Computer Security

09. Biometric authentication

Paul Krzyzanowski
Rutgers University
Spring 2017
Biometrics

Identify a person based on physical or behavioral characteristics

```python
scanned_fingerprint = capture();
if (scanned_fingerprint == stored_fingerprint):
    accept_user();
else:
    reject_user();
```
Biometrics

• Rely on statistical pattern recognition
  – Thresholds

• False Accept Rate (FAR)
  – Non-matching pair of biometric data is accepted as a match

• False Reject Rate (FRR)
  – Matching pair of biometric data is rejected as a match
Biometrics

- Each biometric system has a characteristic ROC curve
  - (receiver operator characteristic, a legacy from radio electronics)
Biometrics: forms

- **Fingerprint**
  - Reasonable uniqueness

- **Iris**
  - Analyze pattern of spokes: excellent uniqueness, signal can be normalized for fast matching

- **Retinal scan**
  - Excellent uniqueness but not popular for non-criminals

- **Hand geometry**: length of fingers, width of fingers, thickness, surface area
  - Low guarantee of uniqueness: generally need 1:1 match

- **Signature, Voice**
  - Behavioral vs. physical system
  - Can change with demeanor, tend to have low recognition rates

- **Others**
  - Facial geometry, facial thermographs, DNA, finger vein scans, palm vein scans, odor
Biometrics: distinct features

Example: Fingerprints – identify minutia

Arches
Loops
Whorls
Ridge endings
Bifurcations
Islands
Bridges

source: http://anil299.tripod.com/vol_002_no_001/papers/paper005.html
Biometrics: desirable characteristics

• Robustness
  – Repeatable, not subject to large changes over time
  – Fingerprints & iris patterns are more robust than voice

• Distinctiveness
  – Differences in the pattern among population
  – Fingerprints: typically 40-60 distinct features
  – Irises: typically >250 distinct features
  – Hand geometry: ~1 in 100 people may have a hand with measurements close to yours.
## Biometrics: desirable characteristics

<table>
<thead>
<tr>
<th>Biometric</th>
<th>Robustness</th>
<th>Distinctiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fingerprint</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Hand Geometry</td>
<td>Moderate</td>
<td>Low</td>
</tr>
<tr>
<td>Voice</td>
<td>Moderate</td>
<td>Low</td>
</tr>
<tr>
<td>Iris</td>
<td>High</td>
<td>Ultra high</td>
</tr>
<tr>
<td>Retina</td>
<td>High</td>
<td>Ultra high</td>
</tr>
<tr>
<td>Signature</td>
<td>Low</td>
<td>Moderate</td>
</tr>
</tbody>
</table>
Irises vs. Fingerprints

• Number of features measured:
  – High-end fingerprint systems: ~40-60 features
  – Iris systems: ~240 features

• False accept rates (FAR)
  – Fingerprints: ~ 1:100,000 (varies by vendor; may be ~1:500)
  – Irises: ~ 1:1.2 million
  – Retina scan ~1:10,000,000
Irises vs. Fingerprints

• Ease of data capture
  – More difficult to damage an iris ... but lighting is an issue
  – Feature capture more difficult for fingerprints:
    • Smudges, gloves, dryness, ...

• Ease of searching
  – Fingerprints cannot be normalized
    1:many searches are difficult
  – Irises can be normalized to generate a unique IrisCode
    1:many searches much faster
0. Enrollment
– The user’s entry in a database of biometric signals must be populated.
– Initial sensing and feature extraction
– May be repeated to ensure good feature extraction
Biometric: authentication process

1. Sensing
   - User’s characteristic must be presented to a sensor
   - Output is a function of:
     • Biometric measure
     • The way it is presented
     • Technical characteristics of sensor

2. Feature Extraction
   - Signal processing
   - Extract the desired biometric pattern
     • remove noise and signal losses
     • discard qualities that are not distinctive/repeatable
     • Determine if feature is of “good quality”
3. **Pattern matching**
   - Sample compared to original signal in database
   - Closely matched patterns have “small distances” between them
   - Distances will hardly ever be 0 (perfect match)

4. **Decision**
   - Decide if the match is close enough
   - Trade-off:
     ⦫ false non-matches leads to ⦪false matches
Identification vs. Verification

• **Identification**: *Who is this?*
  – *1:many* search

• **Verification**: *Is this Bob?*
  – Present a name, PIN, token
  – *1:1* (or *1:small #*) search
Biometrics: Essential characteristics

- Trusted sensor
- Liveness testing
- Tamper resistance
- Secure communication
- Acceptable thresholds
Biometrics: other characteristics

• **Cooperative systems** (multi-factor)
  – User provides identity, such as name and/or PIN

• **vs. Non-cooperative**
  – Users cannot be relied on to identify themselves
  – Need to search large portion of database

• **Overt vs. covert identification**

• **Habituated vs. non-habituated**
  – Do users regularly use (train) the system
Problems with biometric systems

• Requires a sensor
  – Camera works OK for iris scans & facial detection
    (but a good Iris scan will also take IR light into account)

• Tampering with device or device link
  – Replace sensed data– or just feed new data

• Tampering with stored data

• Biometric data cannot be compartmentalized
  – You cannot have different data for your Amazon & bank accounts

