Topic 9: Entity Authentication

Apply authentication techniques to counter impersonation or masquerading attacks

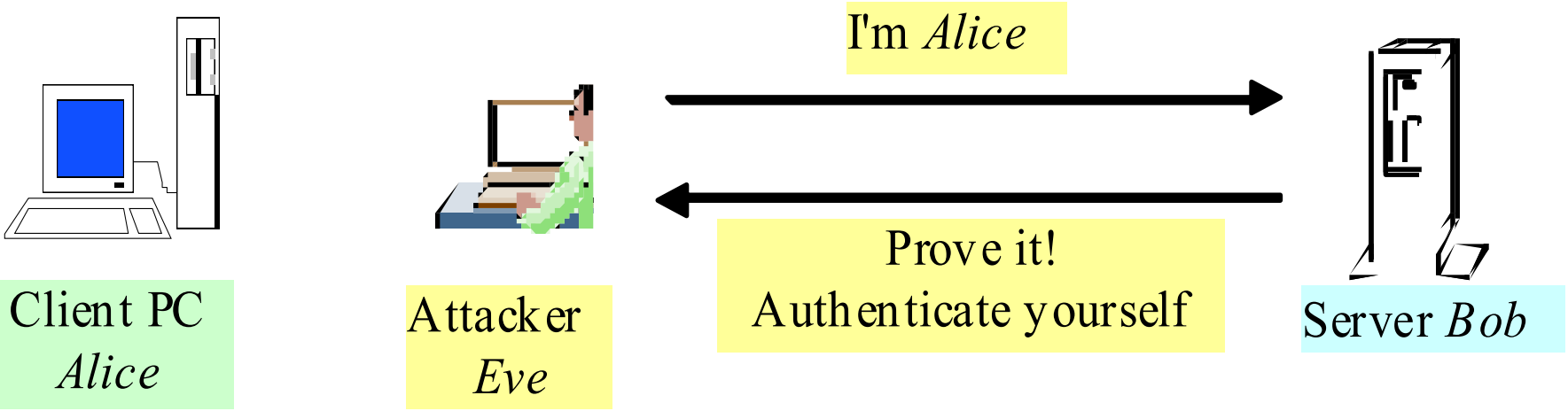
Source: Stallings book, Chapter 15 covers some of the content
An excellent document on e-authentication: NIST Special Publication 800-63-2: Electronic Authentication Guideline available here at: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf

Overview

- Part 1
  - Authentication Overview
  - Password-based Authentication in General
- Part 2
  - Unix authN Solution
- Part 3
  - Challenge-Response (C/R) AuthN Protocols
  - Token-based Authentication
- Part 4
  - Enterprise-wide Authentication (SSO – Single Sign On)
  - Conclusions

Authentication Overview - Why do we need it

- Authentication is the process of verifying a claimed **identity.**
- If the communication takes place over a network, how could *Bob* be assured that the person claiming to be *Alice* really is *Alice*? If *Bob* is a server, an attacker may be able to log in as *Alice* to access data and services, or to use her account to launch further attacks.



Authentication Overview - What it is for

- **Authentication**
  - User identification/authentication or entity authentication
    - Mutual authentication
      - Who the user is?
      - Which system? – you could talk to anybody
    - The user identity is a parameter in access control decisions - authorisation
    - The user identity is recorded when logging security-relevant events in an audit trail – accounting
    - This is the so called AAA services
  - Message authentication - *we did this already*!
    - The message is from the source it claims to be.
    - The message has not been altered or replayed.

Authentication Overview - Methods

- Methods for user identification/authentication:
  - Where you are (location authentication - physical location/specific terminal, e.g. based on IP addresses).
  - Something you know (passwords, PIN).
  - Something you have (keys – soft tokens, and hard tokens (smart cards) - may require special hardware).
  - Something you are (biometrics - fingerprint matching, voice recognition, face recognition, iris scanning, etc - require special hardware).
  - Combined (or multiple) methods may be used for a higher level of assurance.

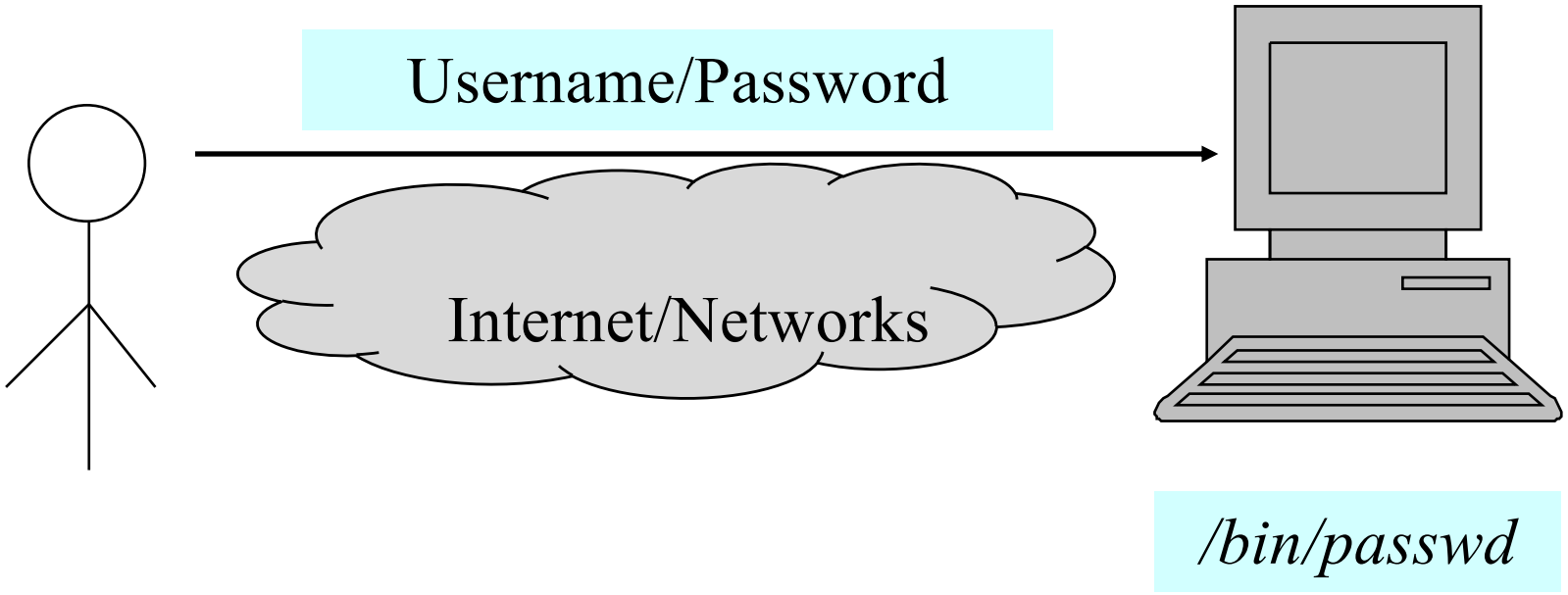
Itself can not support e-authentication

Authentication Overview - Prominent schemes

- Client-server authentication solutions
  - Password-based authentication.
  - Smart-card-based (token-based) authentication.
    - Symmetric key based
    - PKI based - digital signatures and public keys - X.509 certificates.
- Enterprise-wide authentication solutions (the issue of single-sign-on)
  - Kerberos (a password centric solution).
  - RADIUS (a centralised AAA service).
- Shibboleth (authenticating access to multiple enterprises/organisations) – outside the scope of this module.
- Different authentication schemes provide different levels of assurance.
- There is a trade-off between the level of security vs complexity vs cost.

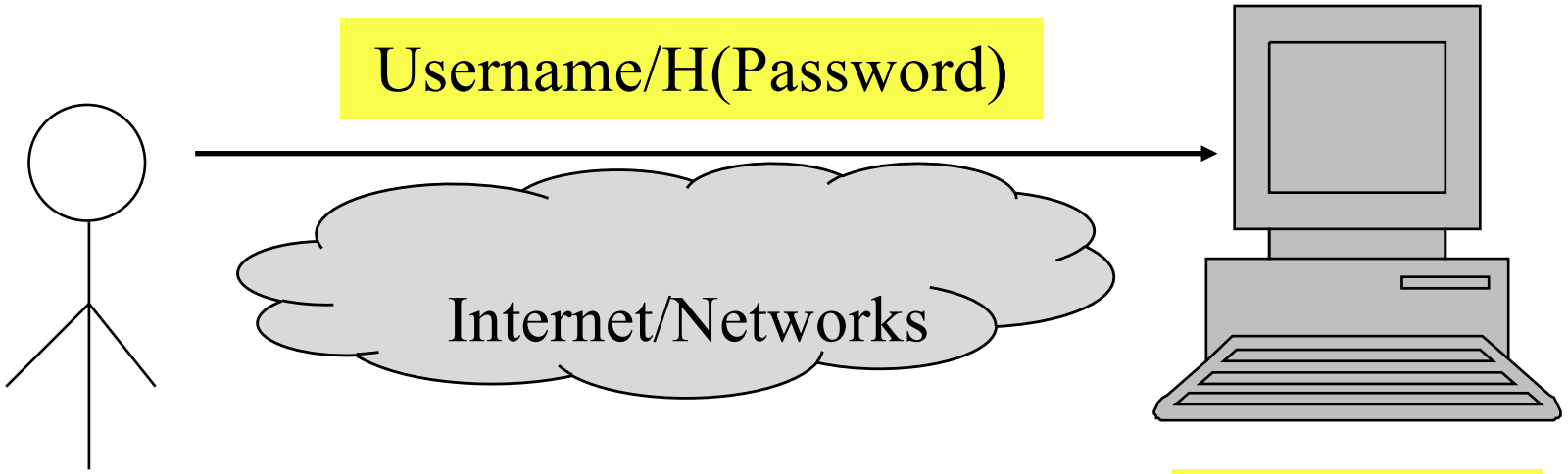
Password-based authentication – Plaintext PW method

- Plaintext passwords are transmitted and stored in the server.
- What are the conditions under which this method can be used securely? What/where are the threats (what is the threat model)?

