CAN304
Computer Systems Security

Lecture 6. User Authentication

Week 6: 2022-04-01, 14:00-16:00, Friday

Jie Zhang
Department of Communications and Networking
Email: jie.zhang01@xjtlu.edu.cn
Office: EE522

Review of Last Week

- Designing security protocols
- Key establishment protocols
- Security protocols for smart lock

Outline

- Introduction to user authentication
- Basic authentication mechanisms

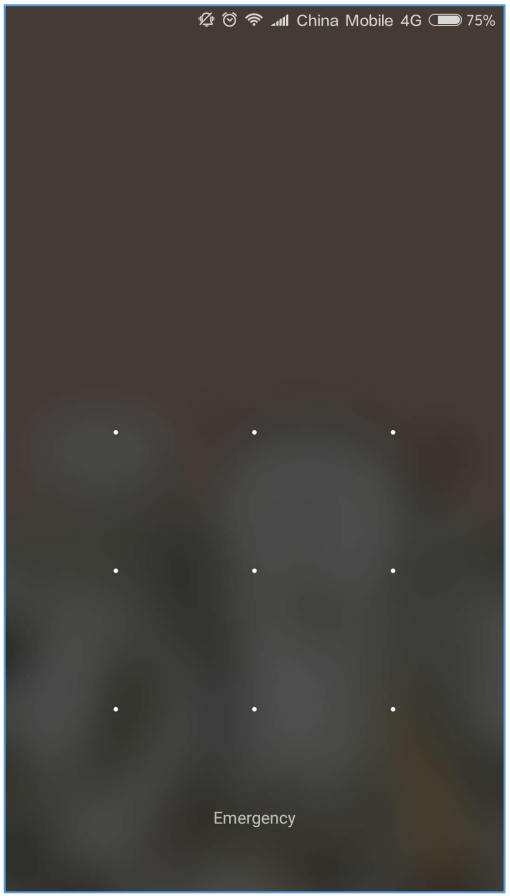
Learning objectives

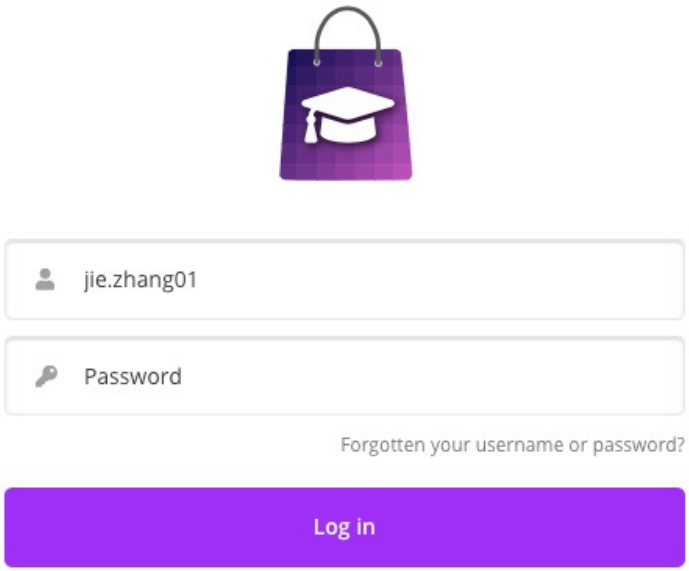
- Discuss the four general means of user authentication.
- Explain the mechanism by which hashed passwords are used for user authentication.

1. Introduction

Introduction

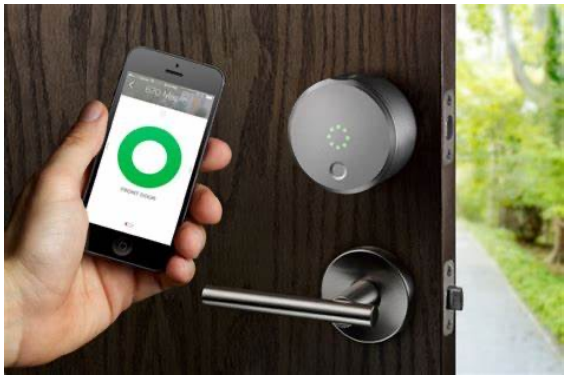
- What is user authentication?

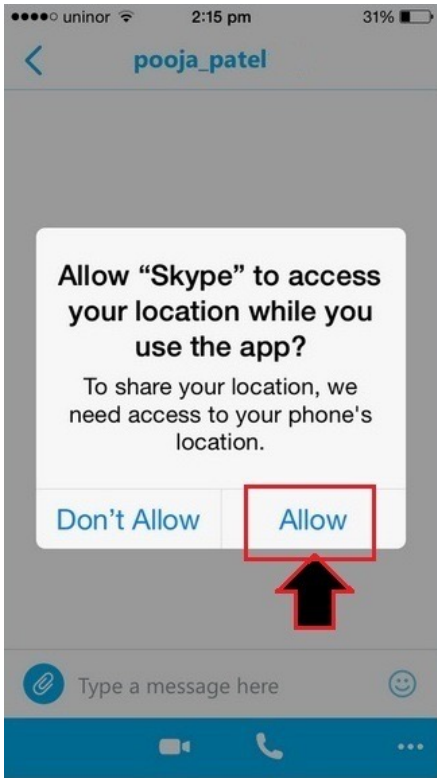




Authentication

- Determining the identity of some entity
  - Process
  - Machine
  - Human user





- Requires notion of identity
- And some degree of proof of identity

Proving identity in the physical world

- Most frequently done by physical recognition
  - I recognize your face, your voice, your body
- What about identifying those we don’t already know?

