Secure Remote Authentication Using Biometric Data

Vanessa End

December 7, 2009
Structure

1. Introduction

2. Motivation and Basics
   - Problems
   - Secure Sketch and Fuzzy Extractor

3. Generic Scheme
   - Well-Formed Sketch and Robust Sketch
   - Robust Fuzzy Extractors
   - Application to Secure Authentication

4. Improved Solution
   - Construction
   - Correctness

5. Conclusion
Introduction

- show how to achieve mutual authentication and/or authenticated key exchange over completely insecure channel with underlying protocol (e.g. for authenticated key exchange)
- two constructions:
  1. generic solution, protecting against modification of public value through secure sketches and fuzzy extractors
  2. specific to remote authentication and key exchange with improvements to generic solution
- both solutions tolerate stronger class of errors (data-dependant errors), i.e. we no longer have the prerequisite, that the public value stays unharmed
Problems of biometric data

- not uniformly distributed
  - no provable security guarantees
  - problem can be addressed using hash function
- not exactly reproducible
  - this especially means that we need some kind of 'fuzzy' solution, as e.g. hash functions are no option
- former solutions:
  - Boyen et al mostly concentrate on *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data* by Dodis et al where **secure sketches** and **fuzzy extractors** are used
Secure Sketch

A secure sketch makes it possible to recall a shared secret $\omega$ through $\omega'$ “close enough” to $\omega$.

Definition

Let $\mathcal{M}$ be a metric space with some metric $d$. Let $t$ be the error-correction bound. A **secure sketch** is a pair of two procedures, a sketch and a recovery procedure, with:

- **SS** : $\mathcal{M} \rightarrow \{0, 1\}^* ; \omega \mapsto s$, where $s$ is some string
- **Rec** : $\mathcal{M} \times \{0, 1\}^* \rightarrow \mathcal{M} ; (\omega', s) \mapsto \omega$ if $d(\omega, \omega') \leq t$
Secure Sketch and Fuzzy Extractor

Fuzzy Extractor

A fuzzy extractor corrects the non-uniformity of our biometric data by always extracting the same nearly uniform randomness $R$ from two inputs similar enough.

Definition

Let $\mathcal{M}$ be a metric space with some metric $d$. Let $t$ be the error correction bound. A fuzzy extractor is a pair of two procedures, a generating and a reproduction procedure, with:

- **Gen**: $\mathcal{M} \rightarrow \{0, 1\}^{\ell} \times \{0, 1\}^* ; \omega \mapsto (R, \text{pub})$
- **Rep**: $\mathcal{M} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\ell} ; (\omega', \text{pub}) \mapsto R$

if $d(\omega, \omega') \leq t$ and if $(R, \text{pub}) \leftarrow \text{Ext}(\omega)$
Dodis et al then construct their fuzzy extractor from a given secure sketch and a strong (nonfuzzy i.e $t = 0$) extractor: For the generation of $(R, \text{pub})$ they first apply the secure sketch $\text{SS}(\omega) = s$ and then an extractor with randomness $x$ s.t. $\text{Ext}(\omega) = R$. $(s, x)$ is then stored as pub.

![Diagram of the Fuzzy Extractor - Gen](image-url)
Construction of the Fuzzy Extractor - Rep

To reproduce $R$ from $\omega'$ and $\text{pub}$, they first use $\text{Rec}(\omega', s) = \omega$ and then $\text{Ext}(\omega, x) = R$:
Problems of above Notion

- There is no guarantee of output of $\text{Rep}$ or $\text{Rec}$ in case $d(\omega, \omega') > t$
  - Solution: so called *well-formed* sketches
- $\text{pub}$ is sent over insecure network, thus an adversary might modify $\text{pub}$ in transit, without anyone knowing.
  - Solution: so called *robust sketches*
Idea of Well-Formed Sketch

- The **well-formed sketch** provides, that if \( d(\omega,\omega') > t \), the output cannot be \( \omega \) in any case, but rather will be \( \bot \).
- \((SS, Rec)\) is transformed into \((SS, Rec')\), where \( Rec' \) runs \( Rec \) and then verifies, that output \( \omega \) complies with \( d(\omega,\omega') \leq t \). If not the output of \( Rec' \) is \( \bot \).
Well-Formed Sketch and Robust Sketch