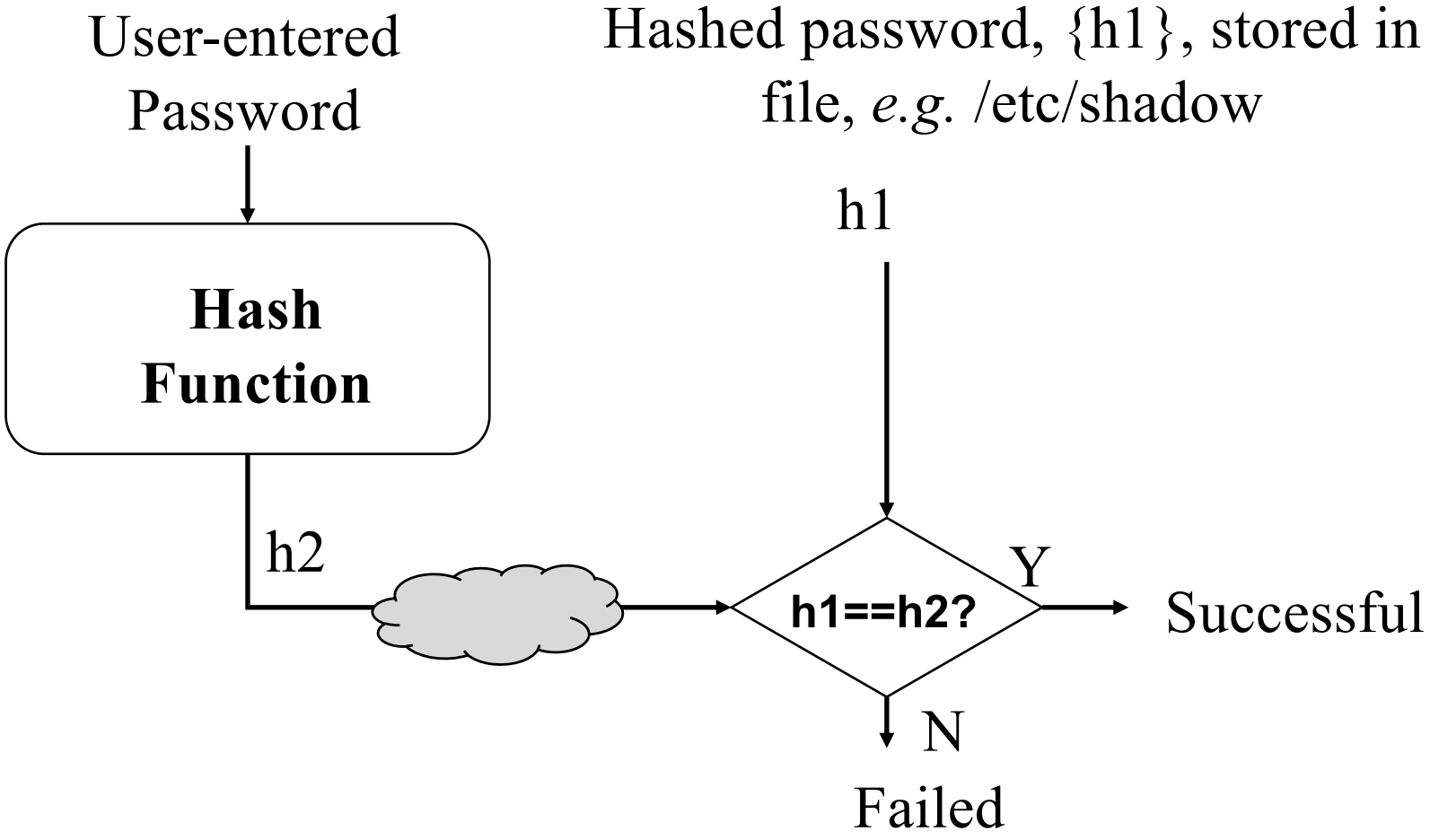


Password-based authentication – Hashed PW method

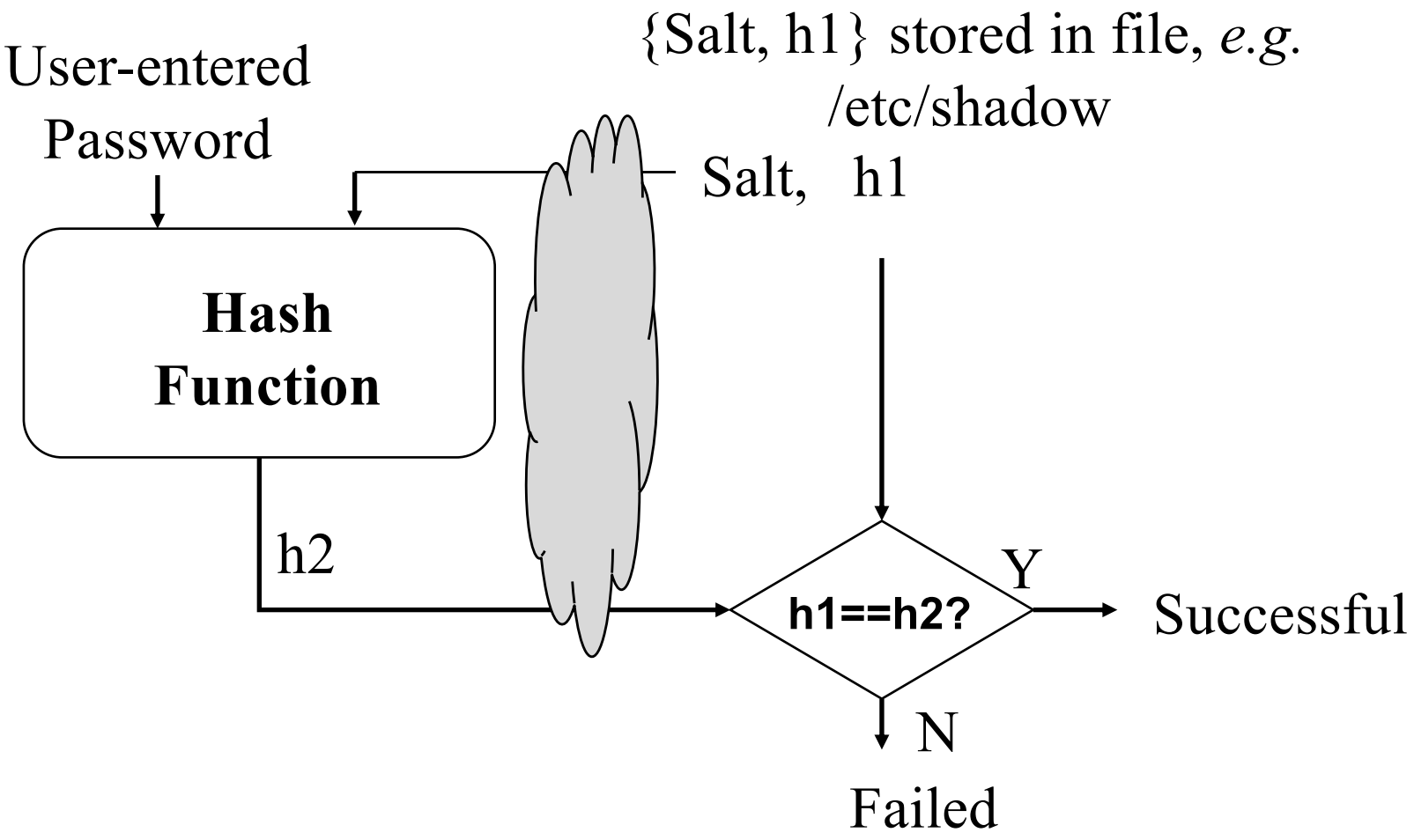
- Hashed passwords are transmitted and stored in the server, but in a root directory.
- What are the conditions under which this method can be used securely? What/where are the threats (what is the threat model)?



