Performance evaluation of cancelable biometrics

Nalini K Ratha*
Exploratory Computer Vision Group
IBM T. J. Watson Research Center
Hawthorne, NY 10532

*Joint work with members of biometrics research team
Cancelable Biometrics

- Intentional repeatable distortion
  - Generates a similar signal each time for the same user

- Compromised scenario:
  - a new distortion creates a new biometrics

- Comparison scenario:
  - different distortions for different accounts

- Backwards compatibility
  - Representation is not changed.
Cancelability requirements of the transform

1. The intrinsic strength (individuality) of the biometric should not be reduced after transformation. (Constraint on FAR)
   \[ D(x_1, x_2) > t \Rightarrow D(T(x_1), T(x_2)) > t \]

2. The transformation should be tolerant to intra-user variation (Constraint on FRR)
   \[ D(x_1, x_2) < t \Rightarrow D(T(x_1), T(x_2)) < t \]

3. The original should not match with the transform,
   \[ D(x, T(x)) > t \]

4. Different transforms of the same user should not match with each other
   \[ D(T_1(x), T_2(x)) > t \]
Registration based
Challenges

Transformation

Registration

Same?

Intra-user variation
Feature Domain Transformation

Feature Extraction

Cartesian Transformation

Polar Transformation

Surface Folding Transformation
How does it affect accuracy?

Same transform for all users

Different transforms for different users

- Results reported in
  - “Cancelable biometrics: A case study in Fingerprints”, ICPR 06
  - “Generating cancelable fingerprint templates”, IEEE PAMI
Registration free
Enrollment
Verification
Cancelable methods

- Can we avoid storing the original patch signatures?
- Ways to transform/hide the feature vector
  - Encryption - representation too unstable for encryption
  - Polynomial transformation
  - Random projection - fits well with NDP distance

Preferred: Ortho normal projections

Polynomial transformation

Random Projections
Cancelability (2)

- Each patch can be used to produce multiple transforms
Cancelability (3)

- Original match among themselves
- Transforms match among themselves
- Transform does not match with original
Cancelability (4)

- Score more than 0.5 is a mismatch
- Different Transforms don’t match with each other
Empirical Results (1)

- **Patch based verification**
  - Performance is less than geometry based matchers (62% GAR at 0.01% FAR)

- **Cancelable features**
  - Complete separation (100% GAR, 0% FAR) achieved by having separate transforms for separate individuals

- **Diversity of key space**
  - Complete separation (100% GAR, 0% FAR) achieved for separate (188) transforms of the same individual

- **Non invertiblity**
  - Complete separation (100% GAR, 0% FAR) achieved for non-invertible construction as well

- Perfect performance because uses entropy from key also

- If everyone uses the same key performance will not change because distances are preserved
Increasing security: Two factor transformation

- The current construction is invertible
  - If we have the projecting matrix $B$, and the transform $T(x) = B^T x$
  - $x = B T(x) = B B^T x$, can be recovered

- Can we increase security?

- Two factor transformation
  - The projection matrix $B$ is constructed using two orthonormal matrices $U, V$
    
    $B = U V^T$
    
    $U U^T = U^T U = V V^T = V^T V = I$
    
    $B B^T = (U V^T) U^T = U (V^T V) U^T = I$
    
    $U, V$ are obtained by performing SVD on a random matrix $R = U S V^T$
    
    $S$ is not recorded anywhere in the system.
    
    $U, V$ do not leak information about each other

- $U$ and $V$ can be separately stored separately (e.g. split between user and application?)
- Symmetric key, public key comparison
More security: Non-invertibility

- We can make the construction non-invertible by introducing some non-linearity
  Define,
  \[ T(x) = \begin{cases} 
  1 & \text{if } B^T x > 0, \ (B = U V^T) \\
  0 & \text{otherwise} 
  \end{cases} \]
- Thus, even if U, V, T(x) are known, it is impossible to recover x from T(x)
- Advantages:
  - The construction is non-invertible
- Disadvantages
  - Brute force attack is easier. (More pre-images of \( B^T x \) produce the same sign)
Invariant features

- Independent triangle features
  - The sides
- Dependent triangle feature
  - Height at largest side
- Fingerprint features
  - Minutiae angles with respect to triangle

INDEX

\[
\begin{align*}
  a_1, \quad a_2, \quad a_3, \\
  s_1, \quad s_2, \quad s_3, \\
  \ldots
\end{align*}
\]
Triangles can be enumerated

\[ (s_1, s_2, s_3) \]

Quantize

Constrain side lengths

\[ s_1, s_2, s_3 \text{ quantized using } p \text{ bits} \]
Enrolment

Minutia Triplets

Triangle Indexing

Triangle Hashing

Binarization

Mutation

Randomization

Encryption
Verification

![Diagram showing the verification process involving fingerprint matching and indexation.](image)
Steps in building a cancelable iris system

- Segmentation
- Feature extraction
- Cancelable techniques
Method 1: GRAY COMBO

- **template based row shift and combination**
  - Step 1: for each row shift circularly:

- Step 2: combine two rows together to get a new one:
  - Intensity +, -
  - One row can be used more than once
  - Easy methods: odd+even, fold like a mirror

Combine rows 1, 3 to the new 1\(^{st}\) row
Combine rows 2, 8 to the new 2\(^{nd}\) row
Combine rows 4, 6 to the new 3\(^{rd}\) row
Combine rows 5, 7 to the new 4\(^{th}\) row
Method 2: BIN COMBO

- **code based row shift and combination**
  - Step 1: for each row shift circularly:
    - Code:
      - ![Code Image]
    - Diagram:
      - ![Diagram Image]

- Step 2: combine two rows together to get a new one:
  - Binary XOR, or NXOR
  - One row can be used more than once
  - Easy methods: odd+even, fold like a mirror

<table>
<thead>
<tr>
<th>Step</th>
<th>Combine Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 3 to the new 1&lt;sup&gt;st&lt;/sup&gt; row</td>
</tr>
<tr>
<td>2</td>
<td>2, 8 to the new 2&lt;sup&gt;nd&lt;/sup&gt; row</td>
</tr>
<tr>
<td>3</td>
<td>4, 6 to the new 3&lt;sup&gt;rd&lt;/sup&gt; row</td>
</tr>
<tr>
<td>4</td>
<td>5, 7 to the new 4&lt;sup&gt;th&lt;/sup&gt; row</td>
</tr>
</tbody>
</table>
Method 3: GRAY SALT

- **template based salty noise**
  - Just plus a unique pattern --- random noise, random pattern or random synthetic iris texture
  - Generate new code according to the new texture
Method 4: BIN SALT

- code based salty noise
  - Just plus a unique binary pattern --- random noise, random pattern or random synthetic iris code
Matcher

- Assume head tilt is not heavy
- Matching algorithm need to be modified:
Key performance metrics

- **Accuracy**
  - How do the error rates change?
    - Same transform vs. different transform

- **Transform space**
  - How many transforms are possible?
    - Brute force non-invertible strength of the transform

- **Backward compatibility**

- **Impact on speed**
Thank you