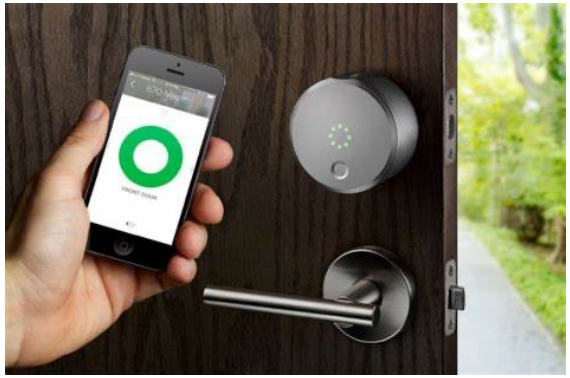
Other physical identification methods

- Identification by recommendation
  - You introduce me to someone
- Identification by credentials
  - You show me your student ID card
- Identification by knowledge
  - You tell me something only you know

- These all have cyber analogs

Differences in cyber identification

- Usually the identifying entity isn’t human
- Often the identified entity isn’t human, either
- Often no physical presence required
- Often no later rechecks of identity



Identifying with a computer

- Not as smart as a human
  - Steps to prove identity must be well defined
- Can’t do certain things as well
  - E.g., face recognition
- But lightning fast on computations and less prone to simple errors
  - Mathematical methods are acceptable

Identifying computers and programs

- No physical characteristics
  - Faces, fingerprints, voices, etc.
- Generally easy to duplicate programs
- Not smart enough to be flexible
  - Must use methods they will understand
- Again, good at computations

Physical presence optional

- Often authentication required over a network or cable
- Even if the party to be identified is human
- So authentication mechanism must work in face of network characteristics
  - Active wiretapping
  - Everything is converted to digital signal

Identity might not be rechecked

- Human beings can make identification mistakes
- But they often recover from them
  - Often quite easily
- Based on observing behavior that suggests identification was wrong
- Computers and programs rarely have that capability
  - If they identify something, they believe it

2. Basic authentication mechanisms

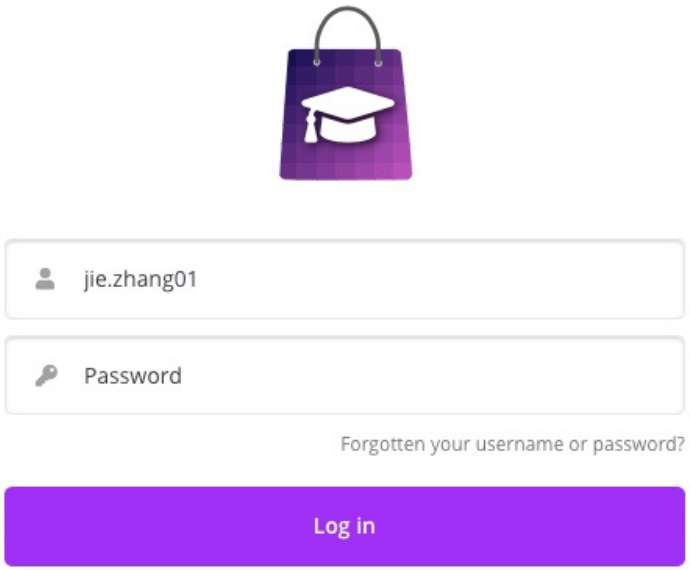
Authentication mechanisms

- Something you know
  - Passwords, challenge/response
- Something you have (token)
  - Smart cards, electronic keycard, physical key
- Something you are (static biometrics)
  - Fingerprint, retina, face
- Something you do (dynamic biometrics)
  - Voice pattern, handwriting, typing rhythm

2.1 Authentication mechanisms based on something you know

Passwords

- One of the oldest and most commonly used security mechanisms
- Authenticate the user by requiring him to produce a secret
  - Usually known only to him and to the authenticator



Problems with passwords

- They have to be unguessable
  - Yet easy for people to remember
- If networks connect remote devices to computers, susceptible to password sniffers
- Unless quite long, brute force attacks often work on them

Proper use of passwords

- Passwords should be sufficiently long
- Passwords should contain non-alphabetic characters
- Passwords should be unguessable
- Passwords should be changed often
- Passwords should never be written down
- Passwords should never be shared
- Hard to achieve all these simultaneously

