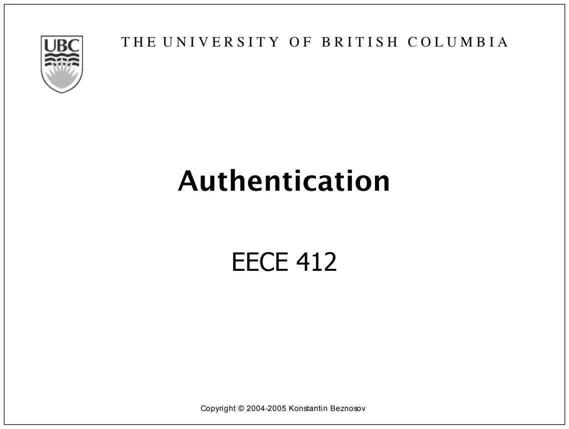
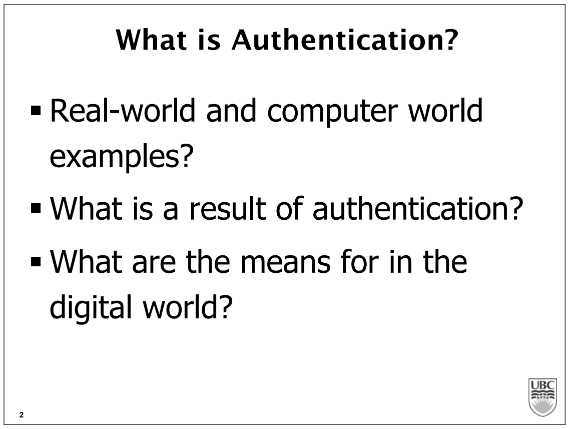


THE UNIVERSITY OF BRITISH COLUMBIA

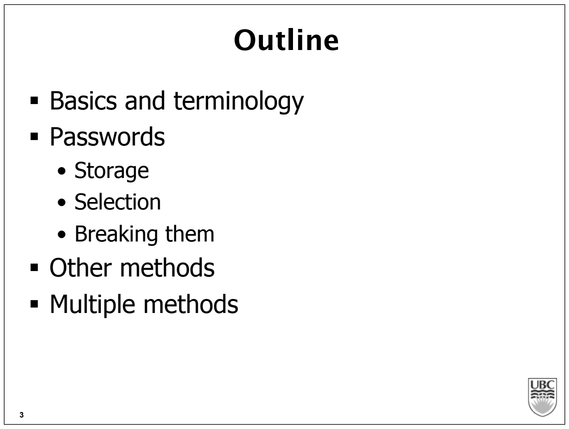
Authentication

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?

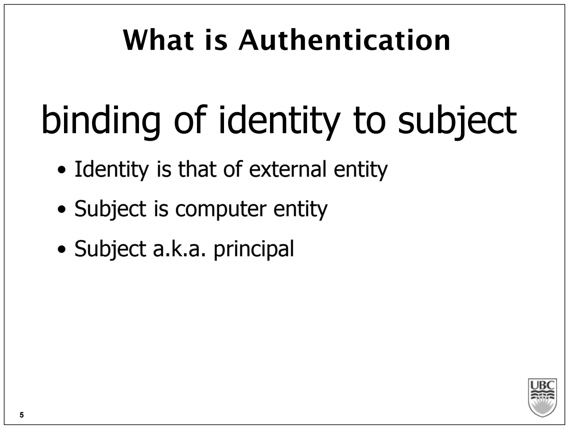


Outline

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

THE UNIVERSITY OF BRITISH COLUMBIA

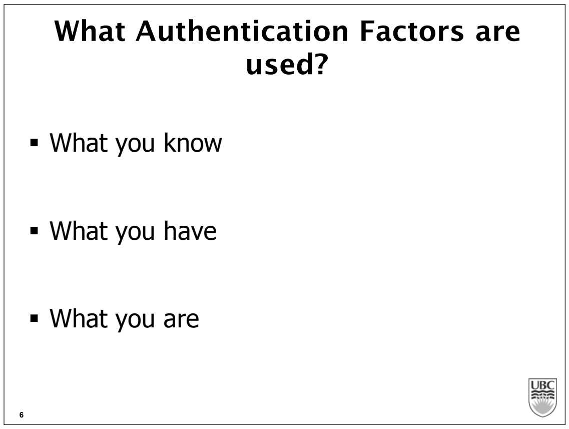
Basics and Terminology



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

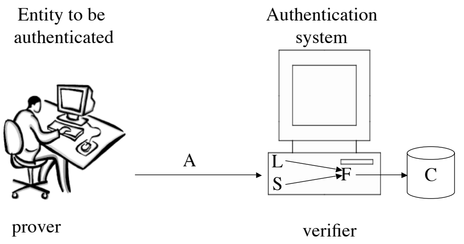
Authentication System Definition

(*A*, *C*, *F*, *L*, *S*)

