THE UNIVERSITY OF BRITISH COLUMBIA

Authentication

EECE 412

What is Authentication?

- Real-world and computer world examples?
- What is a result of authentication?
- What are the means for in the digital world?

Outline

- Basics and terminology
- Passwords
  - Storage
  - Selection
  - Breaking them
- Other methods
- Multiple methods

THE UNIVERSITY OF BRITISH COLUMBIA

Basics and Terminology

What is Authentication

binding of identity to subject

- Identity is that of external entity
- Subject is computer entity
- Subject a.k.a. principal

What Authentication Factors are used?

- What you know
- What you have
- What you are

THE UNIVERSITY OF BRITISH COLUMBIA

Password-based Authentication

What's Password?

- Sequence of characters
  - Lots of things act as passwords!
    - PIN
    - Social security number
    - Mother's maiden name
    - Date of birth
    - Name of your pet, etc.
- Sequence of words
  - Examples: pass-phrases
- Algorithms
  - Examples: challenge-response, one-time passwords

Why Passwords?

- Why is “something you know” more popular than “something you have” and “something you are”?
- **Cost**: passwords are free
- **Convenience**: easier for SA to reset password than to issue new smartcard

Keys vs Passwords

- **Crypto keys**
- Spse key is 64 bits
- Then 2^{64} keys
- Choose key at random
- Then attacker must try about 2^{63} keys

- **Passwords**
- Spse passwords are 8 characters, and 256 different characters
- Then $256^8 = 2^{64}$ pwds
- Users do not select passwords at random
- Attacker has far less than 2^{63} pwds to try (**dictionary attack**)

Why not Crypto Keys?

"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations.

(They are also large, expensive to maintain, difficult to manage, and they pollute the environment.

It is astonishing that these devices continue to be manufactured and deployed.

But they are sufficiently pervasive that we must design our protocols around their limitations.)"

Charlie Kaufman, Radia Perlman, Mike Speciner in "Network Security: Private Communication in a Public World"

Good and Bad Passwords

- Bad passwords
  - frank
  - Fido
  - password
  - 4444
  - Pikachu
  - 102560
  - AustinStamp
- Good Passwords?
  - jfIej,43j-EmmL+y
  - 09864376537263
  - P0kem0N
  - FSa7Yago
  - 0nceuP0nAt1m8
  - PokeGCTall150

Password Experiment

- Three groups of users — each group advised to select passwords as follows
  - **Group A:** At least 6 chars, 1 non-letter
  - winner → **Group B:** Password based on passphrase
  - **Group C:** 8 random characters
- Results
  - **Group A:** About 30% of pwds easy to crack
  - **Group B:** About 10% cracked
    - Passwords easy to remember
  - **Group C:** About 10% cracked
    - Passwords hard to remember

Password Experiment

- User compliance hard to achieve
- In each case, 1/3rd did not comply (and about 1/3rd of those easy to crack!)
- Assigned passwords sometimes best
- If passwords not assigned, best advice is
  - Choose passwords based on passphrase
  - Use pwd cracking tool to test for weak pwds
  - Require periodic password changes?

Attacks on Passwords

- Attacker could…
  - Target one particular account
  - Target any account on system
  - Target any account on any system
  - Attempt denial of service (DoS) attack
- Common attack path
  - Outsider → normal user → administrator
  - May only require **one** weak password!

Password Retry

- Suppose system locks after 3 bad passwords. How long should it lock?
  - 5 seconds
  - 5 minutes
  - Until SA restores service
- What are +'s and -'s of each?

How to Store Passwords in the System?

1. Store as cleartext
   - If password file compromised, *all* passwords revealed
2. Encipher file
   - Need to have decipherment, encipherment keys in memory
3. Store one-way hash of password

Password File

- Bad idea to store passwords in a file
- But need a way to verify passwords
- Cryptographic solution: **hash** the passwords
  - Store y = hash(password)
  - Can verify entered password by hashing
  - If attacker obtains password file, he does not obtain passwords
  - But attacker with password file can guess x and check whether y = hash(x)
  - If so, attacker has found password!

Dictionary Attack

- "online" or "offline"
- Attacker pre-computes hash(x) for all x in a **dictionary** of common passwords
- Suppose attacker gets access to password file containing hashed passwords
  - Attacker only needs to compare hashes to his pre-computed dictionary
  - Same attack will work each time
- Can we prevent this attack? Or at least make attacker's job more difficult?

Password File

- Store hashed passwords
- Better to hash with **salt**
- Given password, choose random s, compute

  y = hash(password, s)

  and store the pair (s,y) in the password file
- Note: The salt s is **not secret**
- Easy to verify password
- Attacker must recompute dictionary hashes for each user — lots more work!

