Authentication

Introduction

- Basics
- Passwords
- Challenge-Response
- Biometrics
- Location
- Multiple Methods

Basics

- Authentication: binding of identity to subject
  - Identity is that of external entity (e.g., Bob)
  - Subject is computer entity (e.g.,process)

Establishing Identity

- One or more of the following
  - What entity knows (password)
  - What entity has (badge, smart card)
  - Who entity is (fingerprints, retinal characteristics)
- Other approaches
  - Where entity is (In front of a particular terminal)

Password System

Password System

- Password system, with passwords stored
  - A set of strings making up passwords
  - *Verify input = stored password*
  - Single equality test function
  - Function to set/change password

Storage

- Store as cleartext
  - If password file compromised, *all* passwords revealed
- Encipher file
  - Need to have decipherment, encipherment keys in memory
  - Reduces to previous problem
- Store one-way hash of password
  - If file read, attacker must still guess passwords or invert the hash

Examples

- The UNIX method
  - */etc/passwd*
  - *crypt ( )*
  - Use DES to encipher 0 message with password as key
  - Iterate 25 times
  - The final 64 bits are unpacked into a string of 11 printable characters
  - *Recent versions use bigcrypt( ), crypt16( ),*Blowfish and MD5

Attacks and Countermeasures

Dictionary Attacks

- Trial-and-error from a list of potential passwords
  - *Off-line*: know the password function, and repeatedly try different guesses until the list is done or passwords guessed
    - Examples: *crack*, *john-the-ripper*
  - *On-line*: have access to functions and try guesses *g* until some *l*(*g*) succeeds
    - Examples: trying to log in by guessing a password

Using Time

- *P* probability of guessing a password in specified period of time
- *G* number of guesses tested in 1 time unit
- *T* number of time units
- *N* number of possible passwords (| *A* |)
- Then $P \geq TG/N$

Salting

- Goal: slow dictionary attacks
- Method: perturb hash function so that:
  - Parameter controls *which* hash function is used
  - Parameter differs for each password
    - E.g., the DES salt is a 12-bit number, between 0 and 4,095
  - So given *n* password hashes, and therefore *n* salts, need to hash guess *n*

Password Aging

- Force users to change passwords after some time has expired
  - How do you force users not to re-use passwords?
    - Record previous passwords
    - Block changes for a period of time
  - Give users time to think of good passwords
    - Don't force them to change before they can log in
    - Warn them of expiration days in advance

Password Selection

Password Selection

- Random selection
  - Any password from *A* equally likely to be selected
- Pronounceable passwords
- User selection of passwords

Pronounceable Passwords

- Generate phonemes randomly
  - Phoneme is unit of sound, eg. *cv*, *vc*, *cvc*, *vcv*
  - Examples: helgoret, juttelon are; przbqxdfl, zxrptglfn are not
- Problem: too few
- Solution: key crunching
  - Run long key through hash function and convert to printable sequence
  - Use this sequence as password

User Selection

- Problem: people pick easy to guess passwords
  - Based on account names, user names, computer names, place names
  - Dictionary words (also reversed, odd capitalizations, control characters, “elite-speak”, conjugations or declensions, swear words, Torah/Bible/Koran/… words)
  - Too short, digits only, letters only
  - License plates, acronyms, social security numbers
  - Personal characteristics or foibles (pet names, nicknames, job characteristics, *etc*.

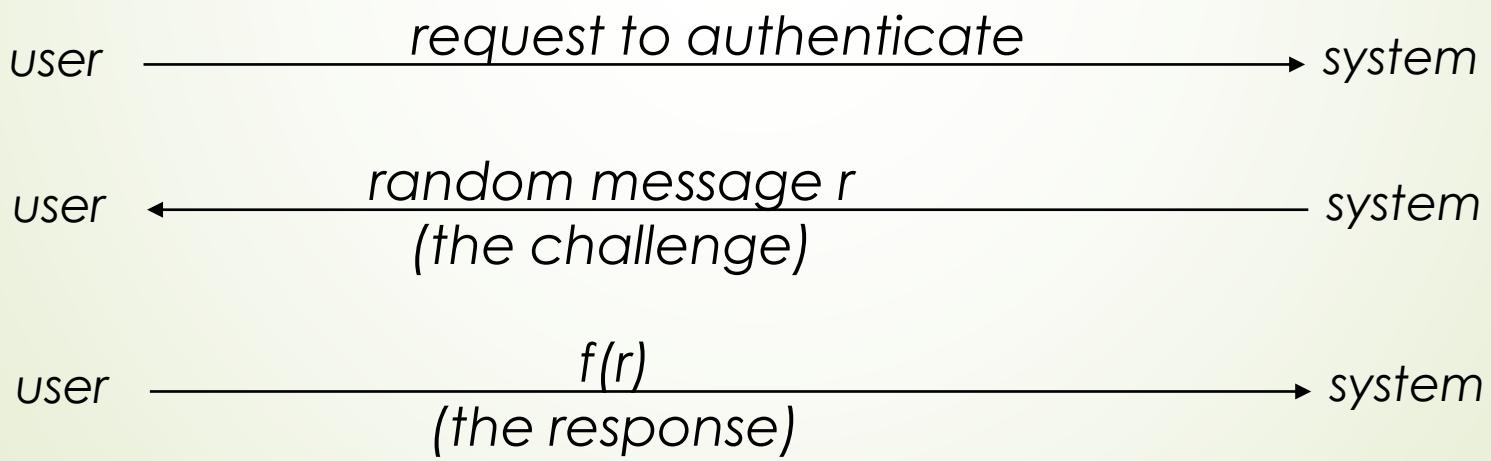
Proactive Password Checking

- Analyze proposed password for “goodness”
  - Always invoked
  - Can detect, reject bad passwords for an appropriate definition of “bad”
  - Discriminate on per-user, per-site basis
  - Needs to do pattern matching on words
  - Needs to execute subprograms and use results
  - Easy to set up and integrate into password selection system