Password-based authentication – Hashed PW method



Password-based authentication – Hashed PW with Salt method



Part 2 Overview

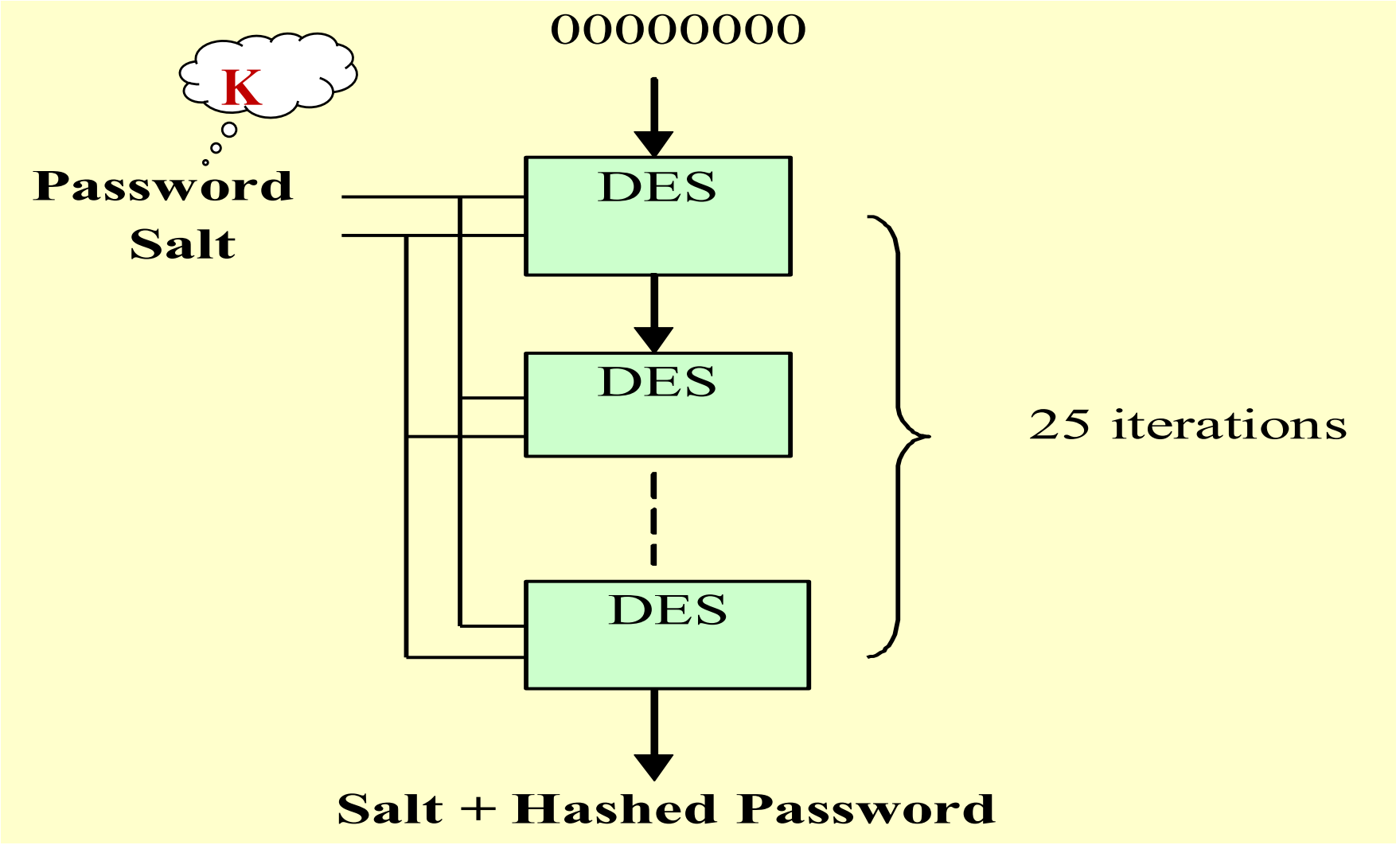
- Unix authN Solution

Unix AuthN Solution – Crypt() algorithm

- Unix system uses the 'Hashed PW with Salt' method.

- Original Hash function is the **Crypt()** algorithm
  - modified based on the DES algorithm.
  - 8 character password form 56-bit key (for each character, 1 bit for parity check and 7 bits for ASCII coding).
  - 12-bit salt to
    - perturb the DES algorithm, so that DES chip can not be used to (dictionary) guess the passwords.
    - make precompiled dictionary attacks harder (by a factor of 4,096).
    - prevent an identical password from producing the same encrypted password.
  - The 64 bits output are unpacked into a string of 11 printable characters, called *the encrypted password*.

Unix AuthN Solution - Crypt() algorithm

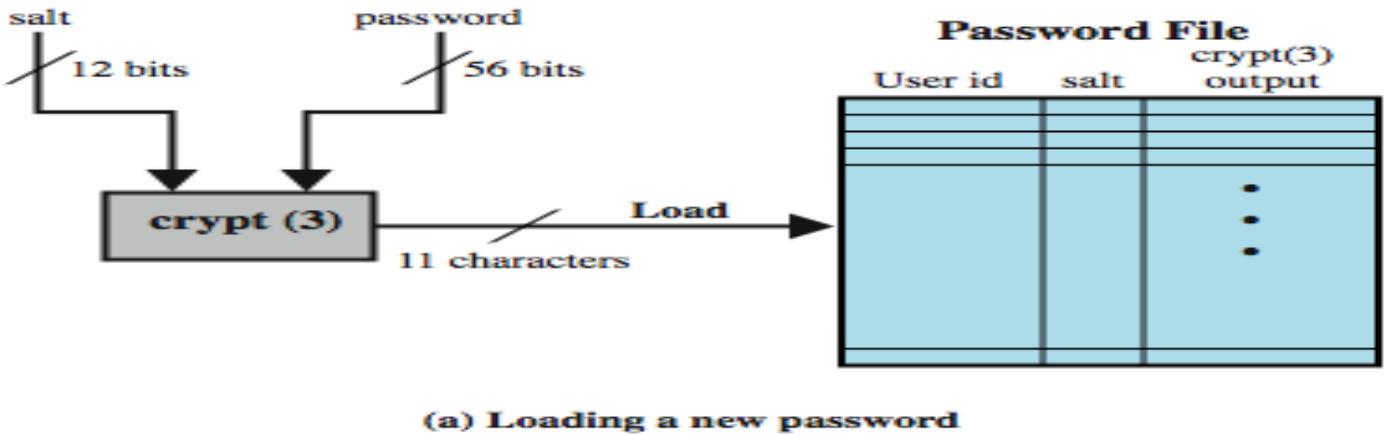


Unix AuthN Solution – AuthN process

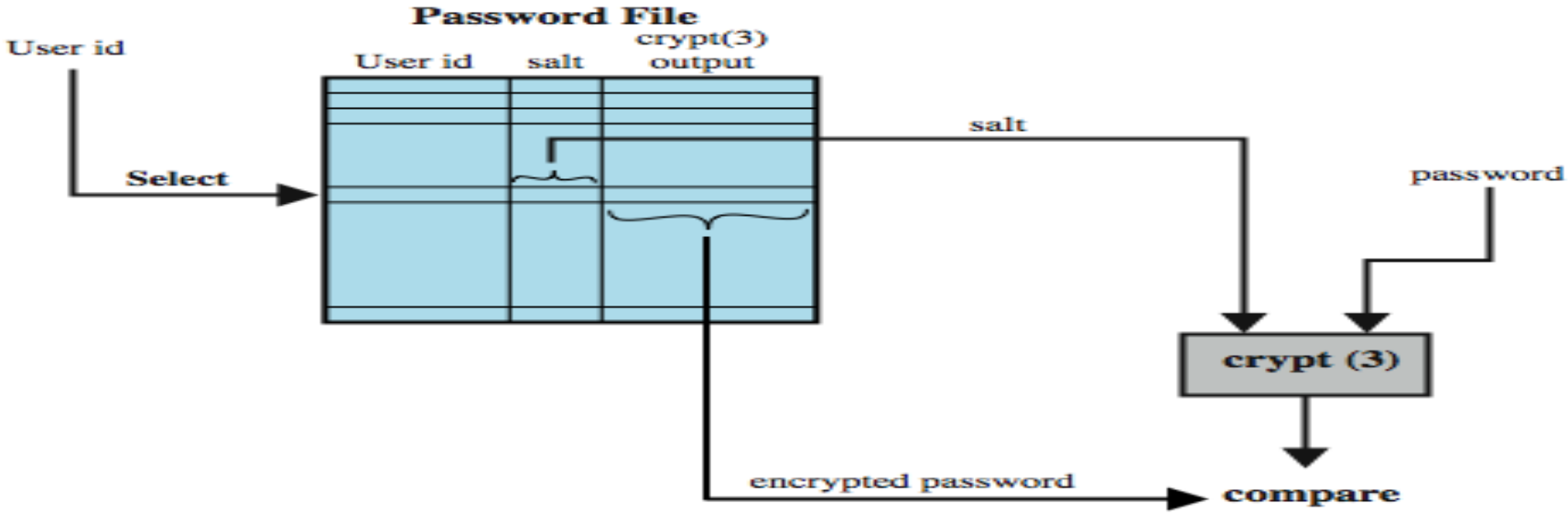
- When a user have an account created or changes the password, the */bin/passwd* program
  - selects a salt based on the time of day; the salt is converted into a two-character string and is stored along with the encrypted password.
  - the password is used as the encryption key to encrypt a block of zero bits using *crypt( )* to generate the encrypted password.

