
Authentication

CS461/ECE422

Fall 2007

Reading

- Chapter 4.5 from *Security in Computing*
- Chapter 10 from *Handbook of Applied Cryptography*

<http://www.cacr.math.uwaterloo.ca/hac/about>

Overview

- Basics of an authentication system
- Passwords
 - Storage
 - Selection
 - Breaking them
 - One time
- Challenge Response
- Biometrics

Ivanhoe, Sir Walter Scott

- Paraphrased:

(Wamba gains entry to the castle dressed as a friar)

Wamba: Take my disguise and escape, I will stay and die in your place.

Cedric: I can't possibly impersonate a friar, I only speak English.

Wamba: If anyone says anything to you, just say "*Pax vobiscum.*"

Cedric: What does that mean?

Wamba: I don't know, but it works like a charm!

Basics

- Authentication: binding of identity to subject
 - Identity is that of external entity (my identity, the Illini Union Bookstore, *etc.*)
 - Subject is computer entity (process, network connection, *etc.*)

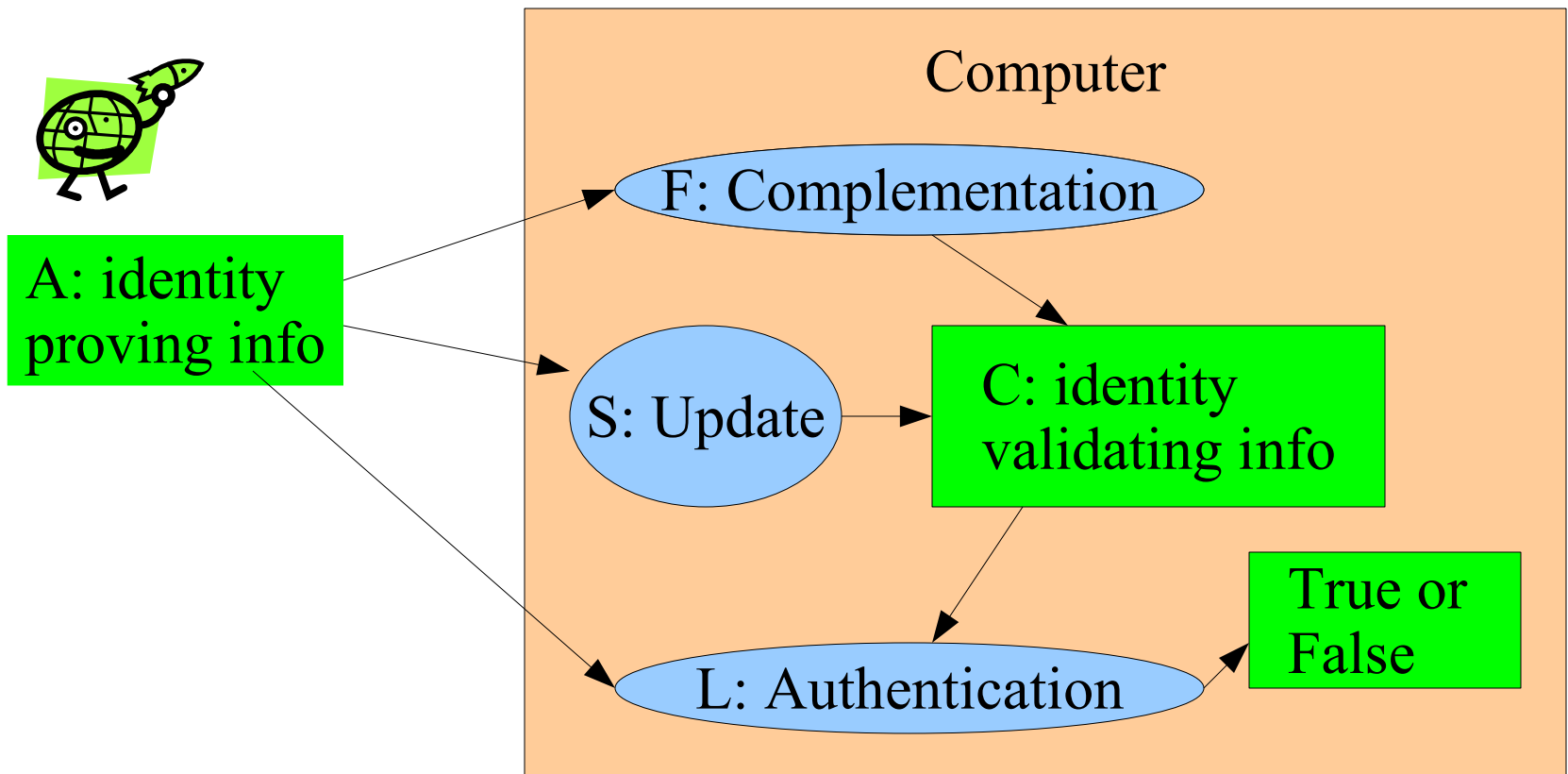
Establishing Identity

- One or more of the following
 - What entity knows (*e.g.* password, private key)
 - What entity has (*e.g.* badge, smart card)
 - What entity is (*e.g.* fingerprints, retinal characteristics)
 - Where entity is (*e.g.* In front of a particular terminal)
- Example: scene from *Ivanhoe*
- Example: Credit card transaction

Authentication System

- (A, C, F, L, S)
 - A : information that proves identity
 - C : information stored on computer and used to validate authentication information
 - F : set of complementation functions that generates C ;
 $f: A \rightarrow C$
 - L : set of authentication functions that verify identity;
 $l: A \times C \rightarrow \{ \mathbf{true}, \mathbf{false} \}$
 - S : functions enabling entity to create, alter information in A or C

Authentication System



Example

- Password system, with passwords stored online in clear text
 - A set of strings making up passwords
 - $C = A$
 - F singleton set of identity function $\{ I(a) = a \}$
 - L single equality test function $\{ \mathbf{eq} \}$
 - S 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

Example

- Original UNIX system standard hash function
