Broken Fingers: A Deep Dive into Open Source Fingerprint Authentication and its Security Issues

Seong-Joong Kim (E-mail: sungjungk@gmail.com)
Research Staff Member @ National Security Research Institute
Open Source Summit North America 2019
More and more use of fingerprint scanners

*source: IHS Markit 2018*
Biometric must satisfy the following requirements

✓ **Universality**
  - All people possess the feature

✓ **Uniqueness**
  - The feature is different for people, so a biometric system can distinguish among them

✓ **Permanence**
  - The feature only varies slightly over time

Biometric must satisfy the following requirements

✓ **Measurability**
  - A biometric system can acquire and process the feature in an efficient way

✓ **Safe against circumvention**
  - A biometric system can distinguish between the real feature and a dummy

Biometric fingerprints

The features of *Fingerprint* biometric

- Skin on finger folds in a random process
  - Leading to ridges and valleys
- Minutiae points are unique for each human and even each finger
  - Minutiae: ridge ending or bifurcation at certain points
  - Orientation and type (e.g., delta, loop, etc.)
- It provides a fast, convenient and ease of use in various applications
Widely used in various applications (1/2)

✓ **Screen Lock**
  - To be in the safe keeping private & sensitive data on the mobile phone & desktop PC

*Windows Hello enables Win10 users to authenticate secure access to their device with just a fingerprint*
Widely used in various applications (2/2)

✓ **Payment**
  - To provide an easy way to make purchase on the mobile device by using biometric security

[*Liquid offers a payment system using a fingerprint*]  

[Samsung Pay]

Threat: Leaking Fingerprint

Come with an inherent risk:

- Fingerprints cannot be changed for the rest of life

✓ Security issues: login credential, banking credential, etc.

✓ Privacy issues: identity record, criminal history, immigration history, etc.
Related Work: Leaking Fingerprint (1/2)

✓ Collected and physically cloned the biometric features with fingerprint fuming

https://www.youtube.com/watch?v=HM8b8d8kSNQ

✓ Extracted the fingerprint minutiae from high resolution cameras

Related Work: Leaking Fingerprints (2/2)

✓ Proposed ‘MasterPrints’ synthesizing different fingerprints with similarity analysis, which can impersonate users with a given probability


✓ Created sophisticated and natural-looking fingerprints only from the numerical template data

In this talk...

✓ Basic operation of fingerprint recognition

✓ Systematic analysis of ‘fprint’ project
  - Most popular open source for fingerprint applications

✓ Security flaws found in ‘fprint’ and its supported devices
  - Focus on leaking fingerprints
    - Cleartext transmission of sensitive information
    - Use of insufficient initial key length
    - Missing encryption of sensitive data
Fingerprint Recognition
Fingerprint Sensors (1/2)

✓ Various technologies have been used to capture fingerprints
  - Optical, capacitive, ultrasonic, thermal, pressure, etc.
✓ Today, the most popular ones are the optical & capacitive sensors

Fingerprint Sensors (2/2)

✓ **Optical Sensor**
- Shine a light on a prism ➔ collect the reflections by a camera
- Reflection: ridge
- Scatter: valley
- Use the effect of frustrated reflection properties

✓ **Capacitive Sensor**
- Provide an array of single capacitor plates to the sensor surface
- Measure the capacitance of each plate on the material above
- Distinguish between the skin of the ridges and the air in the valleys
Fingerprint Recognition (1/7)

✓ Step 1. Preprocess - reducing noise and errors

✓ Graphically preprocess the created image after capturing a fingerprint
  - Image binarization
    - Black pixel ➔ ridge
    - White pixel ➔ valley
  - Image thinning the ridges to a width of one pixel

[Fingerprint image before (left) and after (right) ridge thinning]
Fingerprint Recognition (2/7)

✓ Step 2. Extract minutiae

✓ One of the most popular technique is the minutiae-based matching
  ✓ Provide storage efficiency and accuracy to apply matching algorithm
  ✓ Make an intermediate image, called template, to make it comparable and easy to store

✓ There are commonly used fingerprint classes in minutiae-based matching
  ✓ Tented arch, left & right loop, whorl
Fingerprint Recognition (3/7)

✓ Step 2. Extract minutiae

✓ Arch
  ✓ Ridges that enter from one side rise to a small bump and go out the opposite side (no singularity is presented)

✓ Tented arch
  ✓ Similar to the above arch, except that some ridges
  ✓ Exhibit a high curvature which has one loop and one delta (usually vertically aligned)
Step 2. Extract minutiae

- **Left & right loop**
  - Ridges that enter from the left or right side, curve back, and exit from the same side they entered

- **Whorl**
  - Contain two opposite loops at the same location
  - The most complex among the classes and further divided into subclasses
Fingerprint Recognition (5/7)

✓ Step 3. Make a fingerprint template

✓ Storing the minutiae points’ relative position, orientation, and type
  - E.g., ISO/IEC 19794-2, XYT-format, etc.

