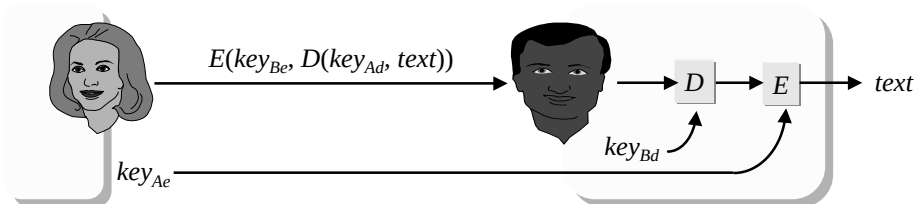


19

Public Key Cryptosystems

Sending signed, private messages

- ◆ For Bob to read a signed, private message from Alice...
 - » Bob decrypts the signed message
 - $D(key_{Bd}, E(key_{Be}, D(key_{Ad}, text))) = D(key_{Ad}, text)$
 - using his private key key_{Bd}
 - » Bob reads Alice's message $E(key_{Be}, D(key_{Ad}, text)) = text$ using her public key key_{Be}

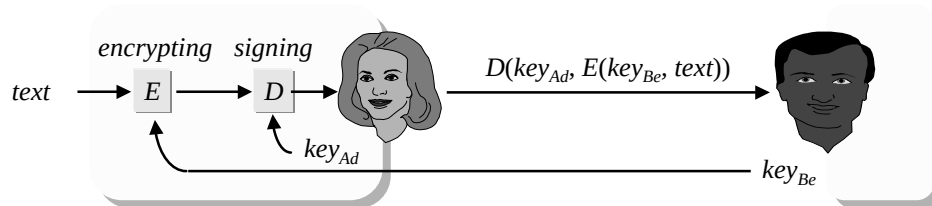


20

Public Key Cryptosystems

Sending signed, private messages

- ◆ What if Alice encrypted the messages in the reverse order?
 - » What if Alice encrypts with Bob's public key before signing the message using her private key?

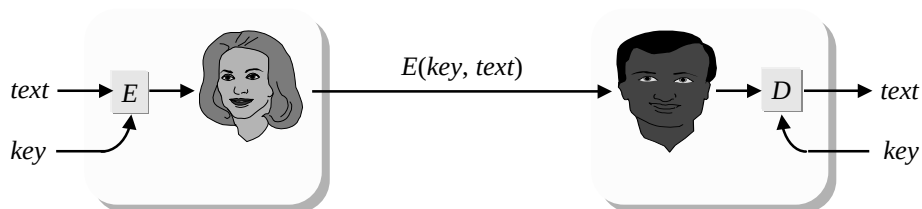


21

Symmetric (Private Key) Cryptosystems

Sending signed, private messages

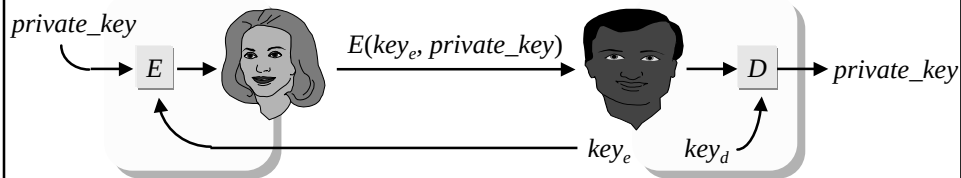
- ◆ For Alice to send a signed, private message to Bob...
 - » Bob and Alice share a private key *key*
 - » Alice sends Bob $E(key, text)$
 - » Bob reads the message $D(key, E(key, text)) = text$ using their shared key *key*



- ◆ How do Bob and Alice get a shared, private key?

22

Bootstrapping a Private Key Cryptosystem



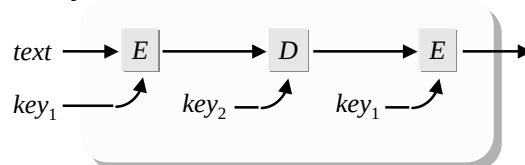
- ◆ Use a public key system to exchange a private key

23

Symmetric v. Asymmetric Cryptosystems

Security of DES encrypted data

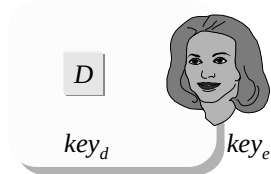
- ◆ Brute force searches of the DES key space
 - » ≤ 7 hours on a \$1 million massively parallel machine
 - » Crack a UNIX password in ≤ 15 minutes
- ◆ “Smart” search
 - » Crack a 56 bit key with 2^{47} selected DES encryptions
- ◆ Solution: longer keys!
 - » “Triple DES”: 3 DES encryptions with either a 112 or 168 bit key