- *A* -- authentication information
  - Used to prove identity
- *C* -- complementary information
  - stored on computer and used to validate information from A
- *F* -- complementation functions
  - generate c ∈ C from a ∈ A
  - $f : A \to C$
- *L* -- authentication functions
  - verify identity:
  - $l: A \times C \to$ { true, false }
- *S* -- selection functions
  - enable an entity to *create* or *alter* information in *A* or *C*

7

Authentication System: prover and verifier



8

Example

passwords stored on-line in clear text

- *A* set of strings making up passwords
- *C* = *A*
- *F* singleton set of identity function *F* = { *I* }
- *L* single equality test function *L* = { *eq* }
- *S* function to set/change password

9

THE UNIVERSITY OF BRITISH COLUMBIA

Attacking Authentication Systems

Attacking Authentication Systems

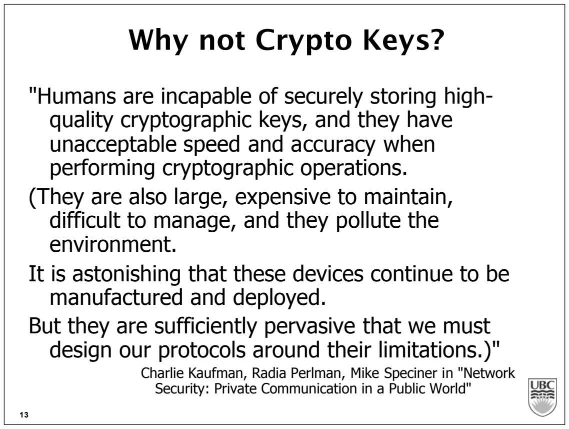
- Attack goal(s)?
- Goal: find $a \in A$ such that:
  - For some $f \in F$, $f(a) = c \in C$
  - *c* is associated with entity
- How to determine whether *a* meets these requirements?
  - Direct approach: as above
  - Indirect approach: as $l(a)$ succeeds iff $f(a) = c \in C$ for some *c* associated with an entity, compute $l(a)$

11

Preventing Attacks

- How to prevent?
  - Hide one of *a*, *f*, or *c*
    - Example: UNIX/Linux shadow password files
      - Hides *c*'s
  - Block access to all $l \in L$ or result of $l(a)$
    - 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*)

12



Why not Crypto Keys?

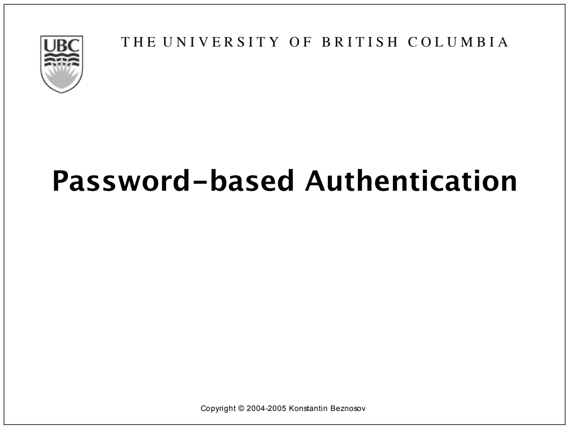
"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations.

(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.

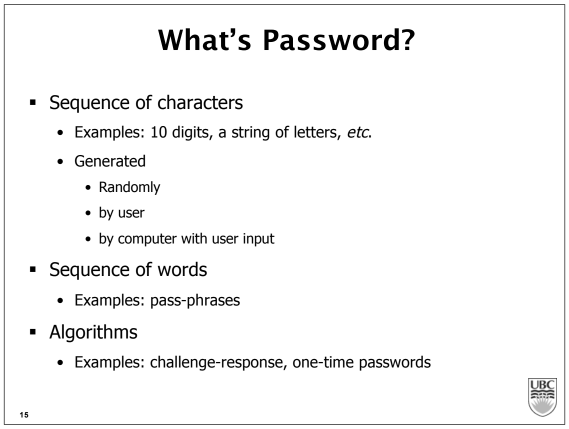
But they are sufficiently pervasive that we must design our protocols around their limitations.)"