- When a user tries to log in, the program */bin/login* takes the password the user typed, and the salt from the password file, to generate a fresh encrypted password, and compares the newly generated one with the one stored in the server. If the two encrypted results match, the system lets you in.

Unix AuthN Solution – AuthN process

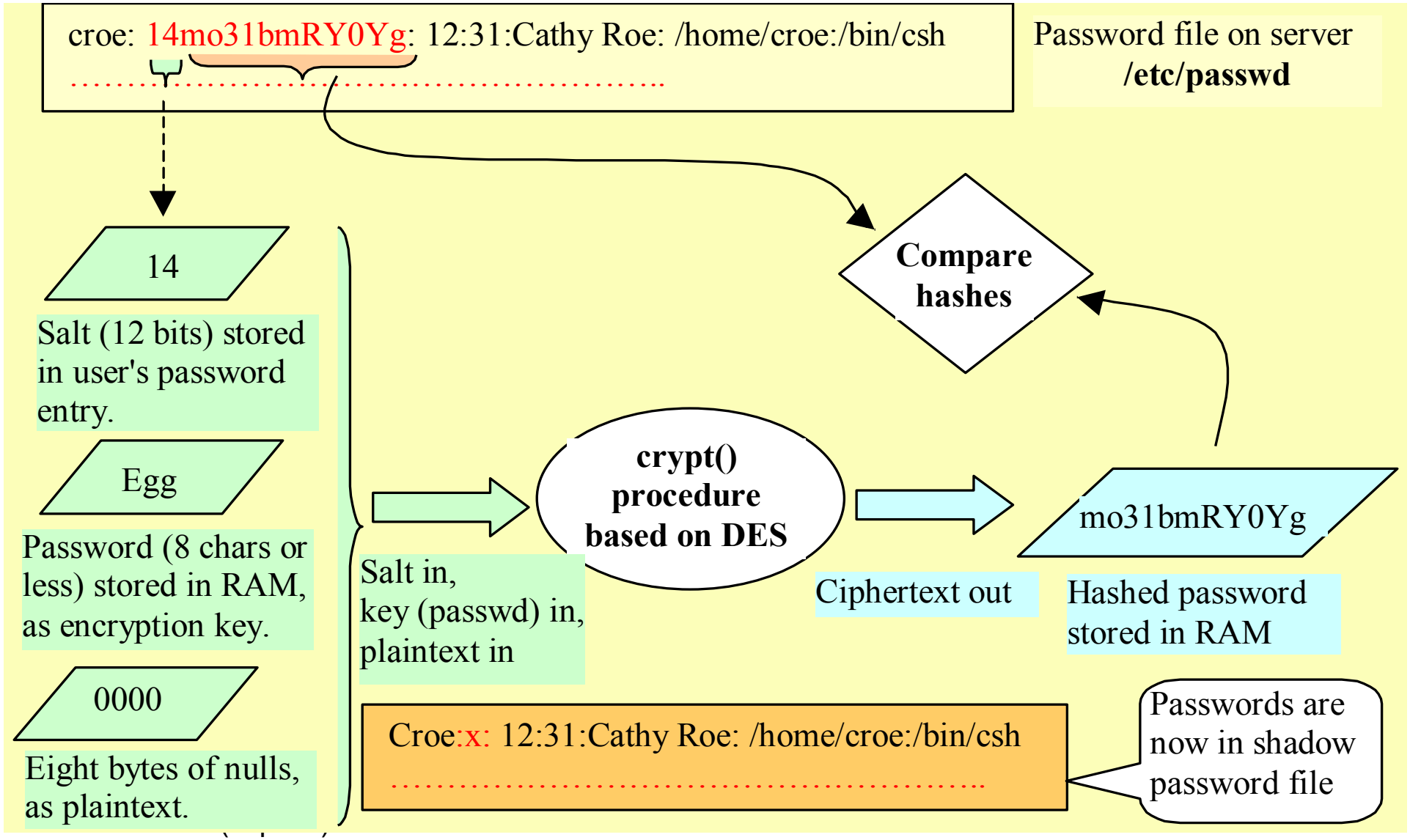


(a) Loading a new password



(b) Verifying a password

Unix AuthN Solution - AuthN process



Unix AuthN Solution – Shadow Password File

- The Unix password file
  - is */etc/passwd*. Each entry in the file is for one account and has several fields separated by colons:
    - User Name (croe): the account name.
    - Password (14mo31bmRY0Yg): the hashed password (mo31bmRY0Yg) preceded by a salt value (14) to be used with the password. Salts make the password
      - more difficulty to guess, and
      - the hashing algorithm slower!
    - User ID (UID=12): a number assigned to this user name for system use in identifying the account.
    - Group ID (GID = 31): a number for the user's group.
    - Home Directory (/home/croe).
    - Shell (/bin/csh): the user's default shell program.

Unix AuthN Solution – Shadow Password File

- **Problem**: /etc/passwd file need to be accessed by processes, so if anyone could get access to the file, then s/he could crack on the passwords at his/er spare time.
- **Countermeasure:** a shadow password file, /etc/shadow, is used, which contains the encrypted passwords, and is put in a root account; put an x (or other placeholder) in the original /etc/passwd file. That is,

/etc/passwd file contains**:**

An example: User1:x:9111:9201:user1:/home/user1:/bin/bash

Meaning: UserName:x *(indicate that the password is stored in the /etc/shadow file)*
:UserID:GroupID:FullName:HomeDirectory:UserShell.

/etc/shadow file contains**:**

An example: User1:1/uTQhcV4$2E…../:13030:0:99999:7:::

Meaning:
UserName:*hashedPassword*:passwdLastChanged:PasswdMayBeChanged:PasswdMustBeChanged:PasswdChangeWarning:DisableAccount:DisabledSince:Reserved.

Unix AuthN Solution - Improved Unix Solution

- But attacks still possible - if you run some software processes with root privileges …, and if the attacker can take over such a program ...

- More problem: An attacker can eavesdrop on a network to get your login ID and encrypted password and later replays (re-send) it to gain access to the system - the replay attack.
  - To perform this attack, the *attacker needs* to
    - eavesdrop on the network (or access to the password file).
    - modify the client/logon software so that it does not encrypt the encrypted -password, but rather replay it directly;
  - *Usually we assume* that
    - the LAN is secure, i.e. eavesdropping can be noticed!
    - You do not bring your own client software in!
  - So we tend to overlook this problem in a LAN environment.

Unix AuthN Solution – Current Unix Solution

- Many *NIX systems now
  - use hash functions such as MD5
  - allows longer password length and longer encrypted passwords
  - use longer salt
  - use PAM (Pluggable Authentication Module) – a flexible way to support the use of many different authentication methods

Part 3 Overview

- Challenge-Response (C/R) AuthN Protocols
- Token-based authentication

C-R AuthN Protocols

- A protocol specifies a set of rules governing the interactions among two/more entities to accomplish a given task.

- For example, an ATM Cash Withdrawal Protocol:

1. Insert ATM card
2. Enter your PIN and make a transaction request
3. PIN Correct?

   Yes – Next step action

   No - Try again (repeated fails will lead to taking your card in)

C-R AuthN Protocols

- A protocol consists of
  - Protocol messages facilitating the communications between the entities.
  - Operations for performing some defined actions (algorithms/functions), including generating/verifying a protocol message.

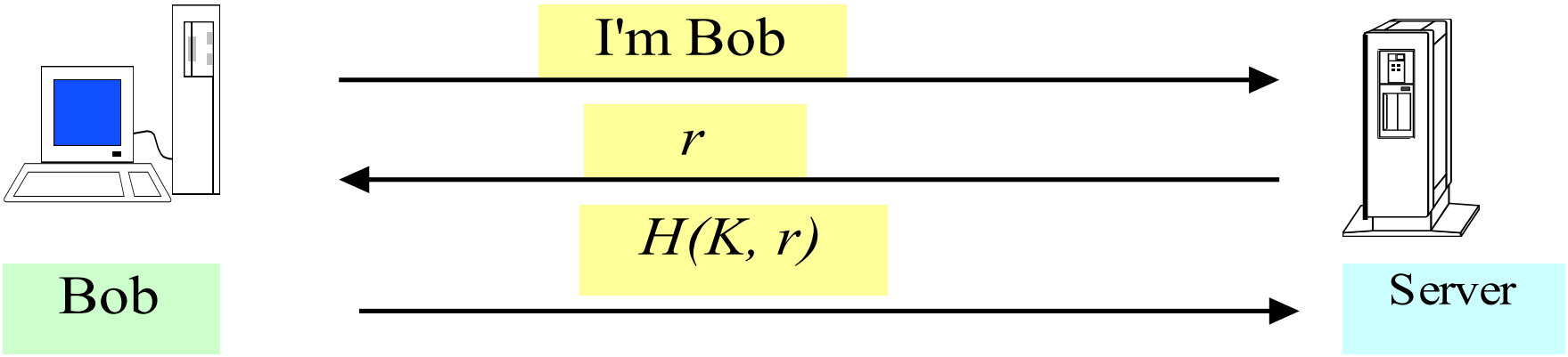
- An AuthN protocol specifies a set of rules governing the interactions among two/more entities in order to accomplish an authN task.