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Valid Values and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format ID</td>
<td>4 bytes</td>
<td>‘F’ ‘M’ ‘R’ 0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Horizontal Size</td>
<td>2 bytes</td>
<td>in pixels</td>
</tr>
<tr>
<td>Image Vertical Size</td>
<td>2 bytes</td>
<td>in pixels</td>
</tr>
<tr>
<td>Horizontal Resolution</td>
<td>2 bytes</td>
<td>in pixels per cm</td>
</tr>
<tr>
<td>Vertical Resolution</td>
<td>2 bytes</td>
<td>in pixels per cm</td>
</tr>
<tr>
<td>Number of Finger Views (n_v)</td>
<td>1 byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finger Position</td>
<td>1 byte</td>
<td>0 to 10</td>
</tr>
<tr>
<td>View Number</td>
<td>4 bits</td>
<td>0 to 15</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Minutiae (n)</td>
<td>1 byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>Type</td>
<td>2 bits</td>
<td>{00=other, 01=termination, 10=bifurcation}</td>
</tr>
<tr>
<td>Position x</td>
<td>14 bits</td>
<td>in pixels</td>
</tr>
<tr>
<td>Reserved</td>
<td>2 bits</td>
<td></td>
</tr>
<tr>
<td>Position y</td>
<td>14 bits</td>
<td>in pixels</td>
</tr>
<tr>
<td>Direction (\theta)</td>
<td>1 byte</td>
<td>0 to 255 (resolution 1.40625 degrees)</td>
</tr>
<tr>
<td>Quality</td>
<td>1 byte</td>
<td>1 to 100 (0=quality not reported)</td>
</tr>
<tr>
<td>Extended Data Block Length</td>
<td>2 bytes</td>
<td></td>
</tr>
<tr>
<td>Extended Data Area Type Code</td>
<td>2 bytes</td>
<td>only present if Extended Data Block Length&gt;0</td>
</tr>
<tr>
<td>Extended Data Area Length</td>
<td>2 bytes</td>
<td></td>
</tr>
<tr>
<td>Data Section</td>
<td>(prev. field)</td>
<td></td>
</tr>
</tbody>
</table>
```

*ISO/IEC 19794-2:2005 fingerprint template format*
Fingerprint Recognition (6/7)

✓ Step 3. Make a fingerprint template

✓ Storing the minutiae points’ relative position, orientation, and type
  - E.g., ISO/IEC 19794-2, XYT-format, etc.
✓ Difference between ISO/IEC 19794-2 and XYT-format
  - Coordinate system & angle resolution

<table>
<thead>
<tr>
<th>Format</th>
<th>X, Y Coordinate System</th>
<th>Angle (θ) Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO/IEC 19794-2</td>
<td>0 → X</td>
<td>1.40625°</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>XYT-format</td>
<td>y → X</td>
<td>1°</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Coordinate system & angle resolution of the minutiae points
Fingerprint Recognition (7/7)

- Step 4. Match between minutiae in each fingerprint
  - Build separate tables for the fingerprints being matched
  - Define distance and orientation between minutiae in each fingerprint
  - Create a match score by looking the number of similar minutiae
‘fprint’ Project
fprint Overview (1/2)

✓ Previously...
  - Scattered amongst different projects
    - Incomplete & inconsistent
    - Have to implement support for each type of fingerprint reader separately

✓ ‘fprint’ Project
  - Support for various consumer fingerprint reader devices in Linux
  - Long term shooting for adoption by Linux distributions
    - GNOME desktop supports fingerprint management through its users settings panel
    - Most Linux distributions support fingerprint login through *fprintd*
**fprint Overview (2/2)**

**FPRINT**

- **libfprint**
  - Talking to fingerprint reading devices
  - Processing fingerprint data
  - Driver: image decode
  - Driver: libusb

- **fprintd**
  - Daemon that provides fingerprint related functionality
  - Integrate with fingerprint authentication and authorization application

**Load/store**

**Template Storage**

---

- **libusb**
- **MINDTCT**
- **BOZORTH3**

---

6 NBIS: NIST Biometric Image Software
'MINDTCT: minutiae detection software, 'BOZORTH3: minutiae based matching software
Fingerprint Reader

✓ *Supported devices
  - Support about 120 fingerprint devices including reader and module
  - Only available on the market as standalone device as follows:
    - DigitalPersona U.are.U 4500&5300, UPEK EikonTouch 510&710

✓ Functionalities of the reader
  - Capture a fingerprint and save it as a grayscale image (e.g., $^\text{PGM}$ format)
  - Encrypt that image before sending it to the driver (i.e., libfprint)

$^\text{Portable graymap format}$
*supported devices list: https://fprint.freedesktop.org/supported-devices.html
libfprint (1/9)

 ✓ Driver @libfprint
   - Talk to fingerprint reader and decrypt the fingerprint image
     - Activate the fingerprint reader