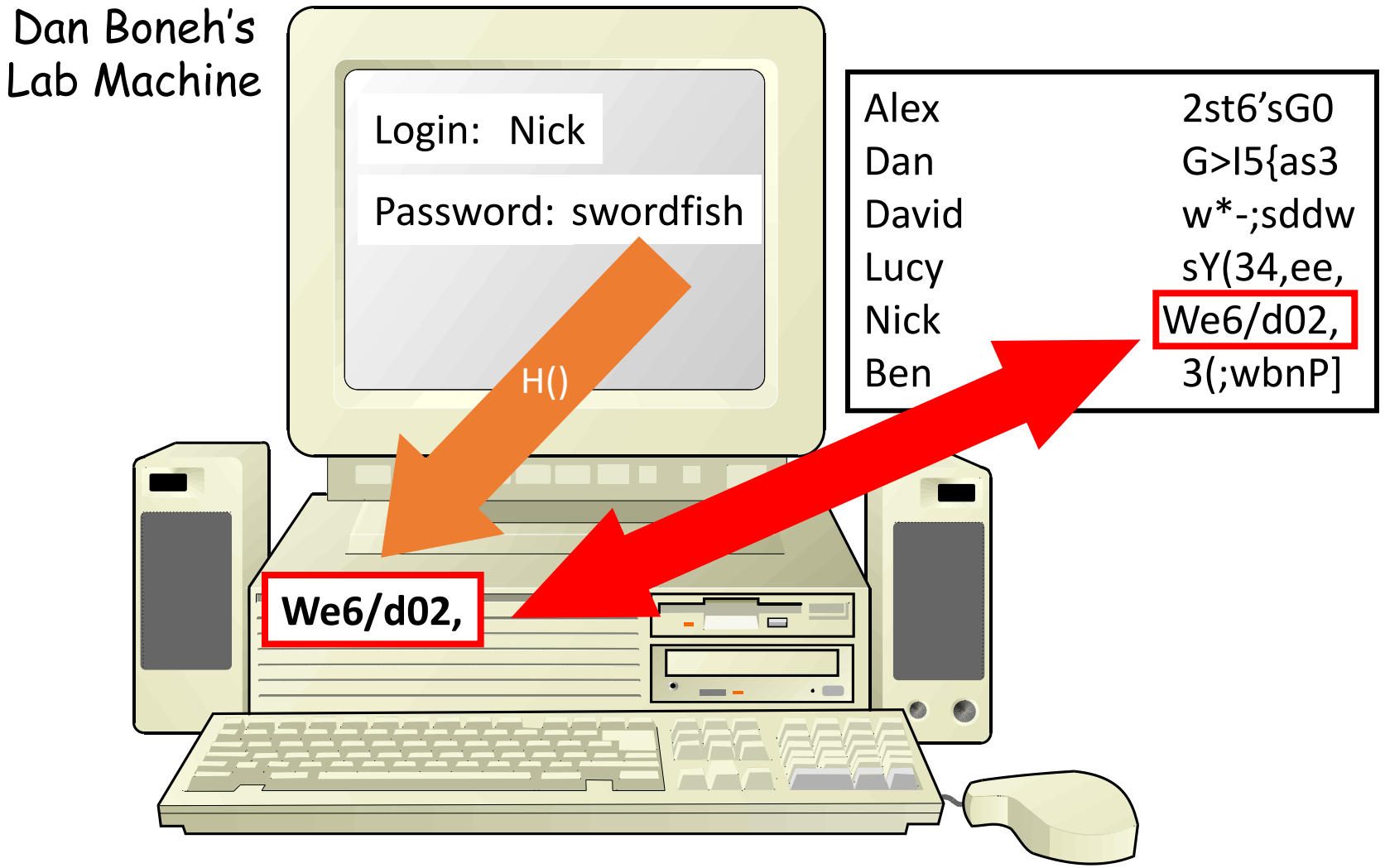
Passwords and single sign-on

- Many systems ask for password once
  - Resulting authentication lasts for an entire “session”
- Trading security for convenience
- Especially if others can use the authenticated machine

Handling passwords

- The OS must be able to check passwords when users log in
- So must the OS store passwords?
- Not really
  - It can store a hashed version
- Hash the offered password
  - e.g., MD5
- And compare it to the stored version

Standard password handling



Is hashing the password file enough?

- What if an attacker gets a copy of your password file?
- Dictionary attacks

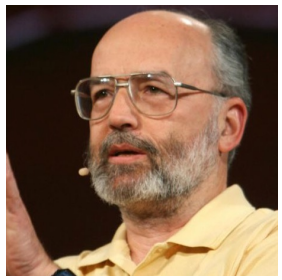
Alex	2st6'sG0
Dan	G>I5{as3
David	w*-;sddw
Lucy	sY(34,ee,
Nick	We6/d02,
Ben	3(;wbnP]

aardvark	340jafg;
aardwolf	K]ds+3a,
abaca	sY(34,ee

Dictionaries

- Dictionary based on probability of words being used as passwords
- Partly set up as procedures
  - E.g., try user name backwards
- Checks common names, proper nouns, etc.

