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 adressed 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 Extraxtors: 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 ω through ω' "close enough" to ω.

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** : $\mathcal{M} \to \{0,1\}^*$; $\omega \mapsto s$, where s is some string
- **Rec** : $\mathcal{M} \times \{0,1\}^* \to M$; $(\omega', s) \mapsto \omega$ if $d(\omega, \omega') \leq t$

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.

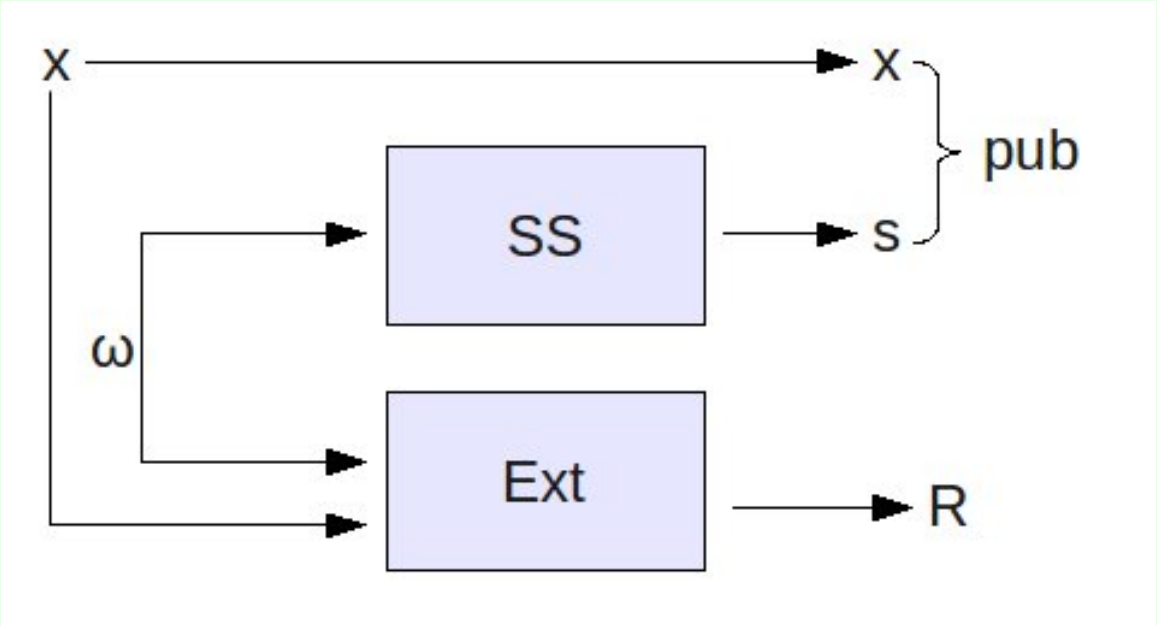
Definition

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** : $\mathcal{M} \to \{0,1\}^\ell \times \{0,1\}^*$; $\omega \mapsto (R, \mathrm{pub})$
- **Rep** : $\mathcal{M} \times \{0,1\}^* \to \{0,1\}^\ell$; $(\omega', \mathrm{pub}) \mapsto R$
  if $d(\omega, \omega') \leq t$ and if $(R, \mathrm{pub}) \leftarrow$ **Ext**(ω)

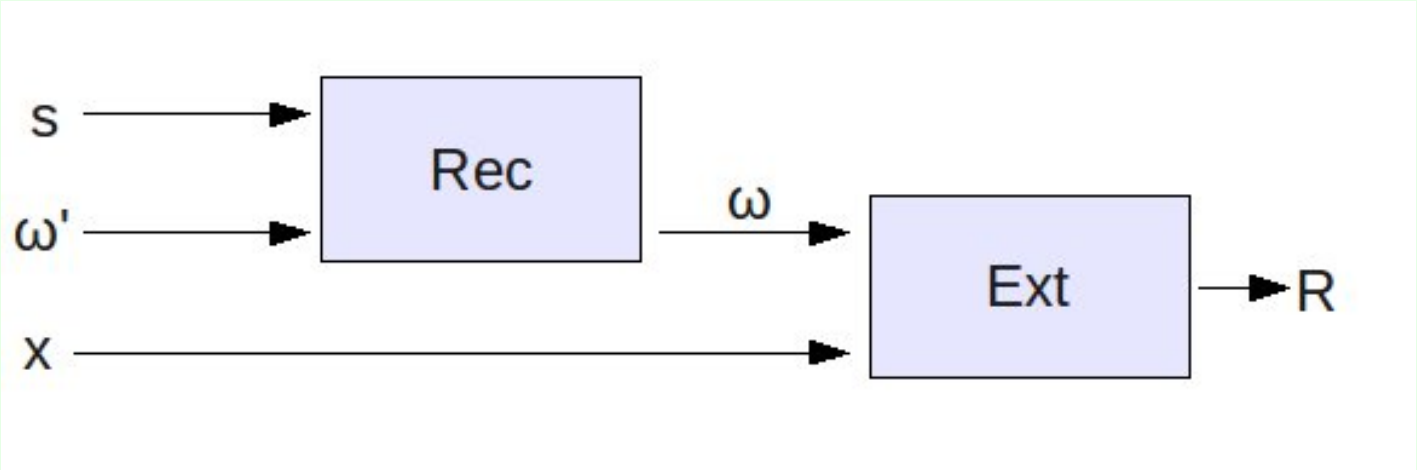
Construction of the Fuzzy Extractor - Gen

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, pub) they first apply the secure sketch $\mathbf{SS}(\omega) = s$ and then an extractor with randomness x s.t. $\mathbf{Ext}(\omega) = R$. (s, x) is then stored as pub.