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



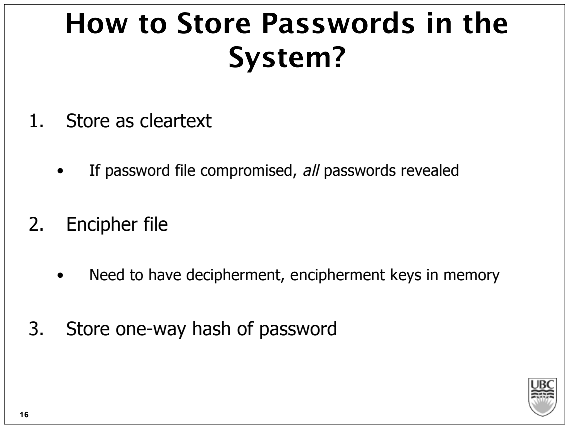
THE UNIVERSITY OF BRITISH COLUMBIA

Password-based Authentication



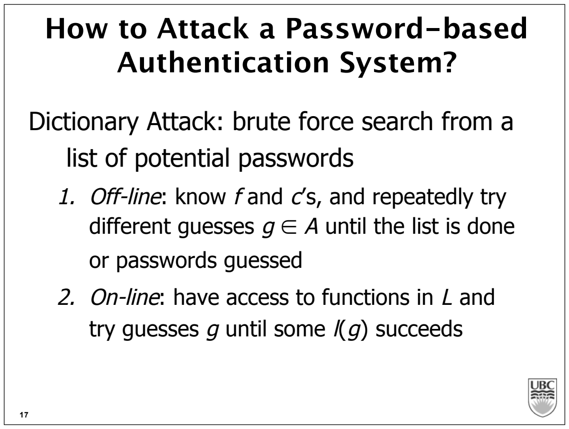
What's Password?

- Sequence of characters
  - Examples: 10 digits, a string of letters, *etc*.
  - Generated
    - Randomly
    - by user
    - by computer with user input
- Sequence of words
  - Examples: pass-phrases
- Algorithms
  - Examples: challenge-response, one-time passwords



How to Store Passwords in the System?

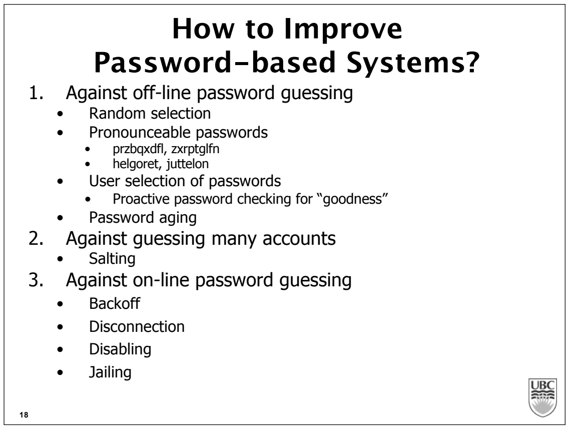
1. Store as cleartext
   - If password file compromised, *all* passwords revealed
2. Encipher file
   - Need to have decipherment, encipherment keys in memory
3. Store one-way hash of password



How to Attack a Password-based Authentication System?

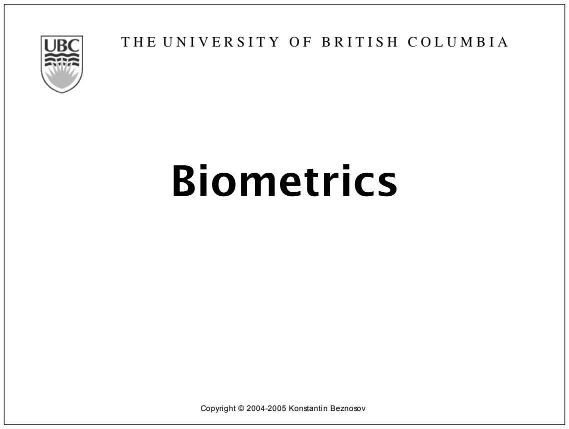
Dictionary Attack: brute force search from a list of potential passwords

1. *Off-line*: know f and c's, and repeatedly try different guesses $g \in A$ until the list is done or passwords guessed
2. *On-line*: have access to functions in L and try guesses g until some $l(g)$ succeeds



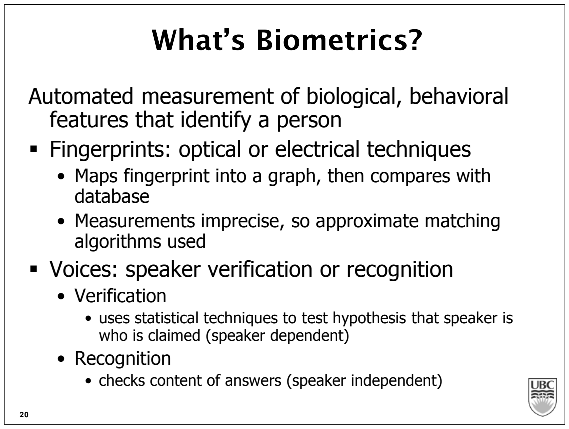
How to Improve Password-based Systems?

