The spammed Code Offset Method

Boris Škorić
Niels de Vreede

PUFFIN workshop
3 Nov. 2013

eprint.iacr.org/2013/527
Outline

• Helper data schemes
  – privacy-preserving biometric databases
  – Physically Obfuscated Keys
• The Code Offset Method
• Adding fake enrollment data
  – while retaining efficient reconstruction
  – LDPC codes, syndromes, ...
• Analysis
  – security
  – storage/work
    \[\text{trade-off}\]
Scenario 1: privacy-preserving biometrics DB

**Aim:** store only the hash of a user's fingerprint/iris/...

**Problem:** noise

**Solution:** helper data scheme (Secure Sketch)

Desired properties:
- High prob. of correct reconstruction.
- W does not leak much about X.

**Database entry:**

\[
\text{Hash}(W, \text{salt}, X)
\]

**Figure of merit:**

\[H(X|W)\]
Scenario 2: Physically Obfuscated Key

**Aim:** Alternative technology for read-proof key storage. Obtain key from measurement on complex physical system ("PUF").

**Problem:** noise

**Solution:** helper data scheme

Figure of merit: $H_2(X|W)$
Intermezzo: Error-correcting codes

k-bit message $\mu$.
n-bit codeword $C_\mu$.
n-bit noise pattern $e$.

$z = C_\mu + e$

Syndrome $\text{Syn}(z) = \text{Syn}(C_\mu) + \text{Syn}(e) = \text{Syn}(e)$

\[
C_\mu H^T = 0
\]

"Low-Density Parity Check" matrix  [Gallager 1960]
The mother of all Secure Sketches

- Source $X \in \{0,1\}^n$.
- Uniformly random $R \in \{0,1\}^k$.
- Binary linear error correcting code (Enc, Dec).
  Message size $k$; codeword size $n$.
How good is this?

If $X$ is uniform:
- $H(R|W) = H(R)$; **no leakage about $R$!**
- $H(X|W) = H(R) = k$
  - $W$ leaks $n-k$ bits about $X$

If $X$ is *not* uniform
- $W$ leaks about $R$
- $W$ still reveals $\text{Syn}(X)$

Can we do better?
Fake helper data

Idea: hide $W$ in lots of fake helper data (with same distribution)

Biometrics database, entry for one user:

$$W_1^\text{fake} \quad W_2^\text{fake} \quad \ldots \quad W \quad W_z^\text{fake} \quad \ldots \quad W_{m-1}^\text{fake} \quad \text{Hash(Wtable,X)}$$

`random location Z`

Legitimate party:
- Has $X'$
- Reconstruction by brute force: Try all entries

Attacker:
- Does not have $X'$
- Brute force attack
- effort multiplied by $m/2 \Rightarrow \log(m/2)$ bits of security gained
More efficient scheme

• Use LDPC code
  – parity check matrix is sparse
  – \(X' \approx X\) implies \(\text{Syn}(X') \approx \text{Syn}(X)\)

• Store \(\text{Syn}(X) = \text{Syn}(W)\) and all \(\text{Syn}(W^{\text{fake}})\)
  – can be computed from \(W\) and \(W^{\text{fake}}\)
  – reveals nothing new
  – Code Offset Method possible with only syndrome

\[
\begin{align*}
\text{Syn } W_1^{\text{fake}} & \quad \cdots \quad \text{Syn } X & \quad \cdots \quad \text{Syn } W_{m-1}^{\text{fake}} & \quad \text{Hash(table},X) \\
\end{align*}
\]

Fast reconstruction: • Compute \(\text{Syn}(X')\)
  • Prioritize entries with \(\text{Syn}(W_i) \approx \text{Syn}(X)\).
Security analysis

Without spam: \( H(X|W) = H(\text{Syn } X) \)

With spam: \( H(X|\Omega) \geq H(X|W) + \log m - \frac{m - 1}{\ln 2} \mathbb{E}_x q_{\text{Syn}}(x) \)

\[ H(X|\Omega) \geq H(X) - \frac{1}{m} \cdot \frac{2^{n-k} - 1}{\ln 2} \]

\( \Omega \): the helper data list

\( q_a = \text{Prob}[\text{Syn } X=a] \)

Typically, \( \frac{m - 1}{\ln 2} \mathbb{E}_x q_{\text{Syn}}(x) \) is of order \( \frac{m}{2^{n-k}} \)

\( m \rightarrow 2^{n-k}: \) Leakage gets close to zero.
The size of the table (assuming LDPC)

<table>
<thead>
<tr>
<th></th>
<th>biometrics (1 user)</th>
<th>phys. obfuscated key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k=64</td>
<td>k=128</td>
</tr>
<tr>
<td>#err</td>
<td>n</td>
<td>log m</td>
</tr>
<tr>
<td>1</td>
<td>72</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>78</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>85</td>
<td>10.5</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

- n values are approximate
- Listed values for log m: \((n-k)/2\) and \(n-k\)
- Choose \(m\) that fits in memory \(\Rightarrow\) sec. gain \(\log(m)-1\) bits
Summary

We added a new "knob" to the Code Offset Method

- better use of source entropy
- price: size of enrollment data
- security analysis: Shannon entropy
- Rényi entropy [not shown]
- interesting for low source entropy

Work in progress:

- explicitly choose LDPC codes
- generate the table (with PRNG)
  - security ↔ memory tradeoff becomes
  security ↔ work tradeoff
Acknowledgements

• Ruud Pellikaan
• Ludo Tolhuizen