 - Hashes password into 11 char string using one of 4096 hash functions
- As authentication system:
 - $A = \{ \text{strings of 8 chars or less} \}$
 - $C = \{ 2 \text{ char hash id} \parallel 11 \text{ char hash} \}$
 - $F = \{ 4096 \text{ versions of modified DES} \}$
 - $L = \{ \textit{login}, \textit{su}, \dots \}$
 - $S = \{ \textit{passwd}, \textit{nispasswd}, \textit{passwd}^+, \dots \}$

Dictionary Attacks

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

Preventing Attacks

- How to prevent this:
 - Hide information so that either a , f , or c cannot be found
 - Prevents obvious attack from above
 - Example: UNIX/Linux shadow password files
 - Hides c 's
 - Block access to all $l \in L$ or result of $l(a, c)$
 - Prevents attacker from knowing if guess succeeded
 - Example: preventing *any* logins to an account from a network
 - Prevents knowing results of l (or accessing l)

Using Time

Anderson's formula:

- 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 $P \geq \frac{TG}{N}$

Example

- Goal
 - Passwords drawn from a 96-char alphabet
 - Can test 10^4 guesses per second
 - Probability of a success to be 0.5 over a 365 day period
 - What is minimum password length?
- Solution
 - $N \geq TG/P = (365 \times 24 \times 60 \times 60) \times 10^4 / 0.5 = 6.31 \times 10^{11}$
 - Choose s such that $\sum_{j=0}^s 96^j \geq N$
 - So $s \geq 6$, meaning passwords must be at least 6 chars long
 - What exactly does that equation mean?

Approaches: Password Selection

- Random selection
 - Any password from A equally likely to be selected
 - See previous example
 - Make sure it's random! (e.g. period of 2^{32} is not enough for $(26+10)^8$ passwords)
- Key crunching (e.g. hashing) a long key to a sequence of shorter keys
- Pronounceable passwords
- User selection of passwords

Pronounceable Passwords

- Generate phonemes randomly
 - Phoneme is unit of sound, *e.g.* *cv*, *vc*, *cvc*, *vcv*
 - Examples: *helgoret*, *juttelon are*; *przbqxdf1*, *zxrptglfn* are not
- ~ 440 possible phonemes
- 440^6 possible keys with 6 phonemes (12-18 characters long), about the same as 96^8
- Used by GNU Mailman mailing list software (?)