24

Symmetric v. Asymmetric Cryptosystems

Security of RSA encrypted data



Private key = (d, n) where

- $n = p \times q$,
- $\text{GCD}(d, (p-1)(q-1)) = 1$

$D(\text{key}_d, \text{msg}) = \text{msg}^d \bmod n$

◆ Breaking public key systems reduces to factoring integers

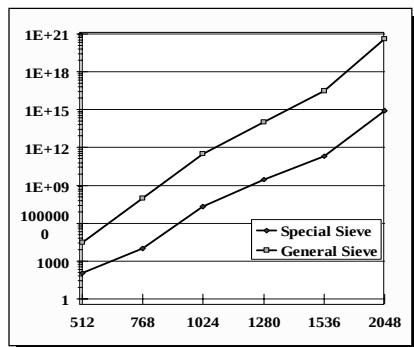
- » Rivest, 1977: “Factoring a 125 digit number will take 40 quadrillion years”
- » 1994: A 129 digit integer was factored in 8 months using the idle capacity on 1,600 workstations (5,000 MIPS-years)
- » Current conjecture: a 512-bit integer could be factored in less than 1 year on a 1,800-node Intel Paragon (56,000 MIPS with i860 processors)

25

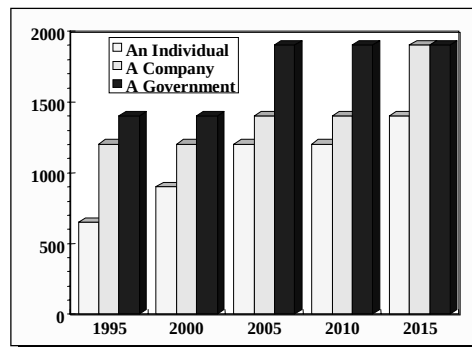
Security of DES & RSA Encrypted Data

Choosing an RSA public key today

MIPS-years need to factor keys of length x -bits



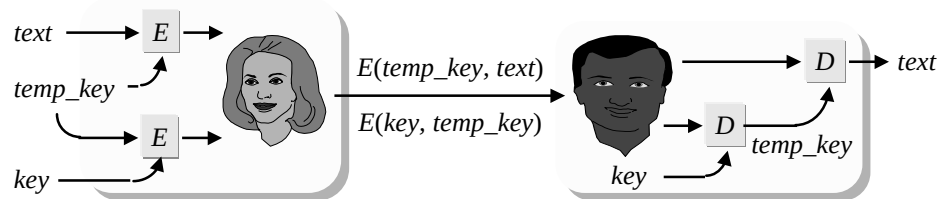
“Recommended” key lengths (in bits) assuming in year x your enemy is...



26

Principles of Using Cryptosystems

Avoid using long-lived keys

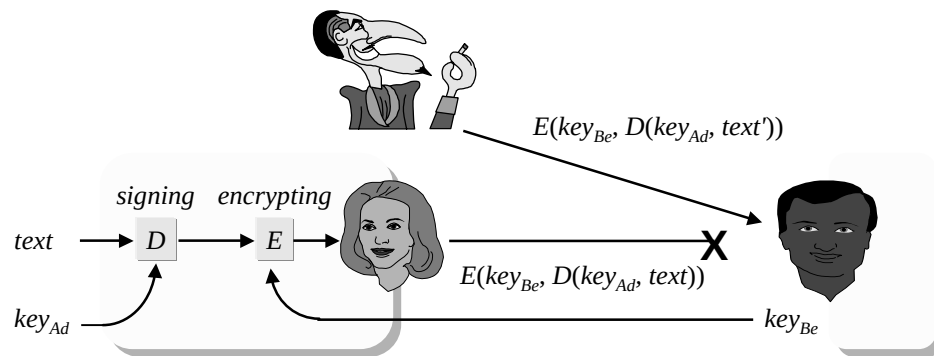


- ◆ Generate a temporary key and transmit
 $E(key, temp_key)E(temp_key, text)$
 rather than
 $E(key, text)$

27

Principles of Using Cryptosystems

Challenge a message sender when unsure of their identity

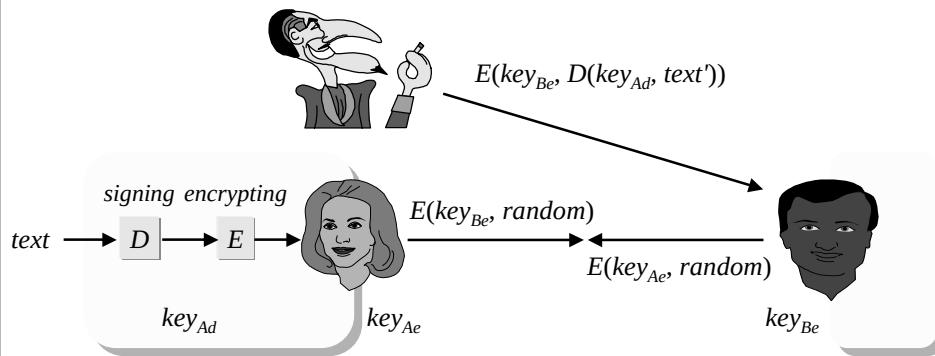


- ◆ Challenge a sender by requiring them to encrypt a message whose contents are known before encryption