C-R AuthN Protocols

- They vary depending on a number of factors, e.g.
  - What is the threat model?
  - Should it support mutual authentication?
  - Should it establish a session key?
  - What key(s)/functions are used? - public keys, symmetric keys, passwords, hash functions, block ciphers, …
  - Any performance requirement?
  - Any privacy requirement?
  - Should scalability be considered?
  - Any other factors in the underlying environment that should be considered? and so on.

C/R AuthN Protocols – A C/R protocol for OTP authN

- A C/R protocol (Challenge-Handshake Authentication Protocol, CHAP) is based on a secret (password/symmetric key/private key).
- Demonstrate the knowledge of a secret without revealing the secret.
- Hashed value in the response serves as an OTP (OneTimePassword).
- Commonly used between a client and an infrastructure edge device, e.g. remote access server, a VPN server.

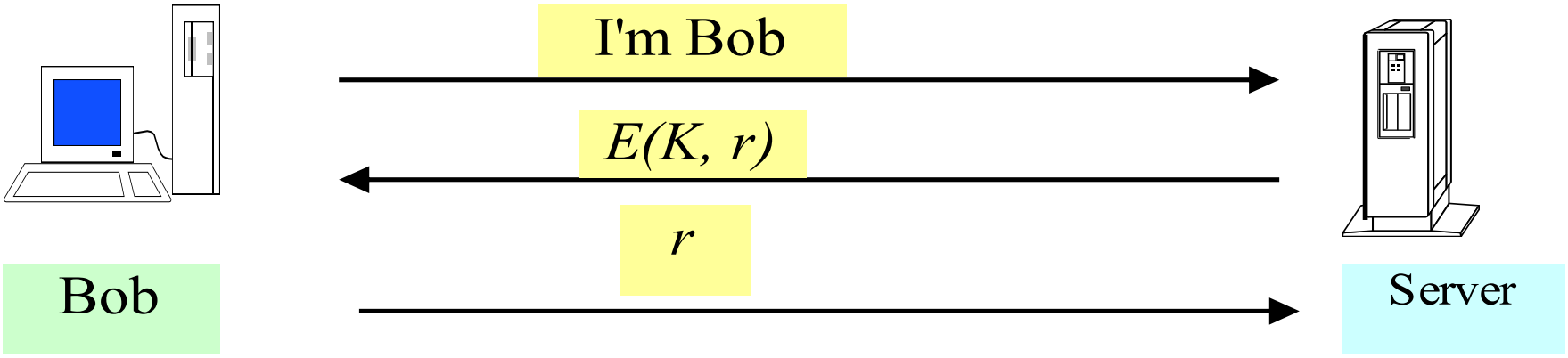


r is a nonce, *K* is a shared key, and H is a hash function

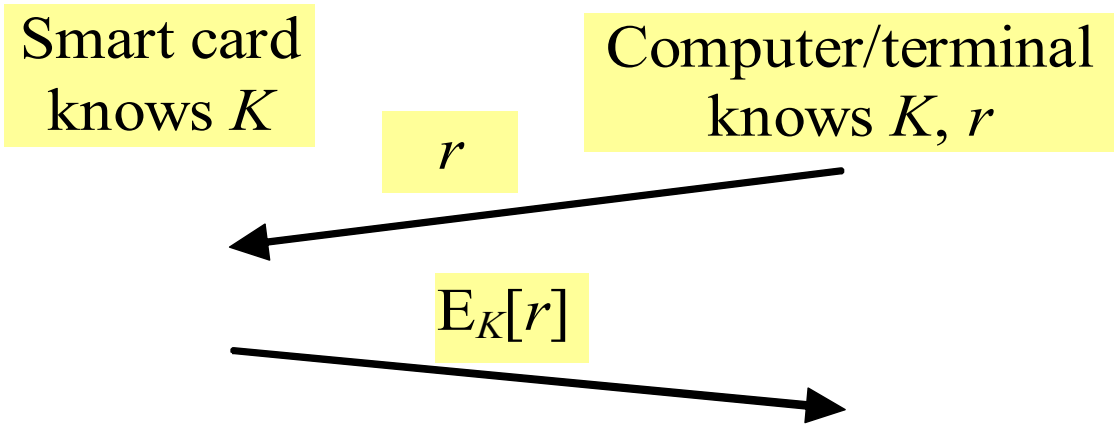
MS-CHAP: identical to CHAP except for using password to replace the key K.

C/R AuthN Protocols – More protocol variants

- *The challenge is an encrypted item and the user is challenged to perform* the decryption, demonstrating the knowledge of the secret key.
- K can be a symmetric key, a key derived from the user's password, or Bob's public key.



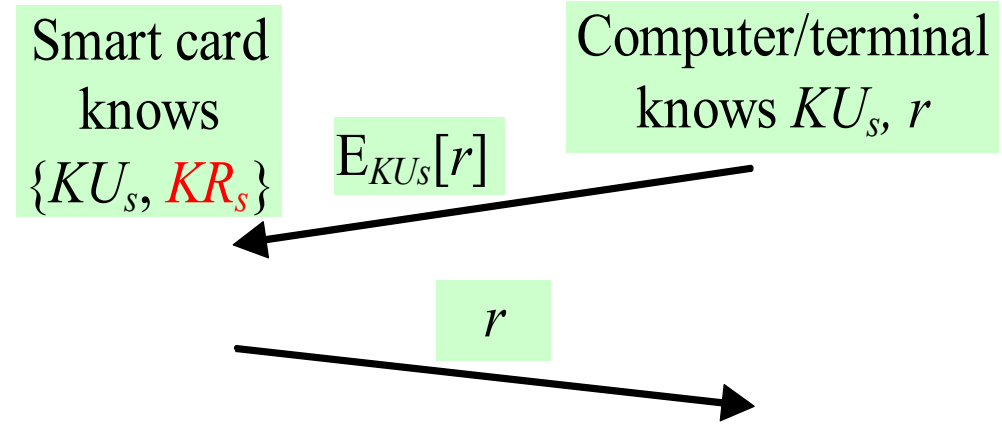
C/R AuthN Protocols – More protocol variants



Shared key challenge/response scheme

To save space, I have omitted the identification step in these protocols – you should not omit this step when doing the design.

If smart card returns a correct value (or *r)*, it means: (a) the card was at present, (b) correct PIN was entered (assuming PIN is used).



Public key challenge/response scheme

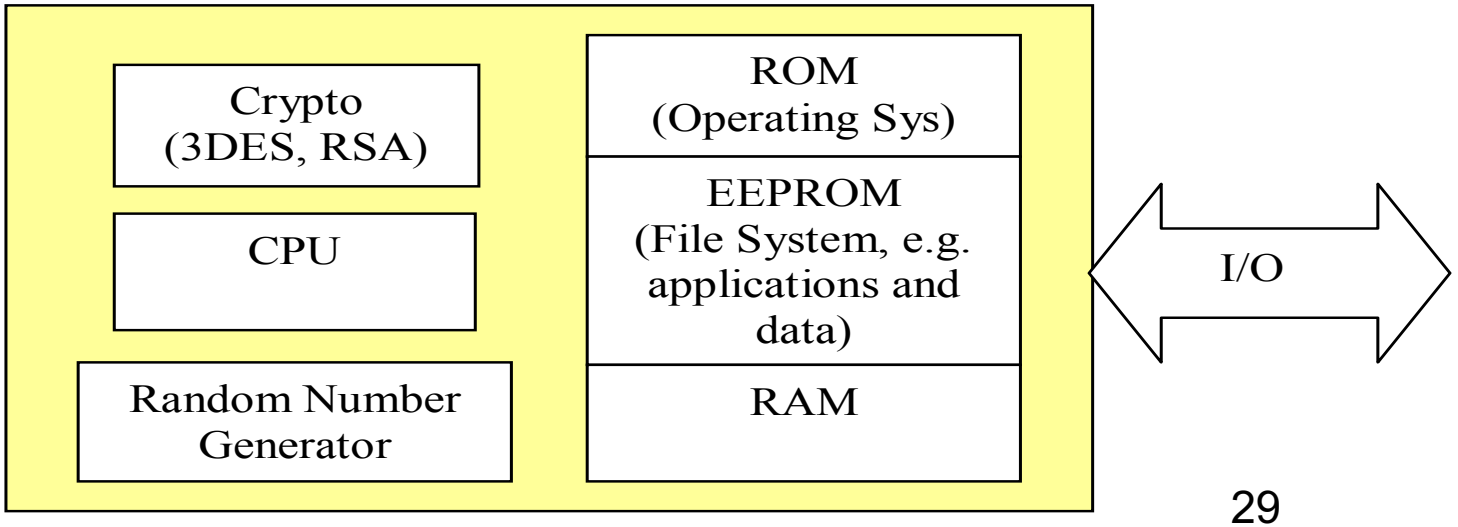
Token-based Authentication

- Various forms
  - PIN protected memory card
    - Enter PIN to get the password
  - PIN and smart phone based token
    - Google authentication
  - Smart cards with embedded CPU and memory
    - Carries conversation with a terminal (a computer or a card reader)
    - Card reader creates a random challenge
    - Enter PIN to encrypt/decrypt the challenge using the card

Smart Cards – Types

EPROM - Electrically Programmable R O M
EEPROM - Electrically Erasable Programmable R O M

- Memory cards
  - Contain EEPROM and ROM.
  - EEPROM holds data that changes with time.
  - ROM holds card number, card-holder name.
- Chip and PIN cards
  - On-board memory (RAM, ROM, and EEPROM).
  - On-board Crypto coprocessors for performing crypto operations, random number generator for key generations, and intelligence for making decisions.