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, “133t-speak”, conjugations or declensions, Torah/Bible/Koran/... words)
 - Too short, digits only, letters only
 - License plates, acronyms, social security numbers
 - Personal characteristics or foibles (pet names, nicknames, *etc.*)

Picking Good Passwords

- Examples from textbook
 - “LlMm*2^Ap”
 - Names of members of 2 families
 - “OoHeO/FSK”
 - Second letter of each word of length 4 or more in third line of third verse of Star-Spangled Banner, followed by “/”, followed by author’s initials
- What’s good here may be bad there
 - “DMC/MHmh” bad at Dartmouth (“Dartmouth Medical Center/Mary Hitchcock memorial hospital”), ok here
- Why are these now bad passwords? ☹️

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
 - Spell checker, for example
 - Easy to set up and integrate into password selection system

Salting

- Goal: slow dictionary attacks
- Method: perturb hash function so that:
 - Parameter controls *which* hash function is used
 - Parameter differs for each password
 - So given n password hashes, and therefore n salts, need to hash guess n
- Doesn't help against attacking a single user
 - Does help in bulk attack

Examples

- Vanilla UNIX method
 - Use DES to encipher 0 message with password as key; iterate 25 times
 - Perturb E table in DES in one of 4096 ways
 - 12 bit salt flips entries 0–11 with entries 24–35
 - E Table is per round expansion table
- Alternate methods
 - Use salt as first part of input to hash function

Guessing Through L

- Cannot prevent these
 - Otherwise, legitimate users cannot log in
- Make them slow
 - Backoff
 - Disconnection
 - Disabling
 - Be very careful with administrative accounts!
 - Jailing
 - Allow in, but restrict activities

Leaking Information

- User friendly system gives cause of login failure
 - Bad user vs bad password
- Speed of response may give clue

Detecting Trojan Login

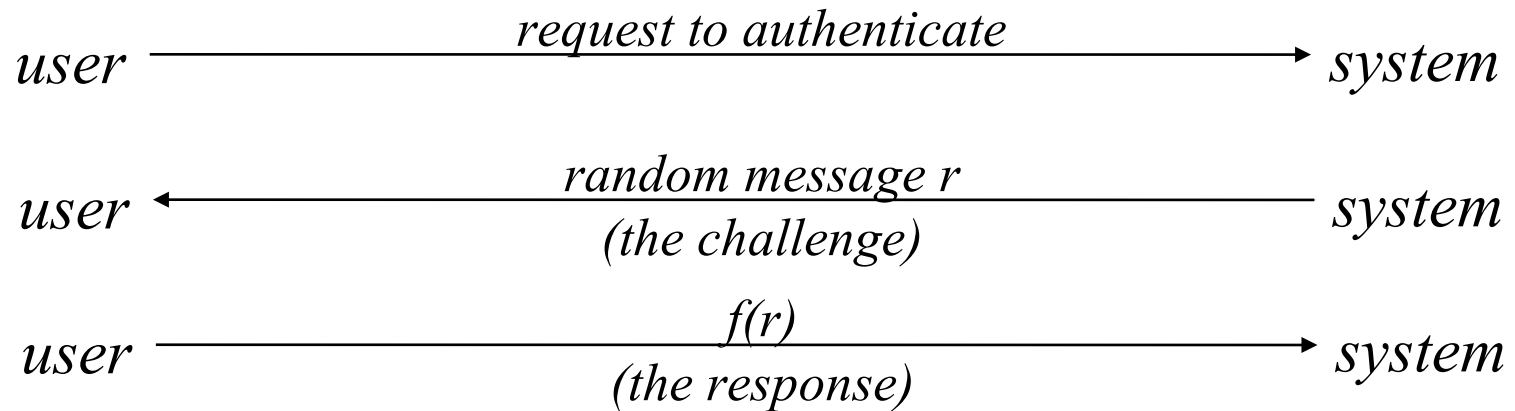
- Trusted path
 - More in next couple lectures
- Recognize last login time stamp
- Recognize previously selected picture
- Certificate