28

Principles of Using Cryptosystems

Challenge a message sender when unsure of their identity

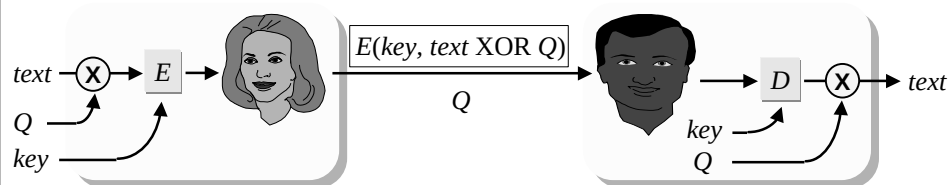


- ◆ What does Bob know when he receives the challenge response?

29

Principles of Using Cryptosystems

Avoid encrypting text of a third party's choosing



- ◆ When challenged:
 “if you’re really Alice, encrypt the message ‘ABCDEFGH’ ”
 don’t transmit
 $E(key, challenge)$
 transmit
 $E(key, challenge \text{ XOR } Q) \text{ } Q$
 where Q is a random number

30

Principles of Using Cryptosystems

Defending against other attacks

- ◆ Need to incorporate “checksums” or *message digests* to detect missing/permutated blocks
- ◆ Need to incorporate timestamps or logs to defend against *replay attacks*
- ◆ Need to protect against attackers guessing “weak” keys

31

Authentication Protocols

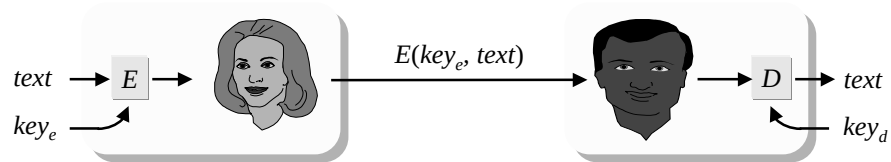
Concept

- ◆ Authentication is concerned with the question “Who said this?”
 - » Did *smithfd@buzzard.cs.unc.edu* really request this file?
 - ❖ Who made the request?
 - » Did *buzzard.cs.unc.edu* really send me this file?
 - ❖ Which server sent the file?
 - » Did Visa really approve the charge to my card?
 - ❖ Can Visa deny they sent it?

32

Authentication Protocols

Outline

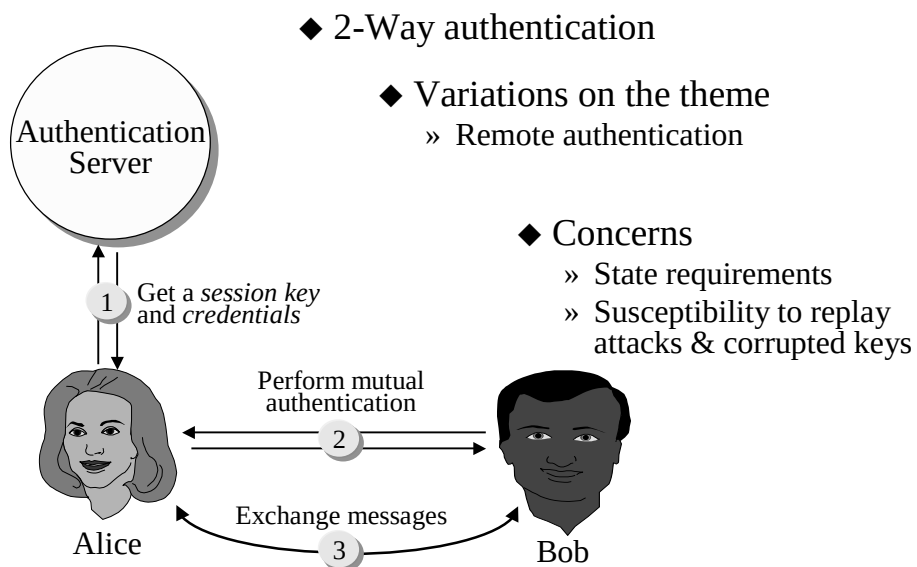


- ◆ Classic Needham & Schroeder authentication model
 - » Theory
- ◆ Kerberos
 - » Practice
- ◆ How safe is safe?
 - » Weaknesses in Kerberos