Smart Cards – Types

- Provide robust security – more difficult to tamper with and uneconomical to counterfeit.
- Supports multiple applications (e.g. ePayment, ePassport, eIDs, Health, Transport, etc).

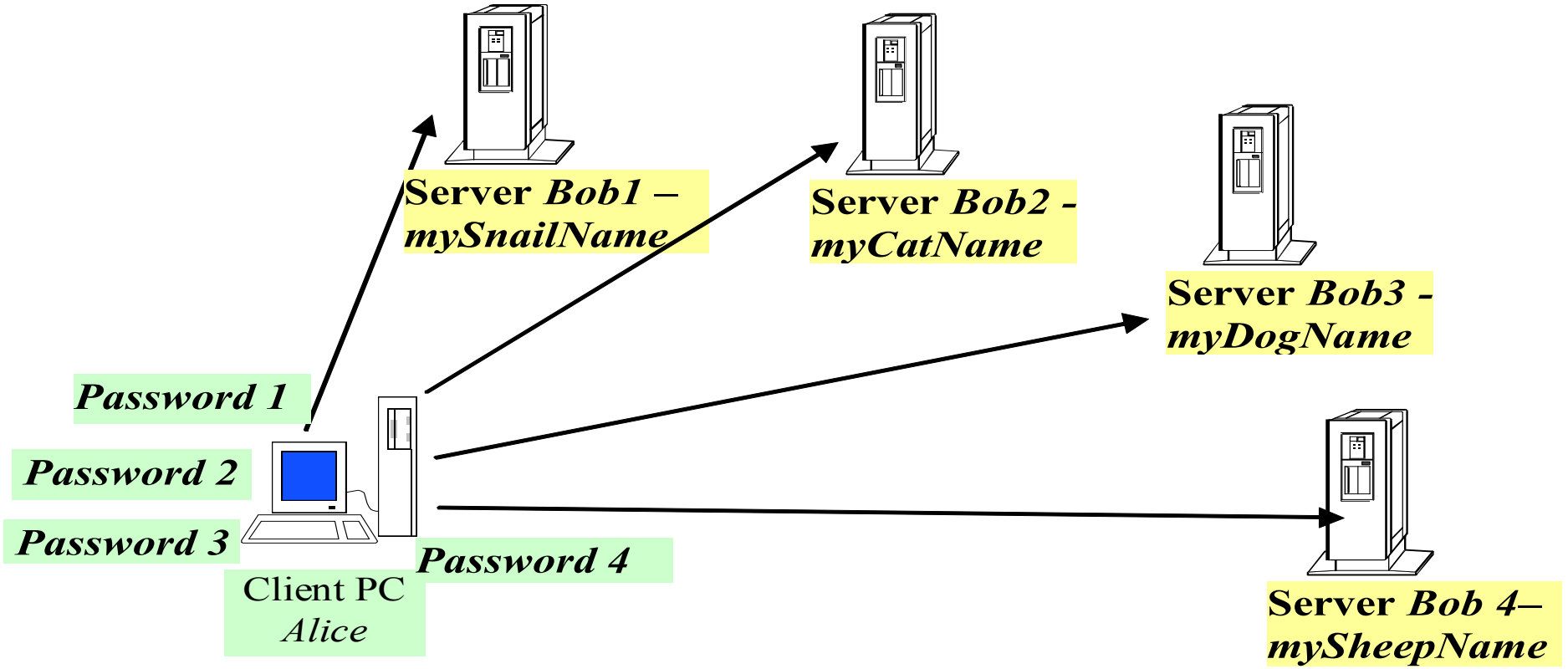
- Interface
  - Contact
  - Contactless (using RFID).
  - Hybrid
- Operating Systems (OS)
  - Java Cards
  - Multos
  - SLCOS
  - etc

Part 4 Overview

- Enterprise-wide authentication
- Conclusions

Enterprise Authentication

- The task of authentication is imposed on all the servers.
- For users, either the same password for accessing all the servers, or different passwords for different servers.