Illustrating the problem

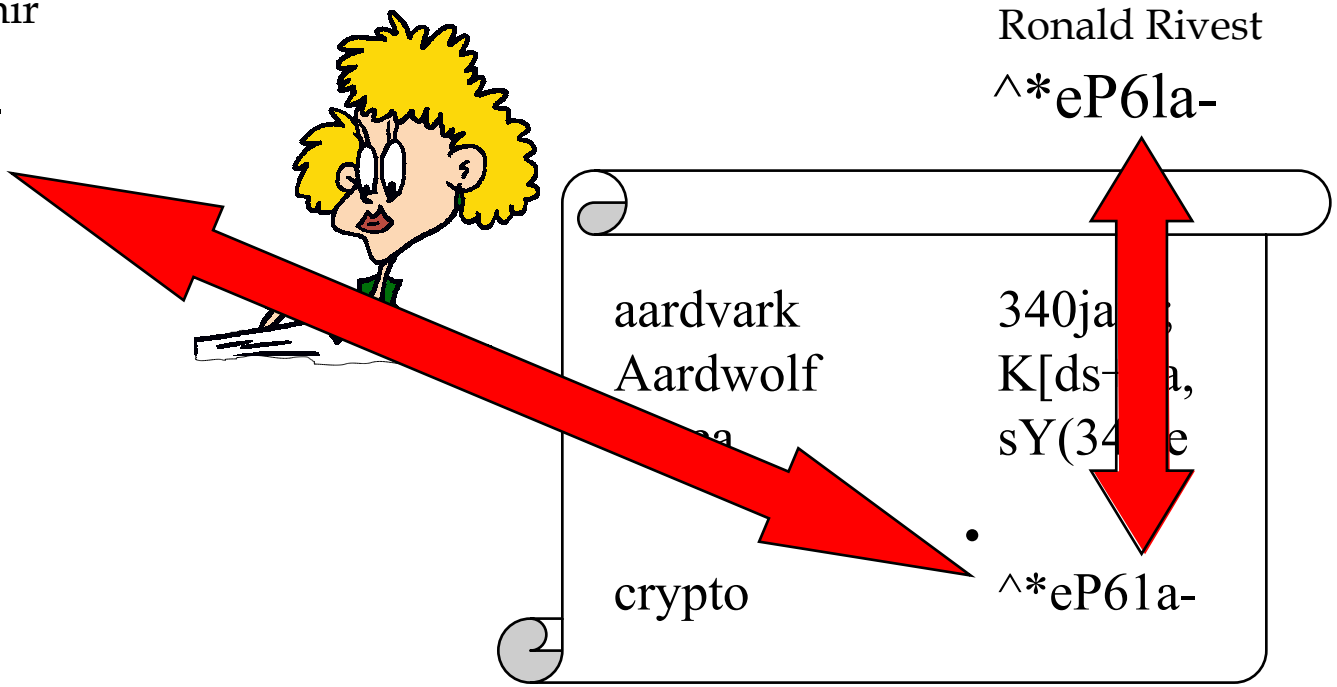


Adi Shamir

^*eP6la-

Ronald Rivest

^*eP6la-



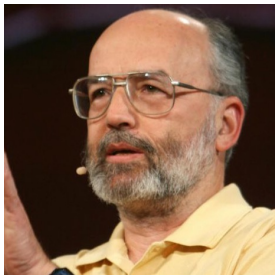
The real problem

- Not just that Shamir and Rivest chose the same password
- But that anyone who chose that password got the same hashed result
- So the attacker need only hash every possible password once
- And then she has a complete dictionary usable against anyone

Salted passwords

- Combine the plaintext password with a random number
  - Then run it through the one-way function
- The random number need not be secret
- It just has to be different for different users

Did it fix our problem?