33

Authentication Protocols

Overview

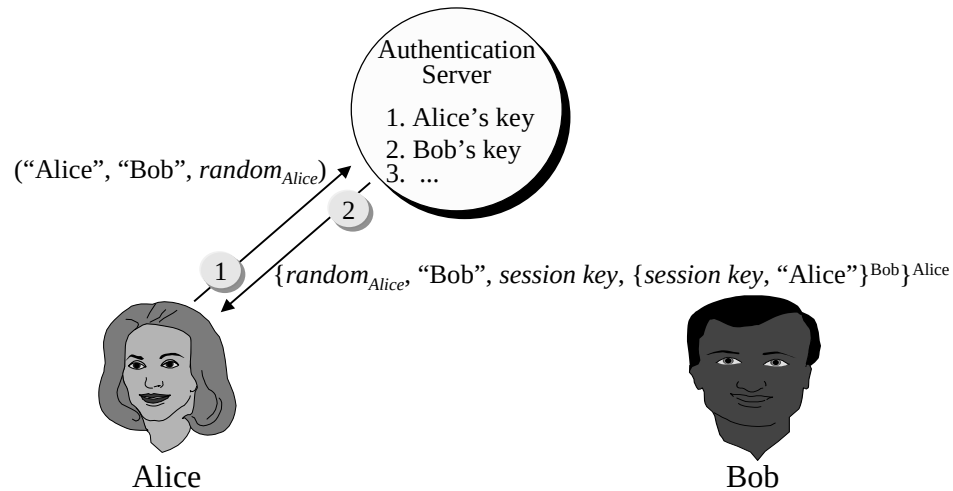


34

2-Way Authentication

Using symmetric cryptography

Step 1 — Obtain a “session key”

