Fuzzy Extractors
Fuzzy extractors: visual intuition

Enrollment phase

(s, w) = Gen(x)
Fuzzy extractors: visual intuition

Reconstruction phase

\[ s' = \text{Rep}(x', w) \]
Is helper data really necessary?
a) All enrollments are finished before authentication
b) Enrollments at any time
Fuzzy extractor: requirements

Correctness:
Pr[S' ≠ S] must be small.
• Various ways to formalize this, e.g. averaged over X, for worst-case X; bound on the Hamming distance, ...

Security:
• W must not leak too much about S.
• Distribution of S must be close to uniform
• Many ways to formalize this statement, e.g. H(S|W) close to ℓ, H∞(S|W) close to ℓ, stat. distance Δ(SW;UW) ≈ 0, ...

\[ \Pr[S' \neq S] \text{ must be small.} \]
• Various ways to formalize this, e.g. averaged over X, for worst-case X; bound on the Hamming distance, ...

\[ \text{Security:} \]
• W must not leak too much about S.
• Distribution of S must be close to uniform
• Many ways to formalize this statement, e.g. H(S|W) close to ℓ, H∞(S|W) close to ℓ, stat. distance Δ(SW;UW) ≈ 0, ...
Secure sketch: requirements

\[
\begin{align*}
X & \rightarrow SS \\
& \downarrow W \\
X' & \rightarrow Rec \\
& \downarrow \hat{X}
\end{align*}
\]

Correctness: \( \Pr[X \neq X'] \) must be small.
- Various ways to formalize this, e.g. averaged over \( X \), for worst-case \( X \); bound on the Hamming distance, ...

Security / Privacy:
- \( W \) must not leak too much about \( X \).
  (But some leakage is inevitable!)
- Many ways to formalize this statement, e.g. \( H(X|W) - H(X) \) as small as possible,
  \( H_\infty(X|W) - H_\infty(X) \) as small as possible, ...
Fuzzy Extractor from Secure Sketch

Trivial construction

• reconstruct x using secure sketch
• then apply strong extractor to derive key from x
  - X is not uniform, key must be uniform
  - needs additional public randomness
Examples of Helper Data Schemes

1. Code Offset Method
2. Partitions
3. Generic Zero Leakage scheme for continuum X
4. Use of Gray codes
5. Reliable component selection for Optical PUFs
6. Reliable component selection for SRAM PUFs
7. Fuzzy Extractor made from Universal Hashes
• $X = \text{binary string}$
• Use a linear Error Correcting Code

Enrollment
• random key $s$
• encode to $c_s$
• $w = c_s - x$

Reconstruction
• $s' = \text{decode}(x' + w)$
Syndrome-only variant of the Code Offset Method

- $X =$ binary string
- Use a linear Error Correcting Code that supports Syndrome Decoding ("SynDec")

**Enrollment:** $w = \text{Syn} \ x$

**Reconstruction:** $\hat{x} = x' \oplus \text{SynDec}(w \oplus \text{Syn} \ x')$

- $\text{Syn}(x \oplus x')$
- gives error pattern
2: Partitions method

- First partition: equiprobable keys $s$
- 2nd partition: helper data $w$, equiprob. subpartitions
- $S | W=w$ is uniform
Why is it good to increase the number of sub-partitions?
3: Limiting case of the partitioning scheme

Number of colours $\rightarrow \infty$.

$W \in [0, 1)$

Set helper data $w$ as:

$$F(x) = \sum_{j=0}^{s-1} p_j + wp_s$$

Claim: $W$ leaks nothing about $S$!
4: Helper data for coating PUF

1. partitioning with $n=2^b$.
2. concatenated Gray code words
3. code offset method
Pointers to prominent features

- stable
- allows for re-alignment
- does not reveal black/white: secret
- [do not choose directly neighboring features]

If necessary apply code offset method in the end
6: Fuzzy Extractor for SRAM PUFs

**Helper data:**
addresses of reasonably stable cells

does not reveal bit value!

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

step 1: majority voting

| 111 | 000 | 1 | 0 |

step 2: code offset method

...
7: Fuzzy Extractor from Universal Hashes

**Enroller**
Measure x.
\[ s = \Phi_r(x); w = \Psi_t(x); \]
\[ v = \Gamma_j(x). \]
\[ a = F(v, w). \]

**MAC on w**

**Device**
Measure x'.
Select neighborhood B of x'.
For \( x_i \) in B: Find matches \( \Psi_t(x_i)=w \).
For these matches compute \( v_i = \Gamma_j(x_i). \)
Find unique match \( F(v_i, w) = a \).
Reconstructed secret is \( s_i = \Phi_r(x_i). \)
Privacy-preserving biometrics

Equivalent of hashed passwords table, but with noisy inputs
## When to use FE or SS

<table>
<thead>
<tr>
<th>Application</th>
<th>privacy of X?</th>
<th>uniform secret?</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>password authent.</td>
<td></td>
<td></td>
<td>One-Way Function</td>
</tr>
<tr>
<td>biometric authent.</td>
<td></td>
<td></td>
<td>Secure Sketch + OWF</td>
</tr>
<tr>
<td>anticounterfeiting PUF</td>
<td></td>
<td></td>
<td>Secure Sketch + OWF</td>
</tr>
<tr>
<td>anticounterfeiting PUF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUF authent. w/o MACs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUF authent. with MACs</td>
<td></td>
<td></td>
<td>Fuzzy Extractor</td>
</tr>
<tr>
<td>POK</td>
<td></td>
<td></td>
<td>Fuzzy Extractor</td>
</tr>
</tbody>
</table>