Idea of Robust Sketch

- Through the use of a robust sketch, there is a high possibility that the user is able to first detect a modification of pub and then abort in that case.
- In general, a robust sketch is a stronger version of a well-formed sketch complying with the following construction
Construction of Generic Robust Sketch

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ be a hash function. A robust sketch $(SS, Rec)$ is constructed from any well-formed sketch $(SS^*, Rec^*)$ by

\[
\begin{align*}
SS(\omega) \\
\text{pub}^* \leftarrow SS^*(\omega) \\
h = H(\omega, \text{pub}^*) \\
\text{return pub} = (\text{pub}^*, h)
\end{align*}
\]

\[
\begin{align*}
Rec(\omega', \text{pub} = (\text{pub}^*, h)) \\
\omega' = Rec^*(\omega, \text{pub}^*) \\
\text{if } \omega' = \bot \text{ output } \bot \\
\text{if } H(\omega', \text{pub}^*) \neq h \text{ output } \bot \\
\text{otherwise, output } \omega'
\end{align*}
\]
Construction of Robust Fuzzy Extractor

Again the Robust Fuzzy Extractor is constructed, as above, through a robust sketch and a strong extractor. The only difference is, that it is needed to bind the hash function key to the sketch itself, making it a labeled robust sketch.
Initialization

For any secure protocol $\Pi$ based on a uniformly distributed key, any robust fuzzy extractor $(\text{Ext}, \text{Rec})$ and any source $W_0$, the protocol $\Pi'$ is constructed as follows:

**Initialization** User scans $W_0$ to retrieve $\omega_0$ and computes $\text{Ext}(\omega_0) = (R, pub)$ and registers this at the server.
Execution

Protocol Execution  User scans his fingerprint $W_i$ to retrieve $\omega_i$. The server sends $\text{pub}$ to user, who computes $\hat{R} = \text{Ext}(\omega_i, \text{pub})$. If $\hat{R} = \perp$, the user aborts, else $\Pi$ is executed by user (with $\hat{R}$) and server (with $R$).
Correctness

If the user obtains correct $\text{pub}$ from server, then user and server will use same $R$ in procedure $\Pi$, because then $d(\omega_0, \omega_i) \leq t$. 
We assume an active adversary, controlling all messages sent between user and server. There are then two possible outcomes:

- The adversary tries to forward a message $\text{pub}' \neq \text{pub}$. These instances will then abort immediately except for a probability of at most $\epsilon$.

- If the adversary forwards $\text{pub}$ unchanged, the user and the server run $\Pi$. Even if the adversary succeeds in forwarding a changed $\text{pub}'$, the protocol $\Pi$ is assumed to be save and thus the adversary will not 'break in'
Improved Solution
Advantages of Improved Solution

- provably secure in the standard model
- improved bounds on the “effective entropy loss”, as there is no randomness extraction
Let $\Pi$ be a password-only authenticated key exchange protocol and $(SS, \text{Rec})$ be a well-formed sketch. Construct $\Pi'$ as follows:

User $U$ scans $\omega_0$ from $W_0$ and computes $SS(\omega_0) = \text{pub}$. $U$ registers $(\omega_0, \text{pub})$ at server $S$. 
pub is sent to user. Then the server executes Π using the following:

- identity: $S \parallel \text{pub}$
- partner identity: $\text{pid}_S = U \parallel \text{pub}$.
- password: $\omega_0$
Protocol Execution User

User retrieves $\omega_i$ and obtains $SS(\omega_i) = pub'$. With these he computes $Rec(\omega_i, pub') = \omega'$. If $\omega' = \bot$ he aborts, else he executes $\Pi$ using the following:

- identity: $U \parallel pub'$
- partner identity: $pid_U = S \parallel pub'$
- password: $\omega'$
Correctness

With no interference from adversary we have:

1. $\text{pid}_S = U \parallel \text{pub}$
2. $\text{pid}_U = S \parallel \text{pub}$
3. $\omega_0 = \omega'$
• the generic solution relies on random oracles, while the improved one doesn't

• but the generic solution is more efficient and simpler

• both have an underlying, presumably save protocol Π on which much of the security depends
The End