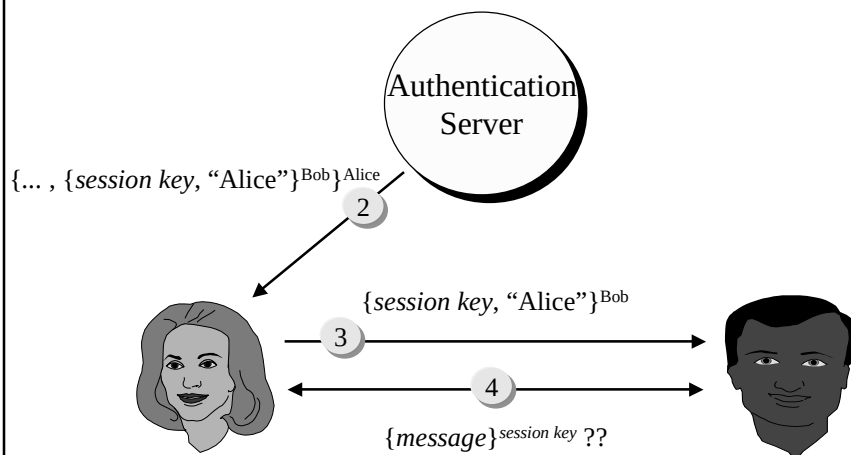


35

2-Way Authentication

Using symmetric cryptography

Step 2 — Alice introduces herself to Bob

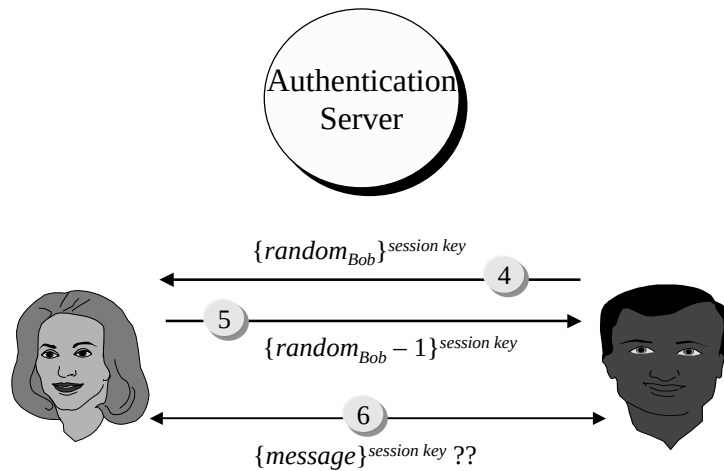


36

2-Way Authentication

Using symmetric cryptography

Step 3 — Bob “challenges” Alice

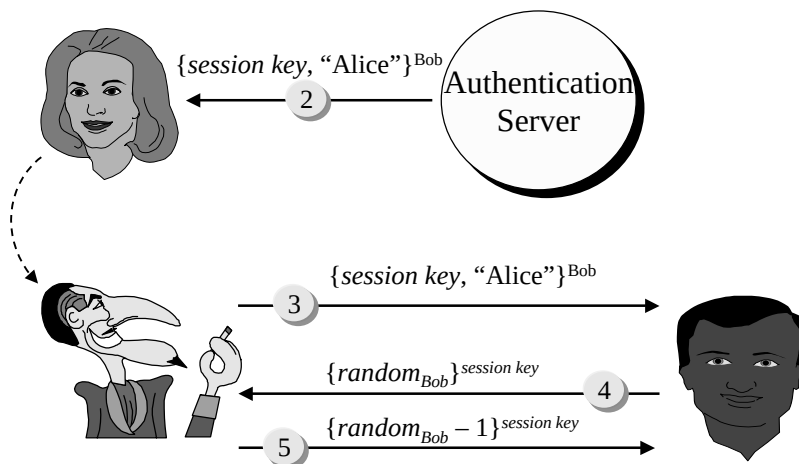


37

2-Way Authentication

Using symmetric cryptography

- ◆ What if Alice allowed the session key to be stolen or a replay attack is under way?

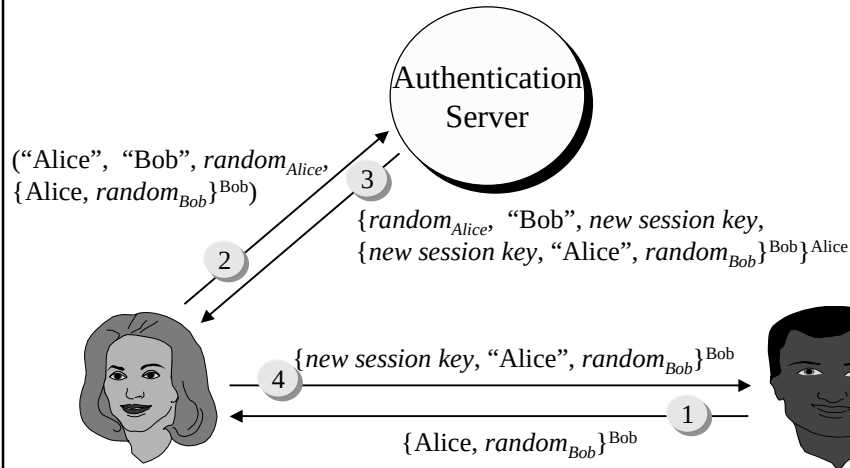


38

2-Way Authentication

Communicating with untrusted parties

Bob needs to challenge Alice by requiring her to reauthenticate herself to the server