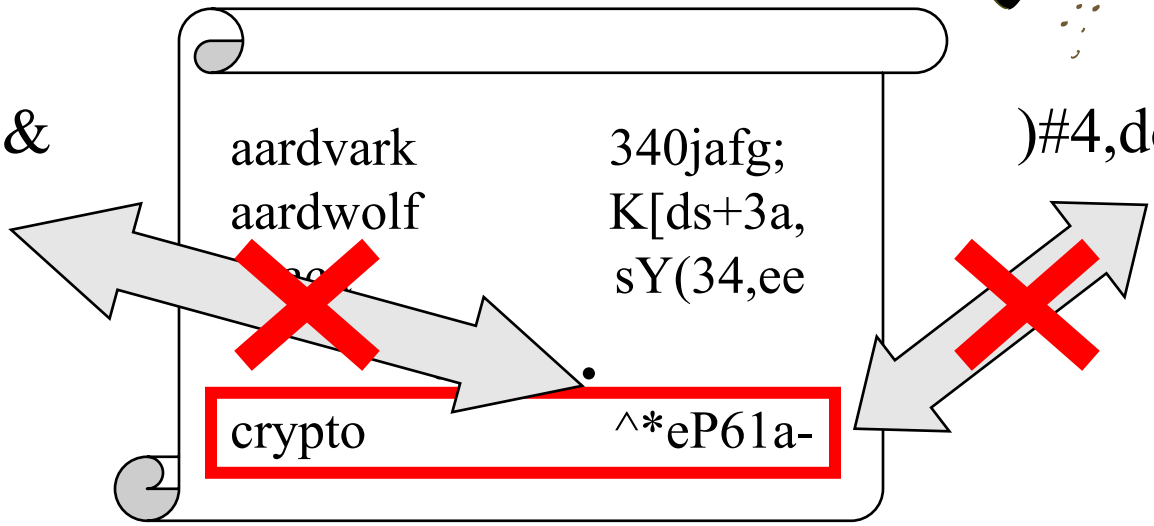


Adi Shamir

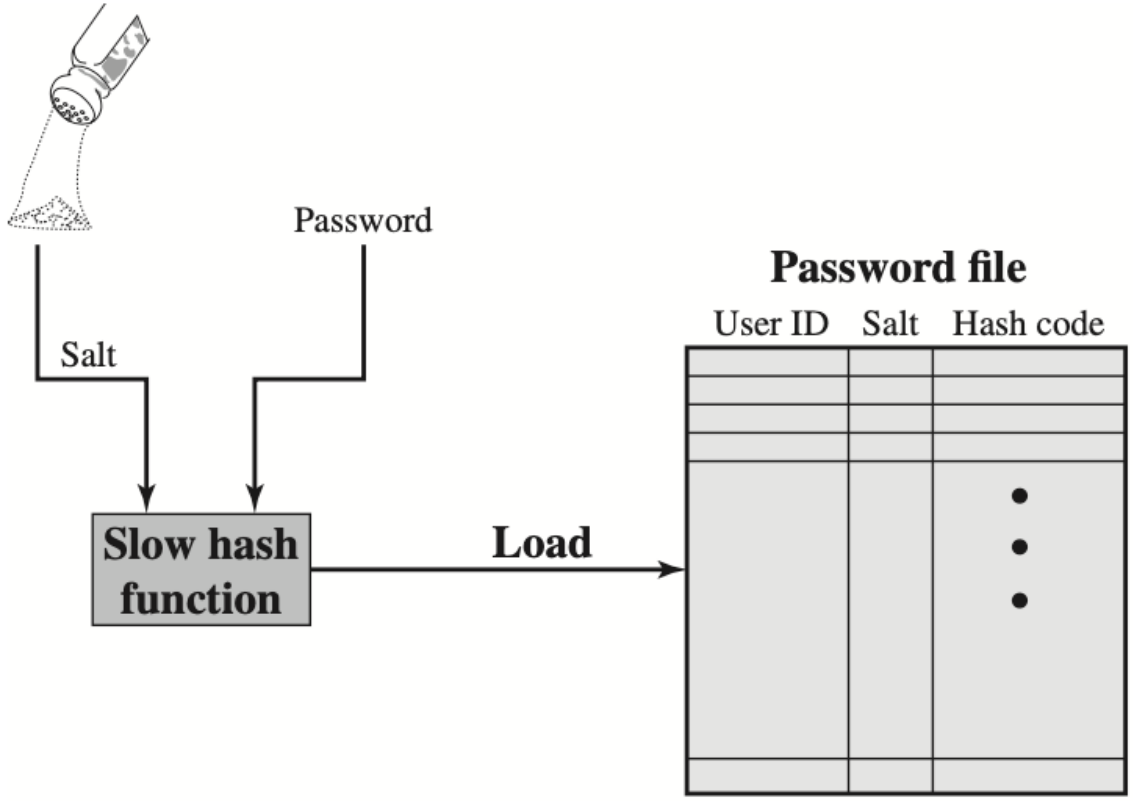
Ronald Rivest

D0Cls6&

)#4,doa8

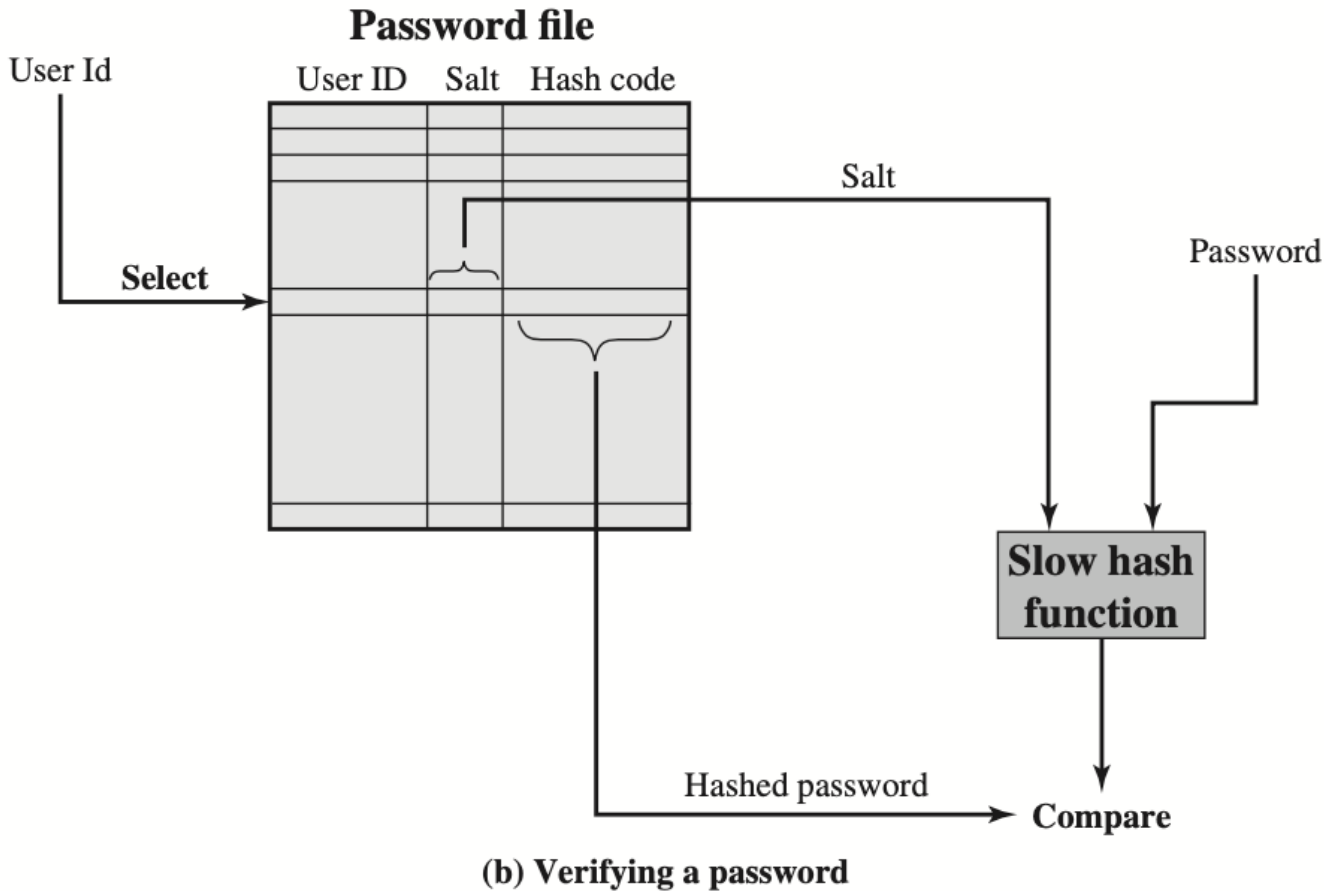


UNIX Password Scheme



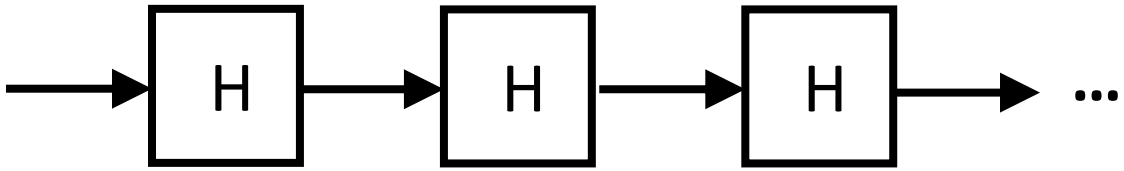
(a) Loading a new password

UNIX Password Scheme



(b) Verifying a password

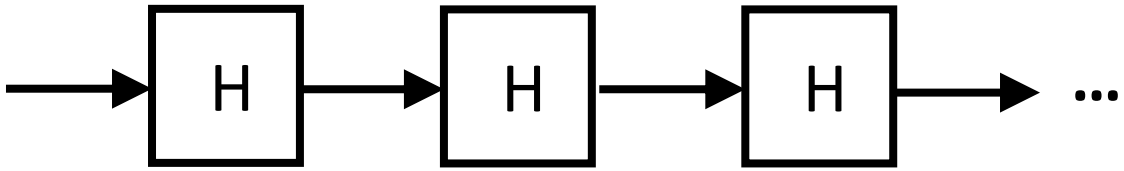
S/Key: One-Time Password



Server (verifier) generates

$x, H(x), H^2(x), \ldots, H^{n+1}(x)$

S/Key: One-Time Password



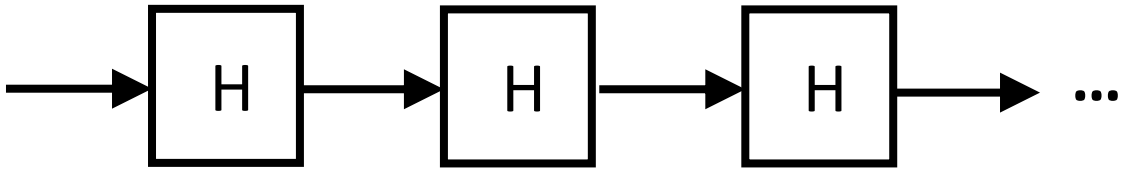
Server (verifier) generates

$x, H(x), H^2(x), \ldots, H^{n+1}(x)$

User (prover) is given n one-time passwords:

$H(x), H^2(x), \ldots, H^n(x)$

S/Key: One-Time Password



Server (verifier) generates

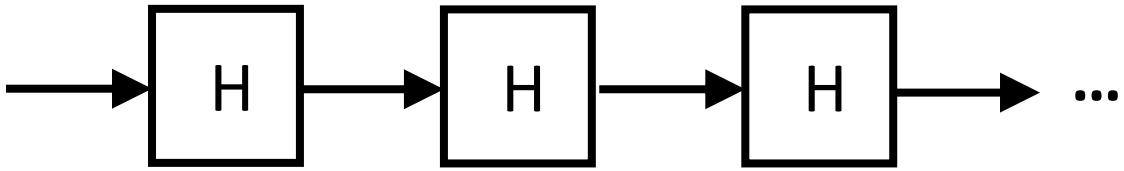
~~x~~, ~~H(x)~~, ~~H^2(x)~~, ..., $H^{n+1}(x)$