     - Request to capture a fingerprint image
     - Decrypt the image after receiving

 ✓ NIST Biometric Image Software (NBIS) @libfprint
   - Perform image processing of the fingerprint image
     - Minutiae extraction software (MINDTCT)
       ➔ Enroll & verify
     - Minutiae based matching software (BOZORTH3)
       ➔ Verify
libfprint (2/9)

- **Talking & Decryption**
  - Receive an encrypted fingerprint image from the reader and decrypt that image
- **Minutiae Extraction**
  - Extract minutiae from that image and make a standard template
- **Minutiae-based Matching**
  - Compare different minutiae points from the template
libfprint (3/9)

✓ Talking & decryption

- Initialization
  - Discover target fingerprint device and initialize it
    - Optionally, CR-based authentication

- Ready to scan
  - Optionally, capture an empty initial scanned image

- Await fingerprint on
  - Place a finger on the scanner
  - Capture a fingerprints and make a grayscale image
libfprint (4/9)

✓ Talking & decryption

- Generate a key
  - For using the encryption

- Encrypt fingerprint image
  - Cipher the fingerprint image using the generated key

- Forward fingerprint image & encryption key
  - Transfer the encrypted image and the key to the driver (libfprint)
libfprint (5/9)

✓ Talking & decryption

- Decrypt image
  - Use a Linear Feedback Shift Register (LFSR)-based synchronous stream cipher algorithm

\[
D_{S_i}(C_i) = C_i \oplus S_i = (P_i \oplus S_i) \oplus S_i = P_i
\]

✓ Binary additive stream cipher
libfprint (6/9)

✓ Image processing

- Minutiae detection software (MINDTCT)
  - Locates all minutiae in the fingerprint image, assigning to each point of location, orientation/angle, type and quality

Grayscale  Binarization  Minutiae
libfprint (7/9)

✓ Image processing

- Minutiae detection software (MINDTCT)
  - Image quality map
    - Measure an information regarding the quality of regions in the image
    - Represent unstable areas in the image where minutiae detection is unreliable

- Direction map
  - Derive and record a directional ridge flows
  - Represent areas of the image with sufficient ridge structure
libfprint (8/9)

✓ Image processing

- Minutiae detection software (MINDTCT)
  - Image binarization
    - Black pixel as a ridge and white one as a valley
  - Detect minutiae
    - Scan the binary image of a fingerprint, identifying localized pixel patterns that indicate the ending or bifurcating of a ridge
- Data cleansing
  - Remove minutiae in regions of poor image quality
libfprint (9/9)

✓ Image processing

- Minutiae based matching software (BOZORTH3)
  - Builds separate tables for the fingerprints being matched
    - Defined distance and orientation between minutiae in each fingerprints
  - Use the location \((x, y)\) and orientation \((\theta)\) of the minutiae points to match the fingerprints
fprintd

✔ Daemon that provides fingerprint related functions through D-Bus API
  - Enroll
    - Request to capture a fingerprint to the reader
    - (MINDTCT) Extract minutiae from the fingerprint
    - Make a template file & store the file on local filesystem
  - Verify & Identify
    - Request to capture a fingerprint to the reader
    - (MINDTCT) Extract minutiae from the fingerprint
    - (BOZORTH3) Load the template(s) & compare them (using threshold)

✔ Integrate with fingerprint authentication and authorization application, spanning from desktop app. to PAM
Security Issues
Security Issues

✓ Cryptographic issues
  - Cleartext transmission of sensitive information
    - Key (initial seed) for LFSR-based PRNG
  - Short key vulnerability
    - Vulnerable to brute-force attack

✓ Sensitive data exposure
  - Missing encryption or data protection scheme of sensitive data
    - Fingerprint template
Cryptographic Issues (1/5)

✓ Synchronous stream cipher – LFSR based

Key Stream Generator

Update Key

Key

Key Stream Generator

Key

\[ E_{S_i}(P_i) = P_i \oplus S_i = C_i \]

\[ D_{S_i}(C_i) = C_i \oplus S_i = (P_i \oplus S_i) \oplus S_i = P_i \]

@fingerprint reader

@libfprint driver

✓ Key Stream Generator

- LFSR output streams are deterministic

- If the current state and the positions of the XOR bits in the LFSR are known, the next state can be predicted

If an attacker knows Key & LFSR internal

- Can reproduce ‘key stream’ & decrypt the cipher
Cryptographic Issues (2/5)

- LFSR-based key stream generator internal in libfprint
- Publicly available the position of the XOR bits in the LFSR