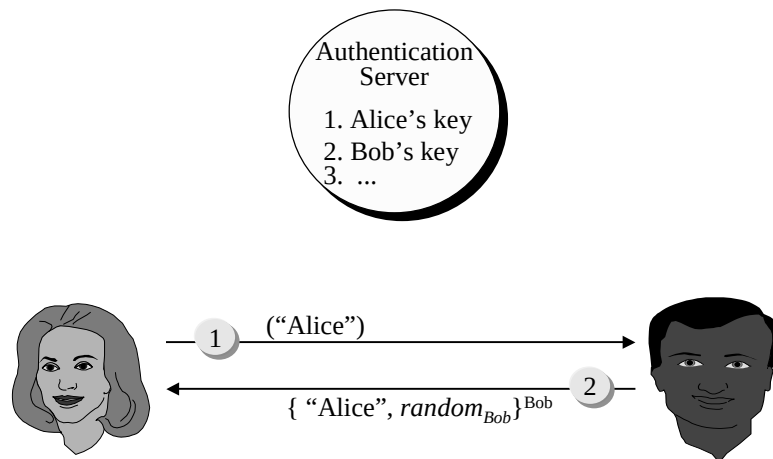


39

2-Way Authentication

Communicating with untrusted parties

Step 0 — Alice requests to communicate with Bob

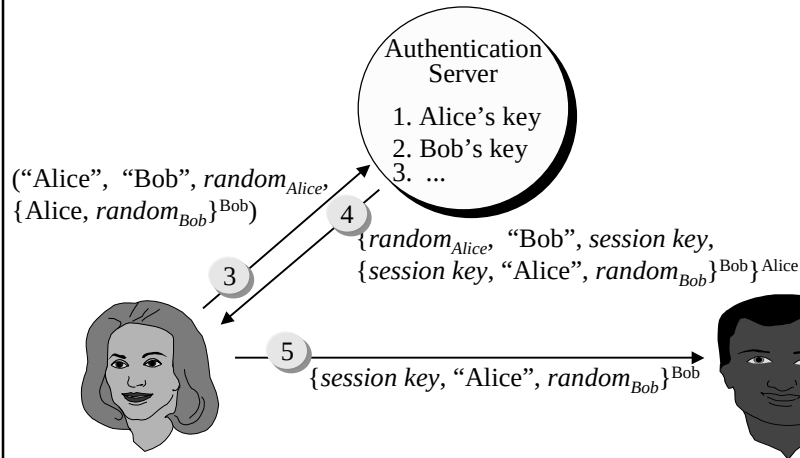


40

2-Way Authentication

Communicating with untrusted parties

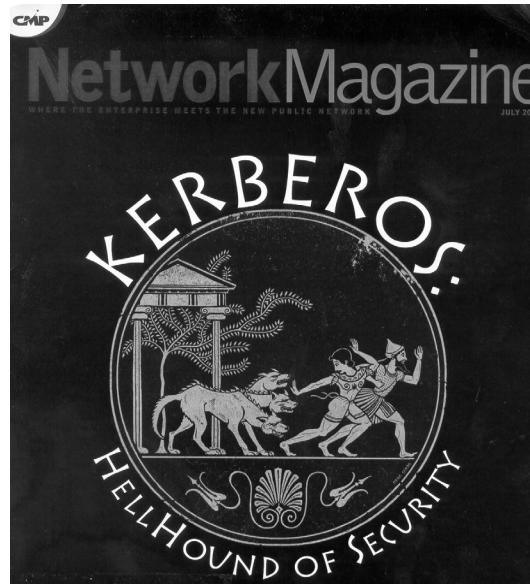
Step 1 — Obtain a session key



Step 2 — Alice authenticates herself to Bob

41

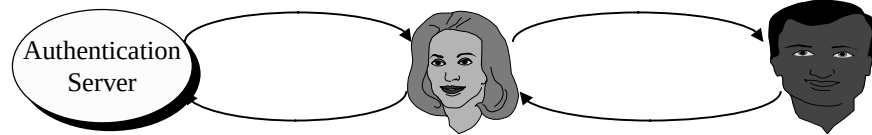
Introduction to Kerberos



42

The Kerberos Authentication System

Overview



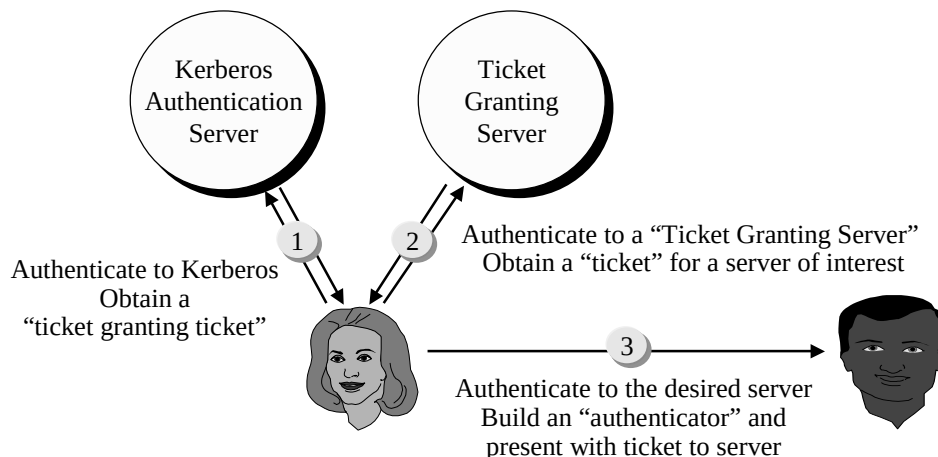
- ◆ An implementation of the Needham & Schroeder protocol
 - » Encryption is based on DES
 - » Timestamps added to defend against replay attacks
- ◆ Used in OSF Distributed Computing Environment (DCE) and separately in AFS
- ◆ Basic services:
 - » Authentication at time of connection establishment
 - » “Safe (authenticated) messages”
 - » “Private (encrypted) messages”