• Biometric data can be stolen
  – Photos, lifting fingerprints
  – Once biometric data is compromised, it remains compromised
    • You cannot change your iris or finger
Detecting Humanness
Gestalt Psychology (1922-1923)

• Max Wertheimer, Kurt Koffka

• Laws of organization
  – Proximity
    • We tend to group things together that are close together in space
  – Similarity
    • We tend to group things together that are similar
  – Good Continuation
    • We tend to perceive things in good form
  – Closure
    • We tend to make our experience as complete as possible
  – Figure and Ground
    • We tend to organize our perceptions by distinguishing between a figure and a background

Source: http://www.webrenovators.com/psych/GestaltPsychology.htm
Authenticating humanness

**Battle the Bots**
- Create a test that is easy for humans but extremely difficult for computers

**CAPTCHA**
- Completely Automated Public Turing test to tell Computers and Humans Apart
- Image Degradation
  - Exploit our limits in OCR technology
  - Leverages human Gestalt psychology: reconstruction

**Origins**
- 1997: AltaVista – prevent bots from adding URLs to the search engine
- 2000: Yahoo! and Manuel Blum & team at CMU
  - EZ-Gimpy: one of 850 words
- Henry Baird @ CMU & Monica Chew at UCB
  - BaffleText: generates a few words + random non-English words
CAPTCHA Example

Microsoft

See captchas.net
Problems

• Accessibility
  – Visual impairment → audio CAPTCHAs
  – Deaf-blind users suffer

• Frustration
  – OCR & computer vision has improved a lot!
  – Challenges that are difficult for computers may be difficult for humans

• Attacks
  – Man in the middle (sort of)
    • Use human labor – CAPTCHA farms
  – Automated CAPTCHA solvers
    • Initially, educated guesses over a small vocabulary
Alternate approaches

• MAPTCHA = math CAPTCHA
  – Solve a simple math problem

• Puzzles, scene recognition
reCAPTCHA

• Ask users to translate images of real words & numbers from archival texts
  – Human labor fixed up the archives of the New York Times

• Two sections
  – One for known text and the other is the image text
  – Assume that if you get one right then you get the next one correct
    • Try it again on a few other people to ensure identical answers before marking it correct

• Google bought reCAPTCHA 2009
  – Used free human labor to improve transcription of old books & street data

2014: Google found that AI could crack CAPTCHA & reCAPTCHA images with 99.8% accuracy
NoCAPTCHA reCAPTCHA

*Ask users if they are robots*

- Reputation management
  - “Advanced Risk Analysis backend”
  - Check IP addresses of known bots
  - Check Google cookies from your browser
  - Considers user’s entire engagement with the CAPTCHA: before, during, and after
    - Mouse movements & acceleration, precise location of clicks

- Newest version: *invisible reCAPTCHA*
  - Don’t even present a checkbox
If risk analysis fails,
- Present a CAPTCHA
- For mobile users, present a image labeling problem
Alternative: Text/email verification

- **Text/email verification**
  - Ask users for a phone # or email address
  - Service sends a message containing a verification code
    - Still susceptible to spamming
    - Makes it a bit more difficult … and slower

- **Measure form completion times**
  - Users take longer than bots to fill out and submit forms
  - Measure completion times
    - Bots can program delays if they realize this is being done
Code Integrity
Review: signed messages

Message M → Hash(M) → Ea(H(M))

Encrypt with Alice’s private key
= digital signature
We can sign code too

• Validate integrity of the code
  – If the signature matches, then the code has not been modified

• Enables
  – Distribution from untrusted sources
  – Distribution over untrusted channels
  – Detection of modifications by malware

• Signature = encrypted hash signed by trusted source
  – Does not validate the code is good … just where it comes from
Code Integrity: signed software

- Windows 7-10: Microsoft Authenticode
  - `SignTool` command
  - Hashes stored in system catalog or signed & embedded in the file
  - Microsoft-tested drivers are signed

- macOS
  - `codesign` command
  - Hashes & certificate chain stored in file

- Also Android & iOS
Code signing: Microsoft Authenticode

A format for signing executable code (dll, exe, cab, ocx, class files)

- **Software publisher:**
  - Generate a public/private key pair
  - Get a digital certificate: VeriSign class 3 Commercial Software Publisher’s certificate
  - Generate a hash of the code to create a fixed-length digest
  - Encrypt the hash with your private key
  - Combine digest & certificate into a Signature Block
  - Embed Signature Block in executable

- **Microsoft SmartScreen:**
  - Manages reputation based on download history, popularity, anti-virus results

- **Recipient:**
  - Call *WinVerifyTrust* function to validate:
    - Validate certificate, decrypt digest, compare with hash of downloaded code
Per-page hashing

• Integrity check when program is first loaded

• Per-page signatures – improved performance
  – Check hashes for every page upon loading (demand paging)

• Per-page hashes can be disabled optionally on both Windows and macOS
Windows code integrity checks

• Implemented as a file system driver
  – Works with demand paging from executable
  – Check hashes for every page as the page is loaded

• Hashes stored in system catalog or embedded in file along with X.509 certificate.

• Check integrity of boot process
  – Kernel code must be signed or it won’t load
  – Drivers shipped with Windows must be certified or contain a certificate from Microsoft
The End