Password Cracking: Do the Math

- Assumptions
- Pwds are 8 chars, 128 choices per character
  - Then $128^8 = 2^{56}$ possible passwords
- There is a **password file** with 2^{10} pwds
- Attacker has **dictionary** of 2^{20} common pwds
- Probability of 1/4 that a pwd is in dictionary
- **Work** is measured by number of hashes

Password Cracking

- Attack 1 password without dictionary
  - Must try $2^{56}/2 = 2^{55}$ on average
  - Just like exhaustive key search
- Attack 1 password with dictionary
  - Expected work is about

$$1/4\ (2^{19}) + 3/4\ (2^{55}) = 2^{54.6}$$

  - But in practice, try all in dictionary and quit if not found — work is at most 2^{20} and probability of success is 1/4

Password Cracking

- Attack any of 1024 passwords in file
- **Without** dictionary
  - Assume all 2^{10} passwords are distinct
  - Need 2^{55} comparisons before expect to find password
  - If no salt, each hash computation gives 2^{10} comparisons $\Rightarrow$ the expected work (number of hashes) is $2^{55}/2^{10} = 2^{45}$
  - If salt is used, expected work is 2^{55} since each comparison requires a new hash computation

Password Cracking

- Attack any of 1024 passwords in file
- **With** dictionary
  - Probability at least one password is in dictionary is $1 - (3/4)^{1024} = 1$
  - We ignore case where no pwd is in dictionary
  - If no salt, work is about $2^{19}/2^{10} = 2^9$
  - If salt, expected work is less than 2^{22}
  - Note: If no salt, we can precompute all dictionary hashes and amortize the work

Other Password Issues

- Too many passwords to remember
  - Results in password reuse
  - Why is this a problem?
- Who suffers from bad password?
  - Login password vs ATM PIN
- Failure to change default passwords
- Social engineering
- Error logs may contain "almost" passwords
- Bugs, keystroke logging, spyware, etc.

Passwords

- The bottom line
- **Password cracking is too easy!**
  - One weak password may break security
  - Users choose bad passwords
  - Social engineering attacks, etc.
- The bad guy has all of the advantages
- All of the math favors bad guys
- Passwords are a **big** security problem

How to Improve Password-based Systems?

1. Against off-line password guessing
   - Random selection
   - Pronounceable passwords
     - przbqxdfl, zxrptglfn
     - helgoret, juttelon
   - User selection of passwords
     - Proactive password checking for “goodness”
   - Password aging
2. Against guessing many accounts
   - Salting
3. Against on-line password guessing
   - Backoff
   - Disconnection
   - Disabling
   - Jailing

THE UNIVERSITY OF BRITISH COLUMBIA

Biometrics

What's 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

- 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

Ideal Biometric

- **Universal** — applies to (almost) everyone
  - In reality, no biometric applies to everyone
- **Distinguishing** — distinguish with certainty
  - In reality, cannot hope for 100% certainty
- **Permanent** — physical characteristic being measured never changes
  - In reality, want it to remain valid for a long time
- **Collectable** — easy to collect required data
  - Depends on whether subjects are cooperative
- Safe, easy to use, etc., etc.

Biometric Errors

- **Fraud rate** versus **insult rate**
  - Fraud — user A mis-authenticated as user B
  - Insult — user A not authenticate as user A
- For any biometric, can decrease fraud or insult, but other will increase
- For example
  - 99% voiceprint match ⇒ low fraud, high insult
  - 30% voiceprint match ⇒ high fraud, low insult
- **Equal error rate:** rate where fraud == insult
  - The best measure for comparing biometrics

Cautions

can be fooled!