43

Authentication in Kerberos

A three step process

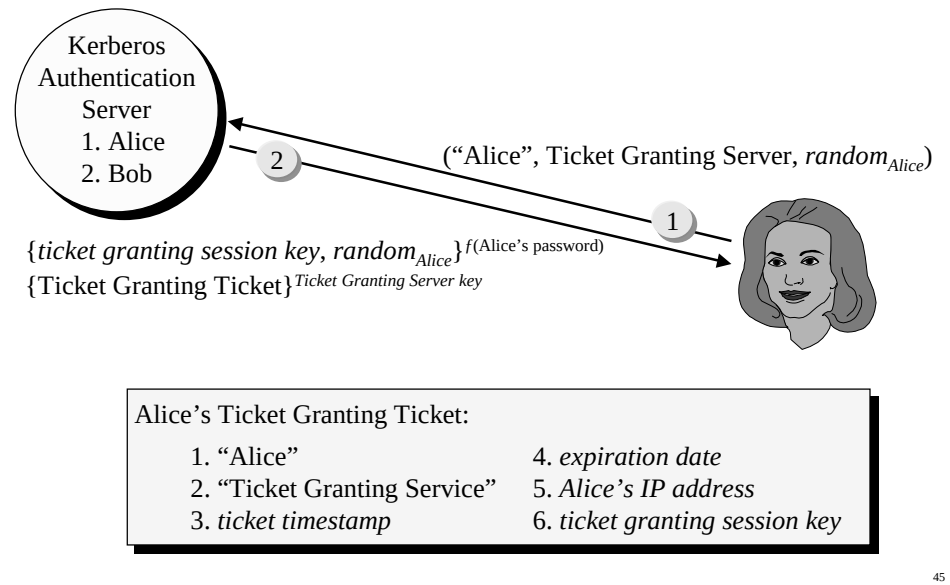
- ◆ User authentication, and client/server session key generation functions separated