Server only stores $H^{n+1}(x)$ and discards others

User (prover) is given n one-time passwords:

H(x), $H^2(x)$, …, $H^n(x)$

S/Key: One-Time Password



Server (verifier) generates

~~x~~, ~~H~~(x), ~~H²~~(x), ~~.~~, $H^{n+1}(x)$

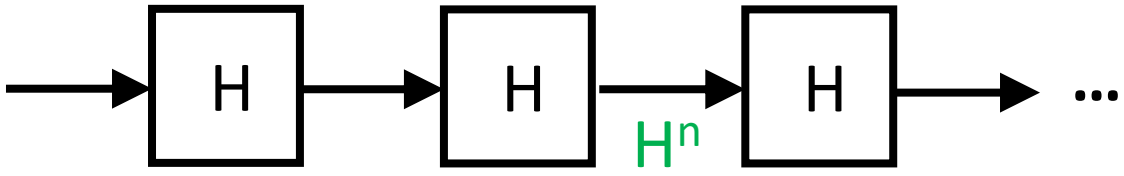
Server only stores $H^{n+1}(x)$ and discards others

User (prover) is given n one-time passwords:

$H(x), H^2(x), \ldots, H^n(x)$

User uses in the reverse order

S/Key: One-Time Password



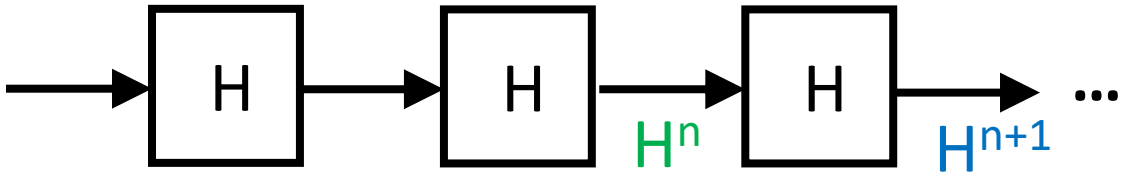
Server (verifier) stores $H^{n+1}(x)$

User (prover) is given n one-time passwords:

$H(x), H^2(x), \ldots, H^n(x)$

User uses in the reverse order

S/Key: One-Time Password



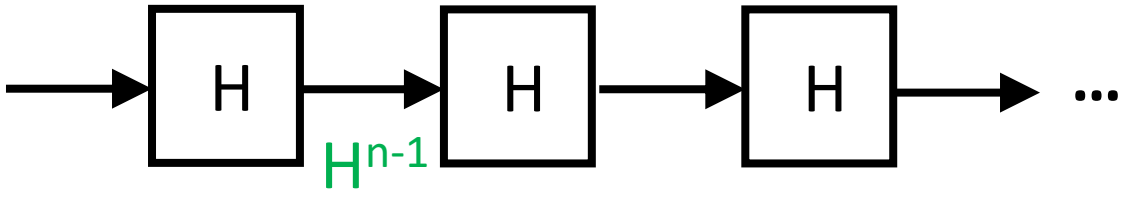
Server (verifier) stores $H^{n+1}(x)$

User (prover) is given n one-time passwords:

$H(x), H^2(x), \ldots, H^n(x)$

User uses in the reverse order

S/Key: One-Time Password



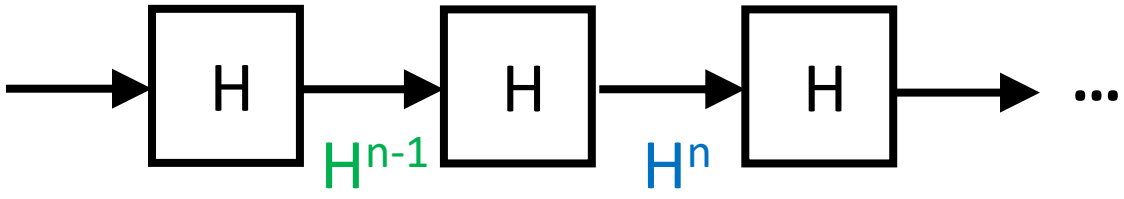
Server (verifier) **updates** to $H^n(x)$

User (prover) is given n one-time passwords:

$H(x), H^2(x), \ldots, H^n(x)$

User uses in the reverse order

S/Key: One-Time Password



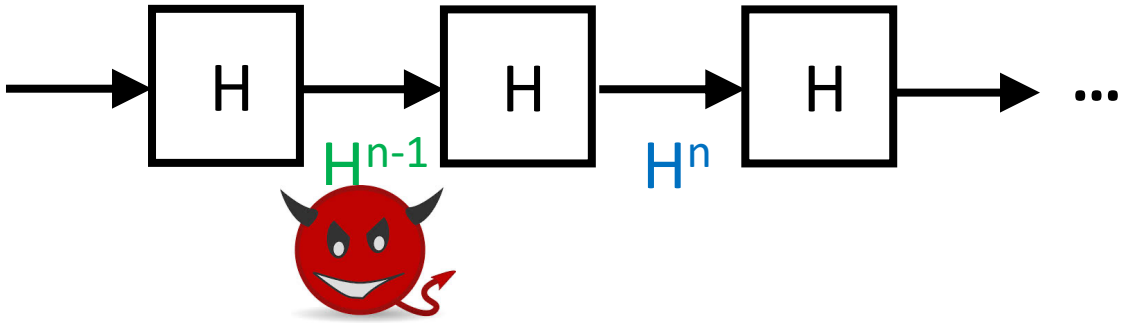
Server (verifier) **updates** to $H^n(x)$

User (prover) is given n one-time passwords:

$H(x), H^2(x), \ldots, H^n(x)$

User uses in the reverse order

S/Key: One-Time Password



Compromise yields outdated passwords

Limits the number passwords to n

Question

- Other than password, list three methods for user authentication and real-life examples for each of them.

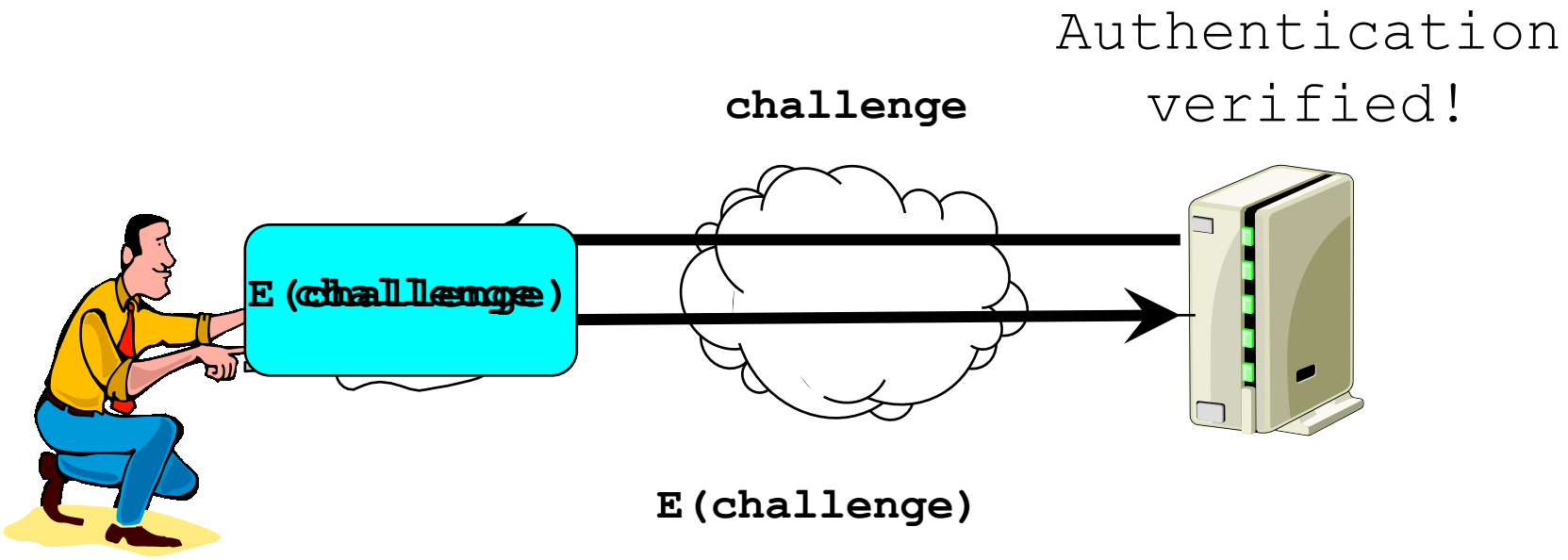
2.2 Authentication mechanisms based on something you have (token)

Identification devices

- Authentication by what you have
- A smart card or other hardware device that is readable by the computer
- Authenticate by providing the device to the computer
  - e.g. USB-KEY used by digital bank

Authentication with smart cards

- How can the server be sure of the remote user's identity?