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

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

- Password that can be used exactly *once*
 - After use, it is immediately invalidated
- Challenge-response mechanism
 - Challenge is one of a 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 based on idea of Lamport
- h one-way hash function (MD5 or SHA-1, for example)
- User chooses initial seed k
- System calculates:

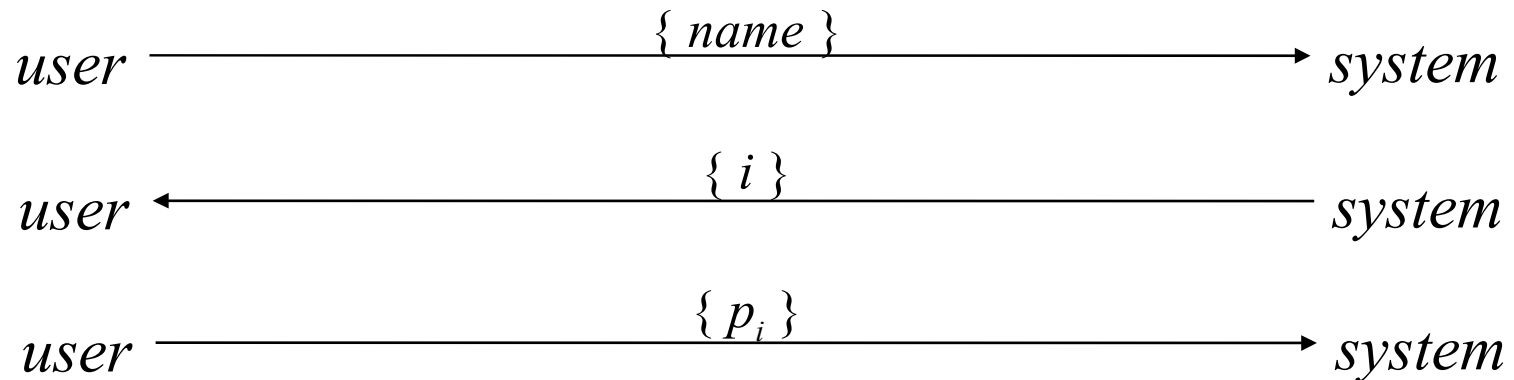
$$h(k) = k_1, h(k_1) = k_2, \dots, h(k_{n-1}) = k_n$$

- Passwords are reverse order:

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

S/Key Protocol

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



System computes $h(p_i) = h(k_{n-i+1}) = k_{n-i+2} = p_{i-1}$. If match with what is stored, system replaces p_{i-1} with p_i and increments i . (Note error in the *CSA&S* textbook.)

Hardware Support

- Token-based
 - Used to compute response to challenge
 - May encipher or hash challenge
 - May require PIN from user
- Temporally-based
 - Every minute (or so) different number shown
 - Computer knows what number to expect when
 - User enters number and fixed password

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)