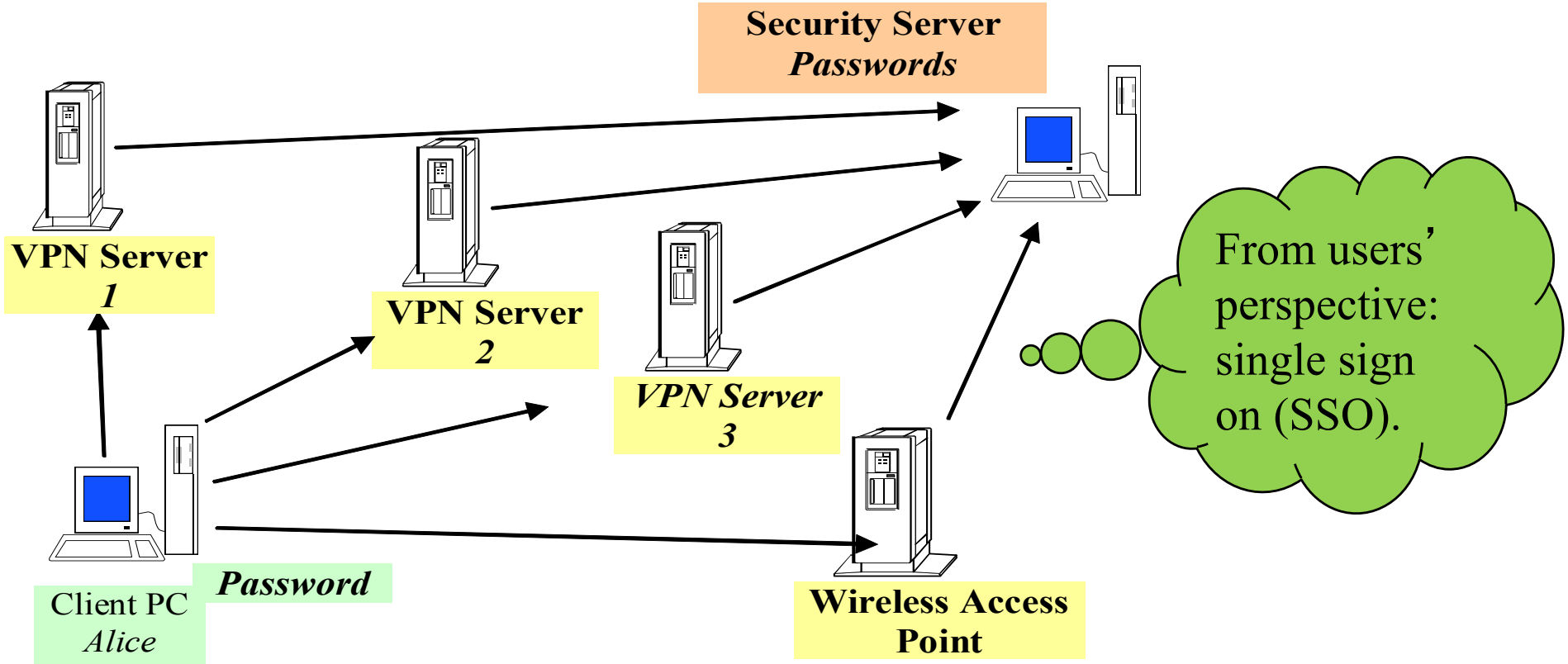


Enterprise Authentication

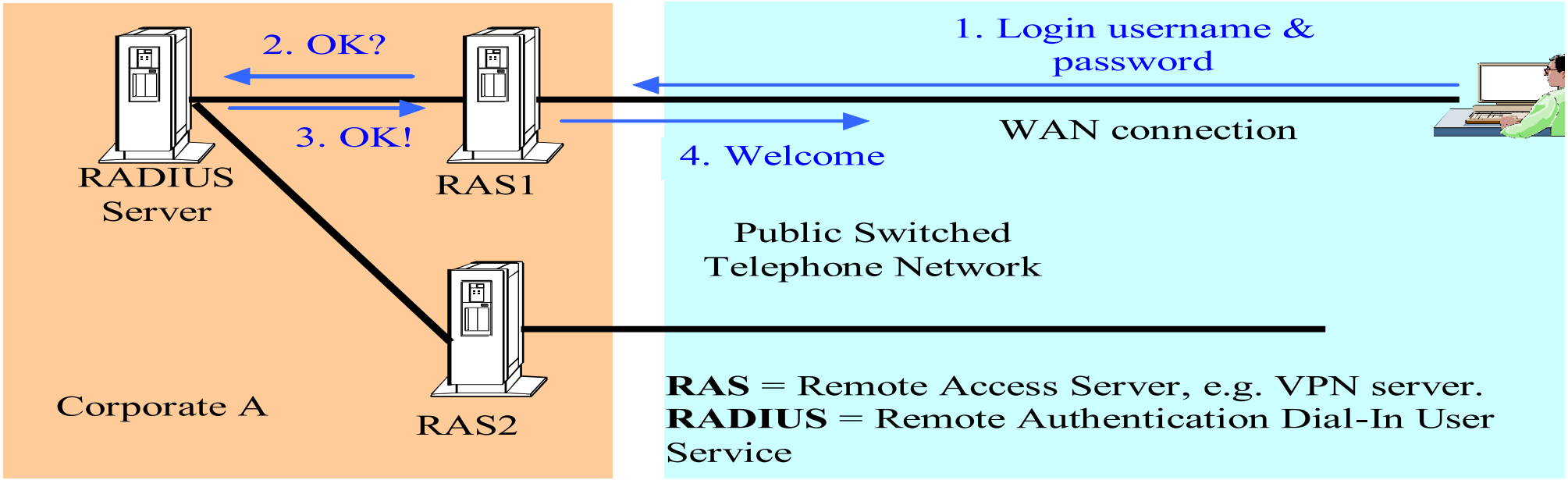
- Centralised authentication
  - **One central authority** (**security server or authentication server**) in **each domain/organisation** performs the task of authentication
    - One password per user;
    - Service servers do not need to handle authentication;
    - More secure as authentication servers only interact with service servers.
    - Unified enforcement of domain/organisation-wide security policies, e.g. AAA services.
- A number of systems exist, e.g.
  - Radius - Remote authentication for dial-in user service
    - Initially designed for remote dial-in, but now extended to handle enterprise-wide AAA services…..
  - Kerberos

Enterprise authentication

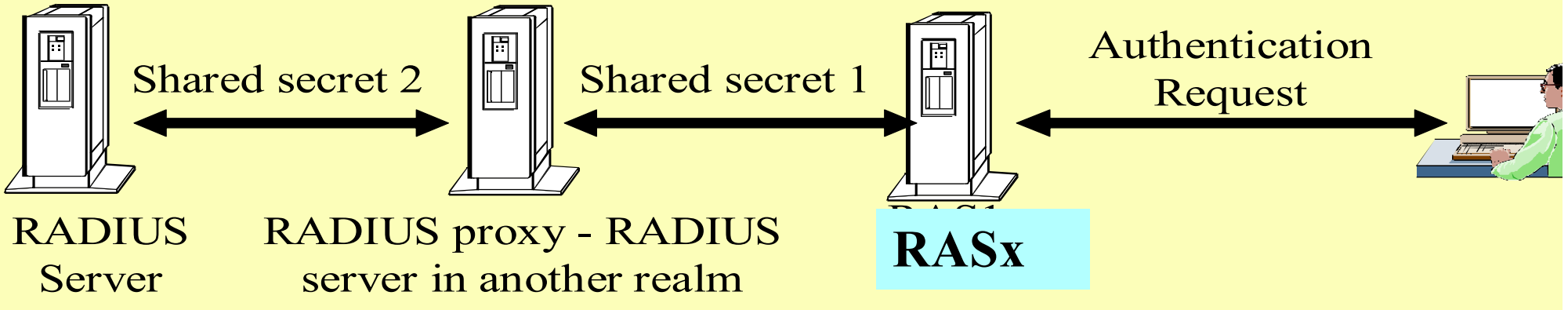
- The security server is responsible for providing AAA services to edge devices, e.g. VPN servers and wireless access points.



Enterprise authentication – RADIUS

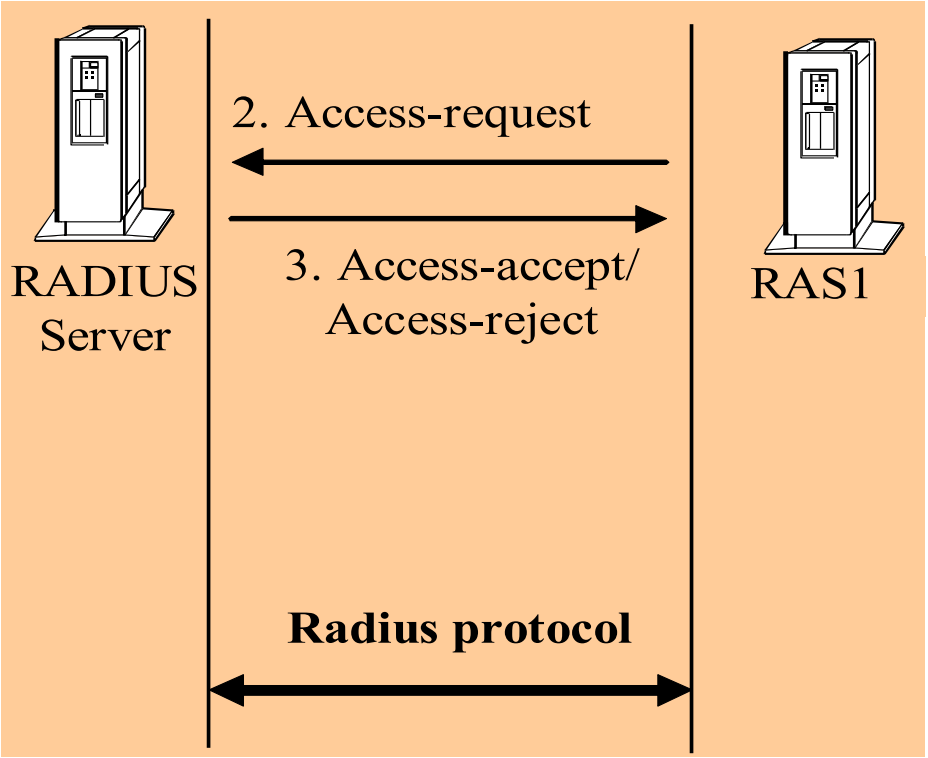


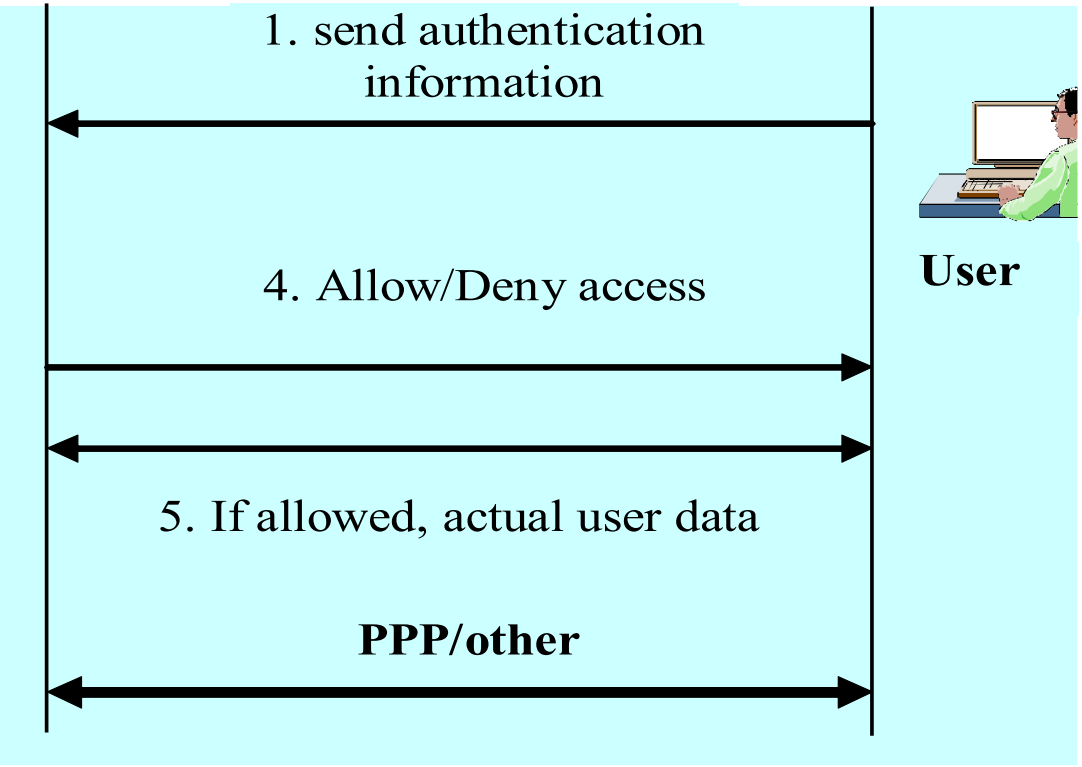
The model of trust: RADIUS uses shared secrets to achieve mutual authentication and confidential communication between RADIUS entities.