44

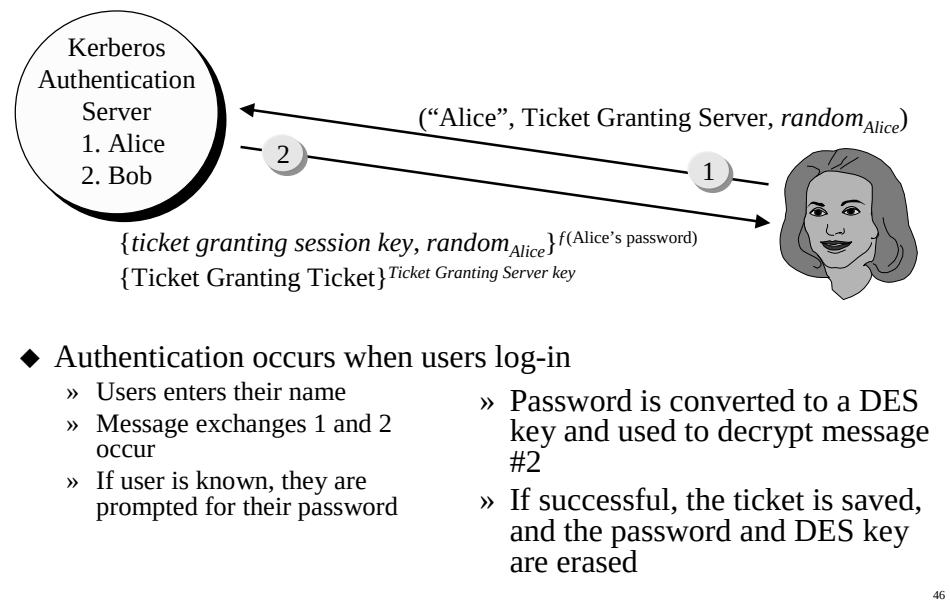
Authentication in Kerberos

Step 1 — Authenticating to Kerberos



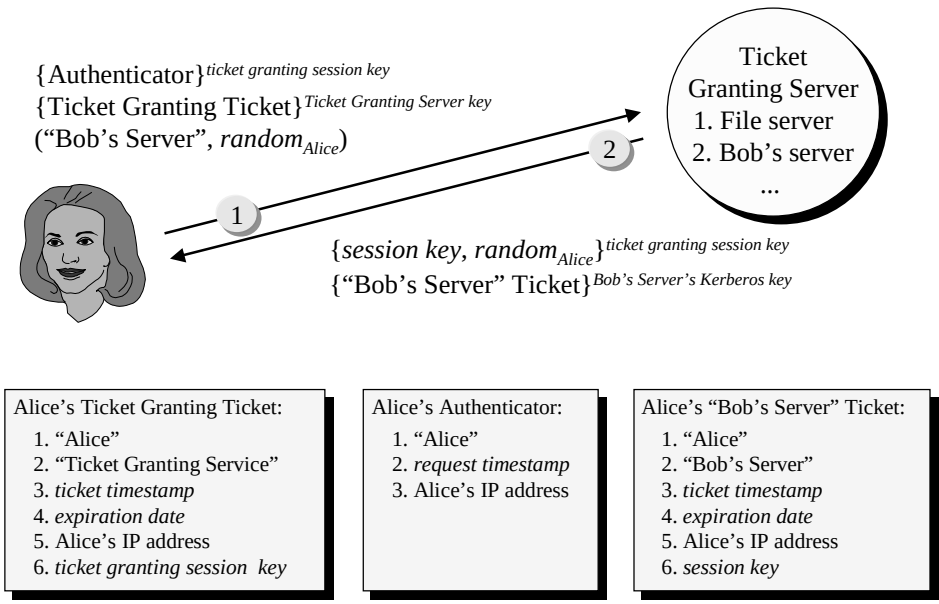
Authentication in Kerberos

login



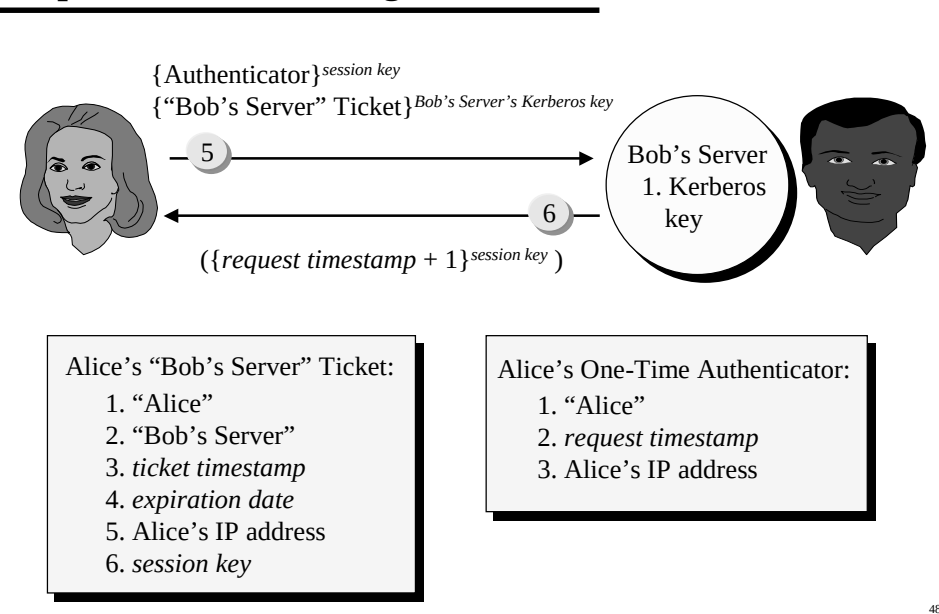
Authentication in Kerberos

Step 2 — Authenticating to the Ticket Granting Server



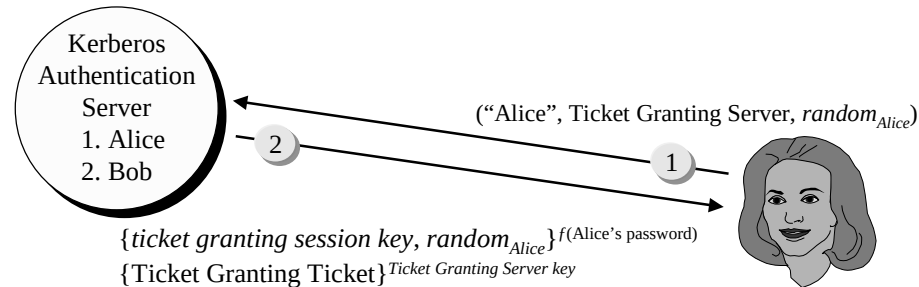
Authentication in Kerberos

Step 3 — Authenticating to Bob's server



Weaknesses of Kerberos

Security of user passwords

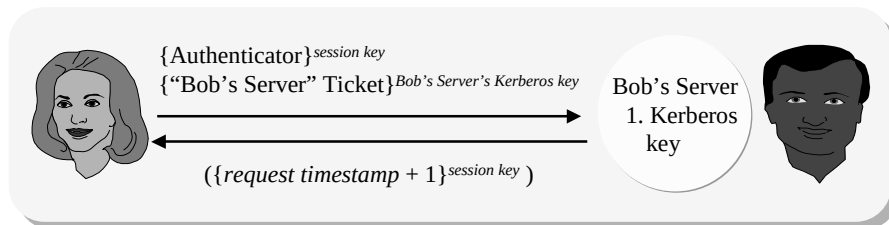


- ◆ Is it really easy to guess a user’s password?
- ◆ login spoofing attacks
 - » Physical security of the end-system cannot be guaranteed

49

Weaknesses of Kerberos

Security against replay attacks



- ◆ Do timestamps effectively guard against replay attacks?
 - » Ideally one should not rely on tightly synchronized clocks
 - » Authenticators can be recorded and reused for 5 minutes
- ◆ Solution: Servers are allowed to cache past requests with still valid timestamps
 - » Caching authenticators problematic for servers using sockets or RPC
- ◆ Are time services secure?

50