```c
for (i = 0; i < num_bytes - 1; i++) {
    /* calculate xor byte and update key */
    xorbyte = ((key >> 4) & 1) << 0;
    xorbyte |= ((key >> 8) & 1) << 1;
    xorbyte |= ((key >> 11) & 1) << 2;
    xorbyte |= ((key >> 14) & 1) << 3;
    xorbyte |= ((key >> 18) & 1) << 4;
    xorbyte |= ((key >> 21) & 1) << 5;
    xorbyte |= ((key >> 24) & 1) << 6;
    xorbyte |= ((key >> 29) & 1) << 7;
    key = update_key(key);

    /* decrypt data */
    data[i] = data[i+1] ^ xorbyte;
}
```

/* Linear feedback shift register */
* taps at bit positions 1 3 4 7 11 13 20 23 26 29 32 */
bit ^= bit << 16;
bit ^= bit << 8;
bit ^= bit << 4;
bit ^= bit << 2;
bit ^= bit << 1;

Open source nature!!
Cryptographic Issues (3/5)

✓ Cleartext transmission of sensitive information
Cryptographic Issues (4/5)

✅ Cleartext transmission of sensitive information

Key Stream Generator

Update Key

Key

Vulnerability Description
- Key used as a seed for ‘Key Stream Generator’ is exhibited as a cleartext during transfers
- Once the key has been stolen, an attacker can replay the key stream and decipher the fingerprint image

PoC
- [https://github.com/sungjungk/fp-scanner-hacking](https://github.com/sungjungk/fp-scanner-hacking)

Demo
- [https://youtu.be/C4MMfQdfyws](https://youtu.be/C4MMfQdfyws)

*CVE-2019-12813 has been assigned*
Cryptographic Issues (5/5)

✓ Short key vulnerability

✓ Vulnerability Description
- Key for encryption the fingerprint image is vulnerable to brute-force attack
  - Short for just 32-bit
  - This allows an attacker to recover the initial key (seed) and decrypt that image using the key stream

✓ Brute-force attack
- Procedure
  - Decode a set of bytes (image block)
  - Measure an entropy of the decoded results
  - Set a threshold for determining the results
  - Reduce time to decryption image
  - Narrow range of target to be decrypted
  - Employ a parallel programming (OpenMP model)

✓ PoC
- https://github.com/sungjungk/fp-img-key-crack

✓ Demo
- https://youtu.be/AwZh_HS9sRU

*CVE-2019-13604 has been assigned

\[ D_{S_i}(C_i) = C_i \oplus S_i = (P_i \oplus S_i) \oplus S_i = P_i \]
Sensitive data exposure

✓ Missing encryption or protection scheme of sensitive data
  - 'fprintd' does not provide data protection scheme fingerprint templates (XYT-format) before storage
    - Just locate template file on the host

✓ Threat on fingerprint image reconstruction from template
  - Create sophisticated and natural-looking fingerprints from the template
    - ISO/IEC 19794-2 template ➔ Fingerprint image
  - Successfully evaluated it against NBIS

Demo
Thank you