- Often user must enter password to activate card

Problems with identification devices

- If lost or stolen, you can’t authenticate yourself
  - And maybe someone else can
  - Often combined with passwords to avoid this problem
- Unless cleverly done, susceptible to sniffing attacks
- Requires special hardware

2.3 Biometric authentication

Biometric authentication

- Attempts to authenticate an individual based on unique physical characteristics
- Based on pattern recognition
- Is technically complex and expensive when compared to passwords and tokens

Biometric characteristics

- Physical characteristics used include:
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
  - Iris
  - Signature
  - Voice

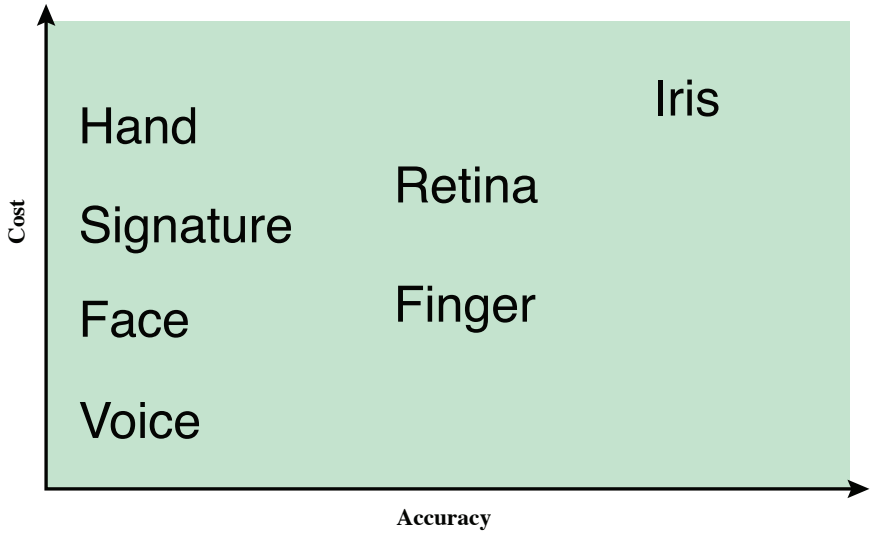


Figure 3.7 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes.

Biometric system

- Enrollment
- Verification
- Identification

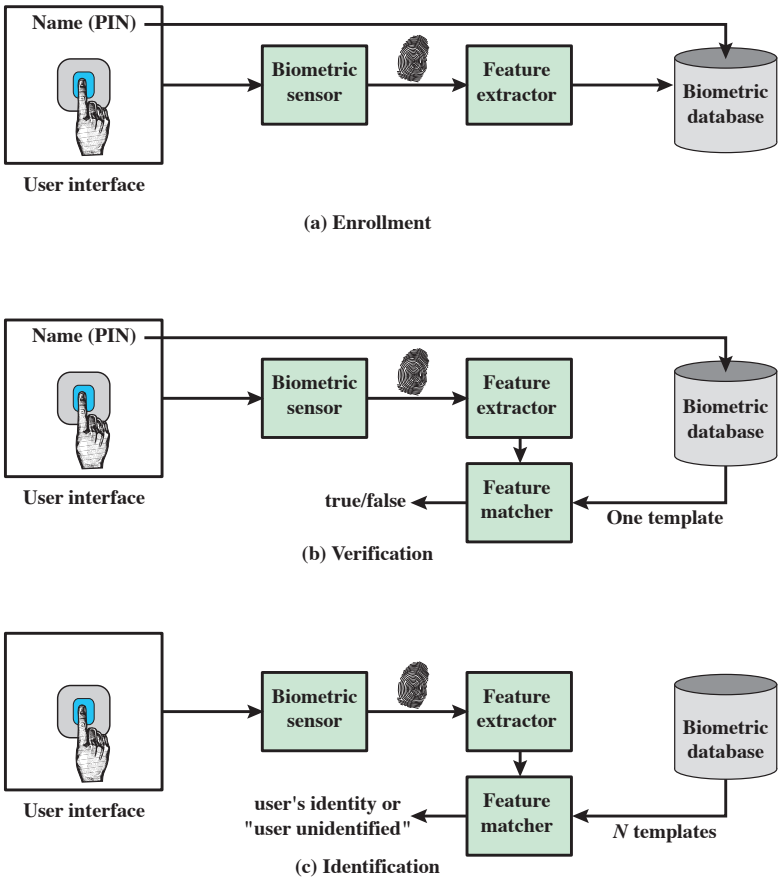


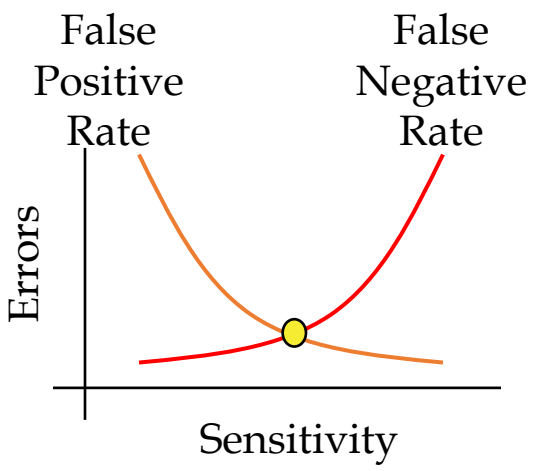
Figure 3.8 A Generic Biometric System. Enrollment creates an association between a user and the user's biometric characteristics. Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

Problems with biometric authentication

- Usually requires very special hardware
- May not be as foolproof as you think
- Many physical characteristics vary too much for practical use
- Generally not helpful for authenticating programs or roles
- What happens when it's cracked?
  - You only have two retinas, after all

Usability vs. security

- False positives
  - Match made when it shouldn't have been
- False negatives
  - Match not made when it should have been



The Crossover Error Rate (CER)

Generally, the lower the CER is, the better the system

But sometimes one rate more important than the other

Biometrics and usability

- Always a tradeoff in false positives vs. false negatives
- For consumer devices, false negatives are very, very bad
  - People discard devices that won't let the legitimate user in

Summary

- Introduction
- Basic authentication mechanisms
  - Something you know
  - Something you have
  - Biometric authentication