Enterprise authentication - RADIUS

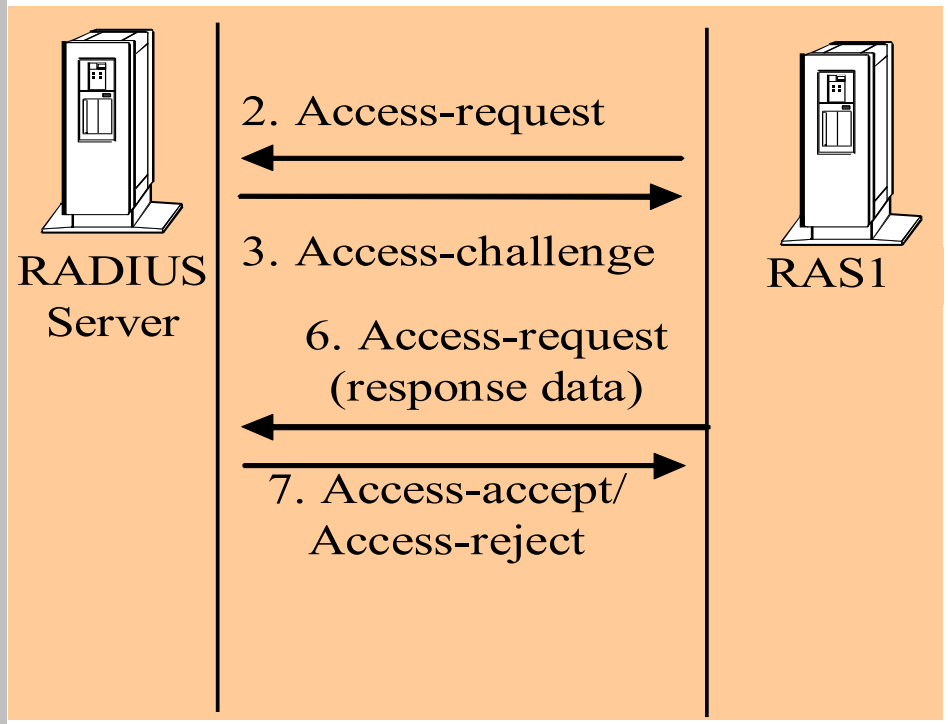
- Simple user authentication and authorisation using RADIUS
- PPP=Point-to-Point Protocol

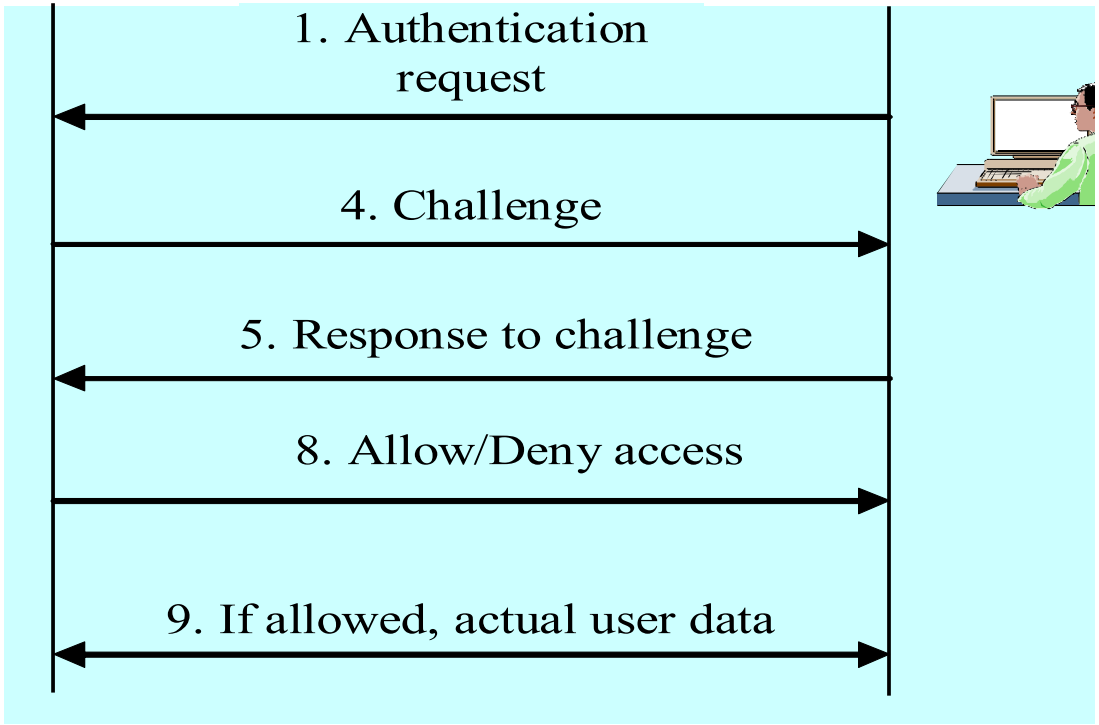




Enterprise authentication - RADIUS

- Challenge-response authentication and authorisation using RADIUS



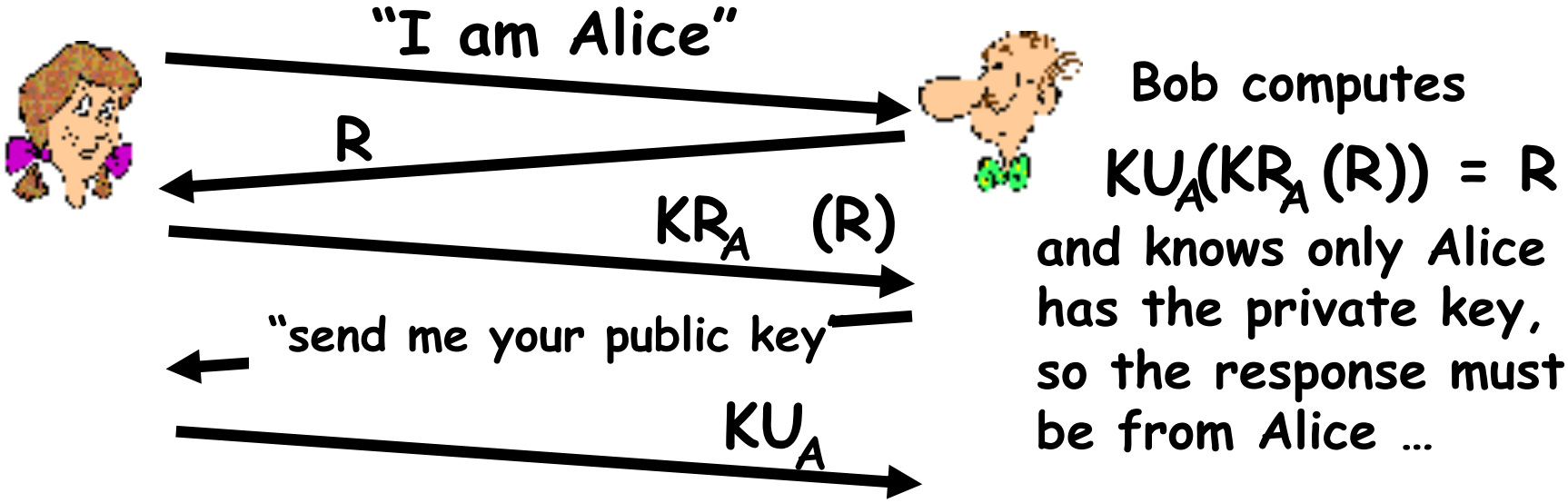


Enterprise authentication - RADIUS

- RADIUS protocol:
  - Client forwards the user access request to a RADIUS server
  - Server
    - Replies with *reject access* or *allow access* based upon a user supplied password/credential.
    - Challenge (when challenge-response protocol is used, e.g. CHAP).
    - If challenge-response is used, client forwards Challenge to the user, and the user sends their Response to the client that then forwards it to the server.
    - One RADIUS server may act as a client to another RADIUS server for consultation, etc.

Exercise Question – E9.1

Many authentication protocols are based on the use of a symmetric key/secret. What if that 'trust relationship' has not been established? Using nonce (random number) and a public-key cipher can be a solution. Is the following authentication protocol secure? Justify your answer.



Exercise Question – E9.2

- Design two different X.509 certificate based authentication protocols to support mutual authentication between two entities, Alice and Bob.

Conclusions (1/3)

- Passwords are the most basic authentication mechanism
  - The security level is only as good as the passwords you select.
  - They are vulnerable to guessing unless precautions ensure there is a large enough set of possible passwords and each potential one in the set is equally likely to be chosen.
  - Challenge-response techniques allow the system to vary the password therefore less vulnerable to guessing attacks; OTPs, an example of this technique, are particular effective against guessing and replaying attacks.
- Authentication can also be achieved with public-key cryptography for which public key certificates are needed - X.509 standard.

Conclusions (2/3)

- There are also other forms of authentication: biometrics measures physical characteristics of the user; location requires the verifier to determine the location of the user.
- In practice, some combination of these methods can be used. This depends on the resources available to the verifier and the user, the strength of the authentication required, and external factors such as lows and customs.
- System designers have to balance convenience and security. Ease-of-use is an important factor in IT systems. However, convenient practices may introduce new vulnerabilities.
- There are authentication issues when multiple systems or multiple sites are involved.

Conclusions (3/3) – Applications of PKC

- Confidentiality – PKC allows encryption, and therefore protection against unauthorised disclosure of info.
- Signatures and integrity – PKC can be used to generate a signed token for a message so that the recipient of the message can determine if the message is authentic, or has been modified.
- Signatures and non-repudiation – PKC signatures can be used to determine if a message has indeed been sent by a particular sender.
- Identification and authentication – PKC can be used to determine an identity of a user and a service.
- Timestamping - PKC can be used to generate a time stamp to certify that a particular message has indeed been sent at a particular time and date.