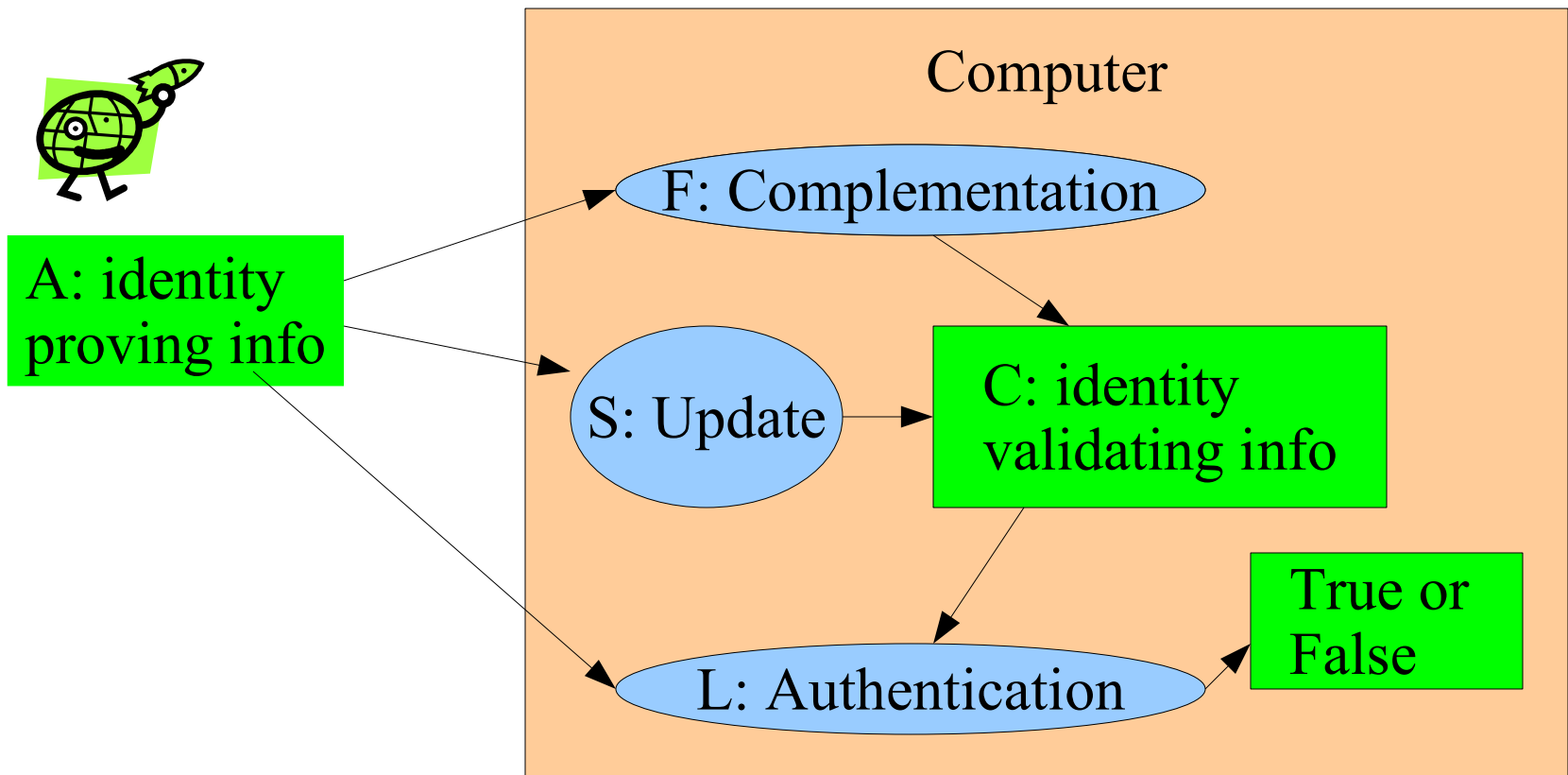
Other Characteristics

- 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

Biometric

- Physical characteristics encoded in a template
 - The C or complement information
- User registers physical information (S)
 - Generally with multiple measurements
- The L function takes a measurement and tries to line up with template

Authentication System



Authentication vs Identification

- Used for surveillance
 - Subject is motivated to avoid detection
- Used for authentication
 - Subject is motivated to positively identify
 - Perhaps pick up other's characteristics
- False positives vs false negatives

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 (remember *pax vobiscum*)
- Physical characteristics change over time
- Some people may not be able to identify via specific characteristics
 - Albinos and iris scans

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

- 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
 - Also includes controls on access (time of day, *etc.*), resources, and requests to change passwords
 - Pluggable Authentication Modules

Key Points

- Authentication \neq cryptography
 - You have to consider system components
- Passwords are here to stay
 - They provide a basis for most forms of authentication
- Biometrics can help but not magic bullet