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

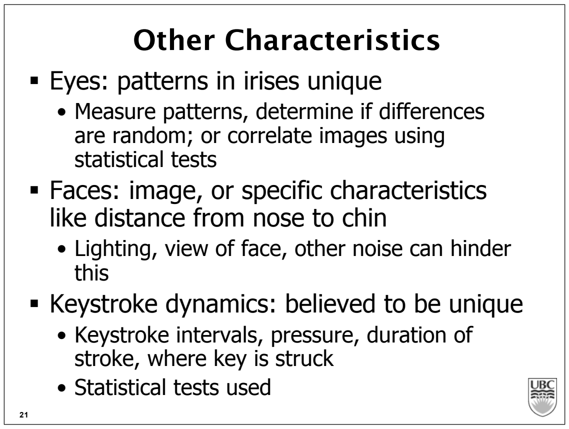
Biometrics



What's Biometrics?

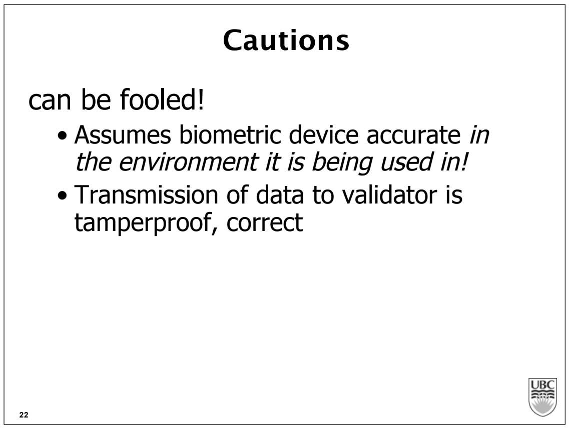
Automated measurement of biological, behavioral features that identify a person

- 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

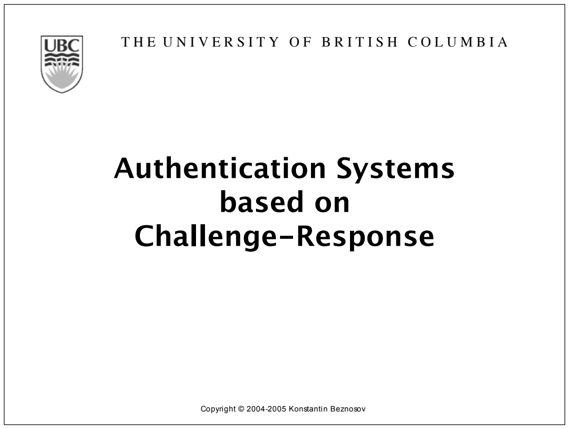
- Eyes: patterns in irises unique
  - Measure patterns, determine if differences are random; or correlate images using statistical tests
- Faces: image, or specific characteristics like distance from nose to chin
  - Lighting, view of face, other noise can hinder this
- Keystroke dynamics: believed to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



Cautions

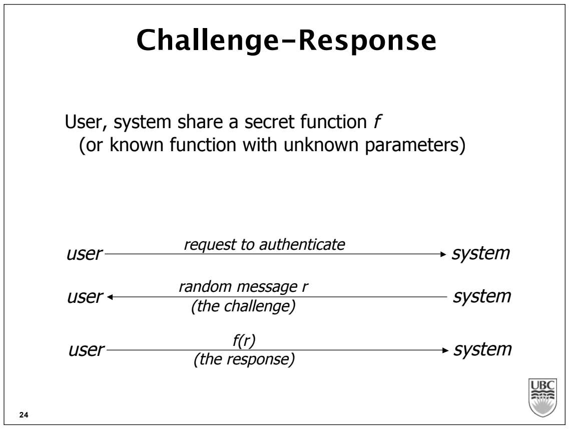
can be fooled!

- 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)

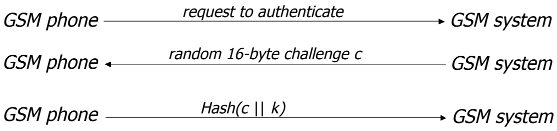
user → *request to authenticate* → *system*

user ← *random message r (the challenge)* ← *system*

user → *f(r) (the response)* → *system*

Example: Authentication in GSM

Phone & system share 16-byte secret *k*

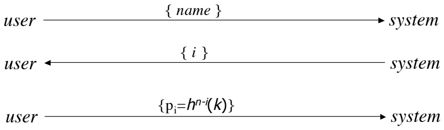


One-Time Passwords

- 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