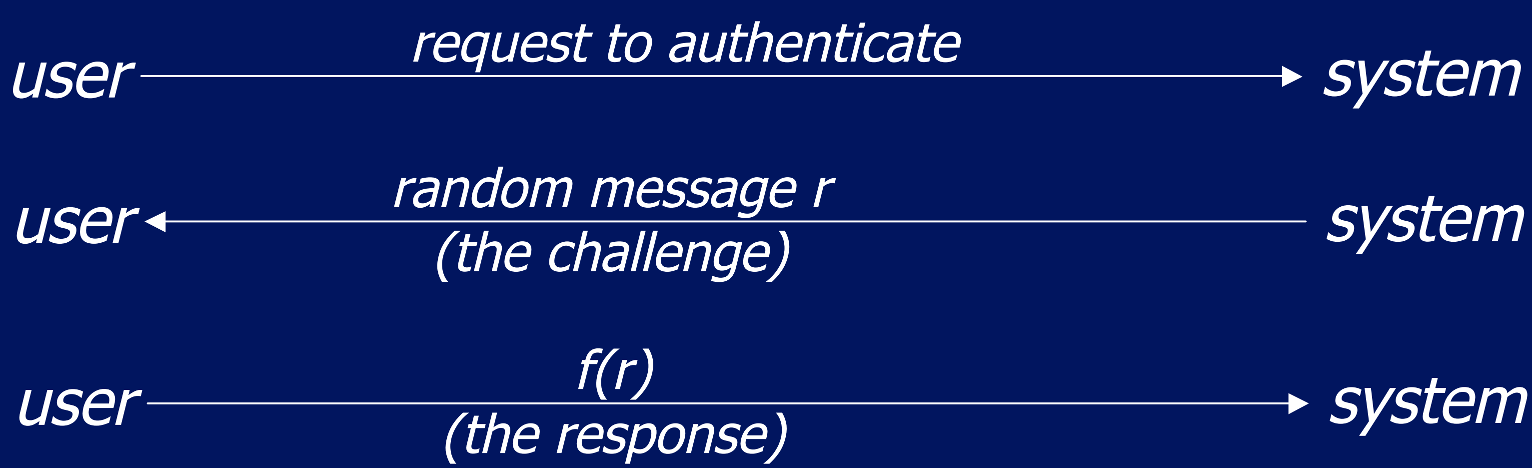
- Assumes biometric device accurate *in the environment it is being used in!*
- Transmission of data to validator is tamperproof, correct

THE UNIVERSITY OF BRITISH COLUMBIA

Authentication Systems based on Challenge-Response

Challenge-Response

User, system share a secret function *f*
(or known function with unknown parameters)



Example: Authentication in GSM

Phone & system share 16-byte secret *k*

GSM phone —— *request to authenticate* ——→ *GSM system*

GSM phone ←—— *random 16-byte challenge c* —— *GSM system*

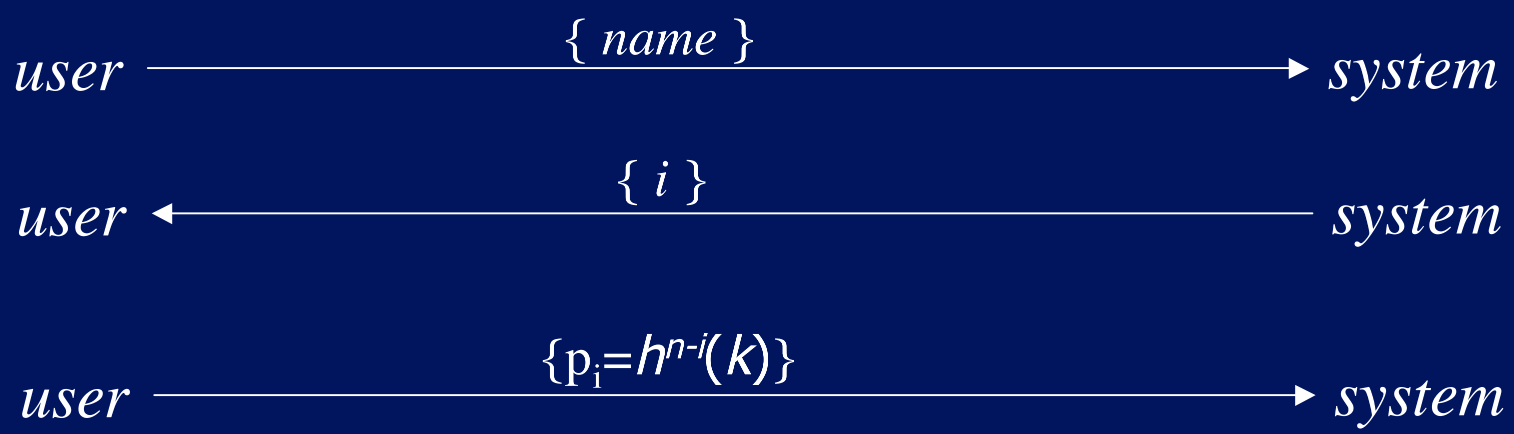
GSM phone —— *Hash(c || k)* ——→ *GSM system*

One-Time Passwords

- Password that can be used exactly *once*
  - After use, it is immediately invalidated
- Challenge-response mechanism
  - Challenge: number of authentications
  - Response: password for that particular number
- Problems
  - Synchronization of user, system
  - Generation of good random passwords
  - Password distribution problem
- How to solve the problems?

S/Key Protocol

- $h(k)$, $h^1(k)$, …, $h^{n-1}(k)$, $h^n(k)$
- Passwords: $p_1 = h^{n-1}(k)$, $p_2 = h^{n-2}(k)$, …, $p_{n-1} = h(k)$, $p_n = k$



What does the system store?

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

Key Points

- Authentication is not just about cryptography
  - You have to consider system components
- Passwords are here to stay
  - They provide a basis for most forms of authentication
- Multi-factor Authentication