Challenge-Response

Challenge-Response

- User, system share a secret function *f* (in practice, *f* is a known function with unknown parameters, such as a cryptographic key)



One-time Passwords

One-Time Passwords

- Password that can be used exactly *once*
  - After use, it is immediately invalidated
- Challenge-response mechanism
  - Challenge is number of authentications;
  - response is password for that particular number
- Problems
  - Synchronization of user, system
  - Generation of good random passwords
  - Password distribution problem

S/Key

- One-time password scheme
- *h* one-way hash function (MD5 or SHA-1, for example)
- User chooses initial seed *k*
- System calculates:

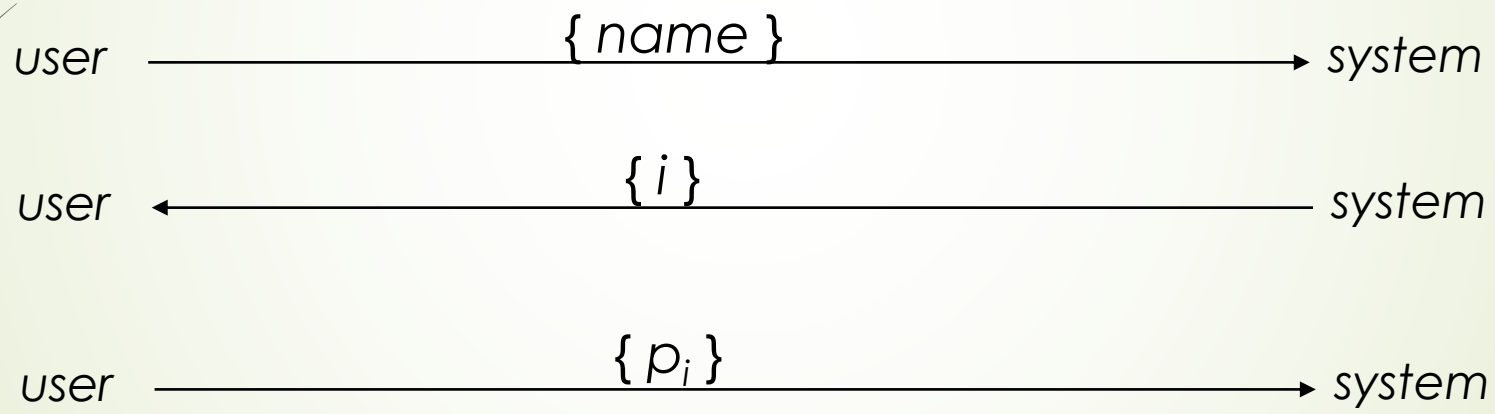
$$h(k) = k_1,\ h(k_1) = k_2,\ \ldots,\ h(k_{n-1}) = k_n$$

- Passwords are reverse order:

$$p_1 = k_n,\ p_2 = k_{n-1},\ \ldots,\ p_{n-1} = k_2,\ p_n = k_1$$

S/Key Protocol

System stores maximum number of authentications n, number of next authentication i, last correctly supplied password p_{i-1}.



If match with what is stored, system replaces p_{i-1} with p_i and increments i.

Biometrics

Biometrics

- Automated measurement of biological, behavioral features that identify a person
  - Fingerprints: optical or electrical techniques
    - Maps fingerprint into a graph, then compares with database
    - Measurements imprecise, so approximate matching algorithms used
  - Voices: speaker verification or recognition
    - Verification: uses statistical techniques to test hypothesis that speaker is who is claimed (speaker dependent)
    - Recognition: checks content of answers (speaker independent)

Other Characteristics

- Can use several other characteristics
  - Eyes: patterns in irises unique
    - Measure patterns, determine if differences are random; or correlate images using statistical tests
  - Faces: image, or specific characteristics like distance from nose to chin
    - Lighting, view of face, other noise can hinder this
  - Keystroke dynamics: believed to be unique
    - Keystroke intervals, pressure, duration of stroke, where key is struck
    - Statistical tests used

Cautions

- These can be fooled!
  - Assumes biometric device accurate *in the environment it is being used in!*
  - Transmission of data to validator is tamperproof, correct

What You Have

What You Have

Location

Location

- If you know where user is, validate identity by seeing if person is where the user is
  - Requires special-purpose hardware to locate user
    - GPS (global positioning system) device gives location signature of entity
    - Host uses LSS (location signature sensor) to get signature for entity

Multiple Methods

Multiple Methods

- Example: “where you are” also requires entity to have LSS and GPS, so also “what you have”
- Can assign different methods to different tasks
- As users perform more and more sensitive tasks, must authenticate in more and more ways (presumably, more stringently) File describes authentication required
- Includes controls on access (time of day, *etc*.), resources, and requests to change passwords

Key Points

Key Points

- For authentication, consider system requirements and components
- Passwords are here to stay
- One-time passwords
- Biometrics
- What you have
- Protocols are important
- Authentication methods can be combined