Construction of the Fuzzy Extractor - Rep

To reproduce R from ω' and pub, they first use $\mathbf{Rec}(\omega', s) = \omega$ and then $\mathbf{Ext}(\omega, x) = R$:

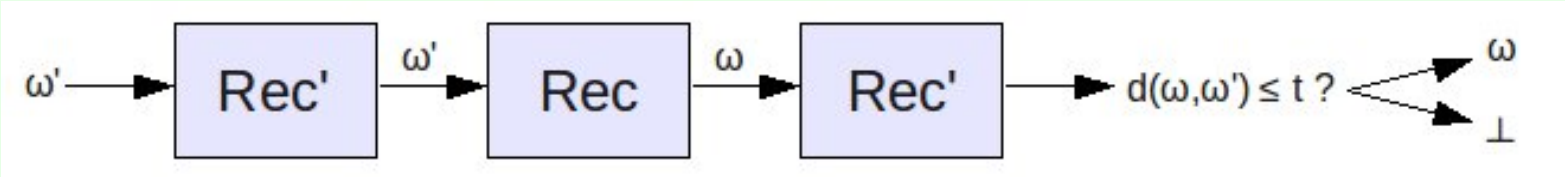


Problems of above Notion

- There is no guarantee of output of **Rep** or **Rec** in case $d(\omega, \omega') > t$
  - Solution: so called *well-formed* sketches
- pub is sent over insecure network, thus an adversary might modify 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 ω in any case, but rather will be $\perp$.
  - $(\mathbf{SS}, \mathbf{Rec})$ is transformed into $(\mathbf{SS}, \mathbf{Rec}')$, where $\mathbf{Rec}'$ runs **Rec** and then verifies, that output ω complies with $d(\omega, \omega') \leq t$. If not the output of $\mathbf{Rec}'$ is $\perp$



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

$\underline{\mathbf{SS}(\omega)}$	$\underline{\mathbf{Rec}(\omega', \mathrm{pub} = (\mathrm{pub}^*, h))}$
$\mathrm{pub}^* \leftarrow \mathbf{SS}^*(\omega)$	$\omega' = \mathbf{Rec}^*(\omega, \mathrm{pub}^*)$
$h = H(\omega, \mathrm{pub}^*)$	if $\omega' = \perp$ output $\perp$
return $\mathrm{pub} = (\mathrm{pub}^*, h)$	if $H(\omega', \mathrm{pub}^*) \neq h$ output $\perp$
	otherwise, output ω'

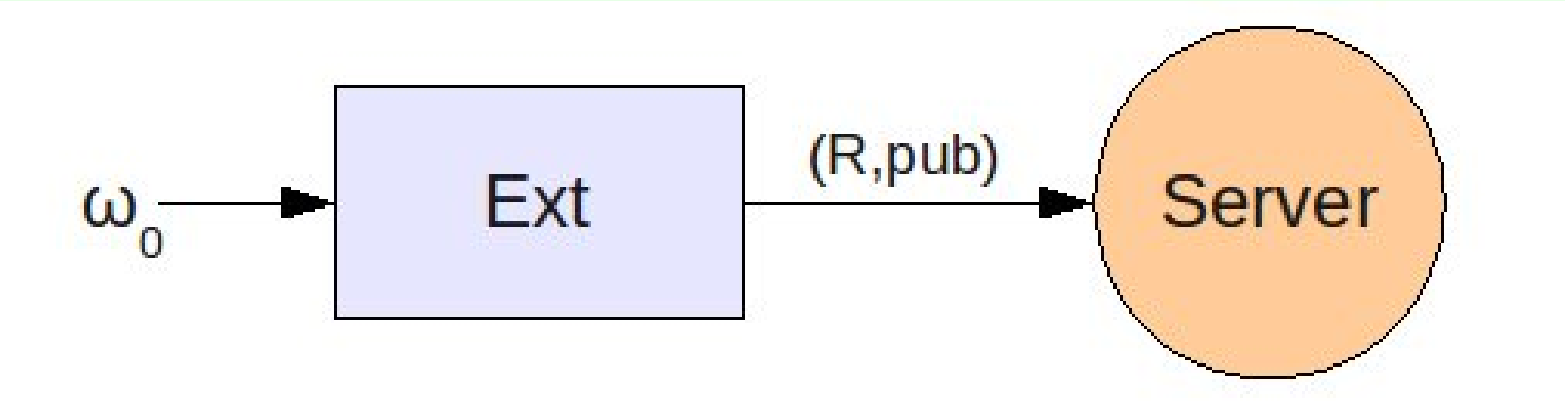
Construction of Robust Fuzzy Extractor

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

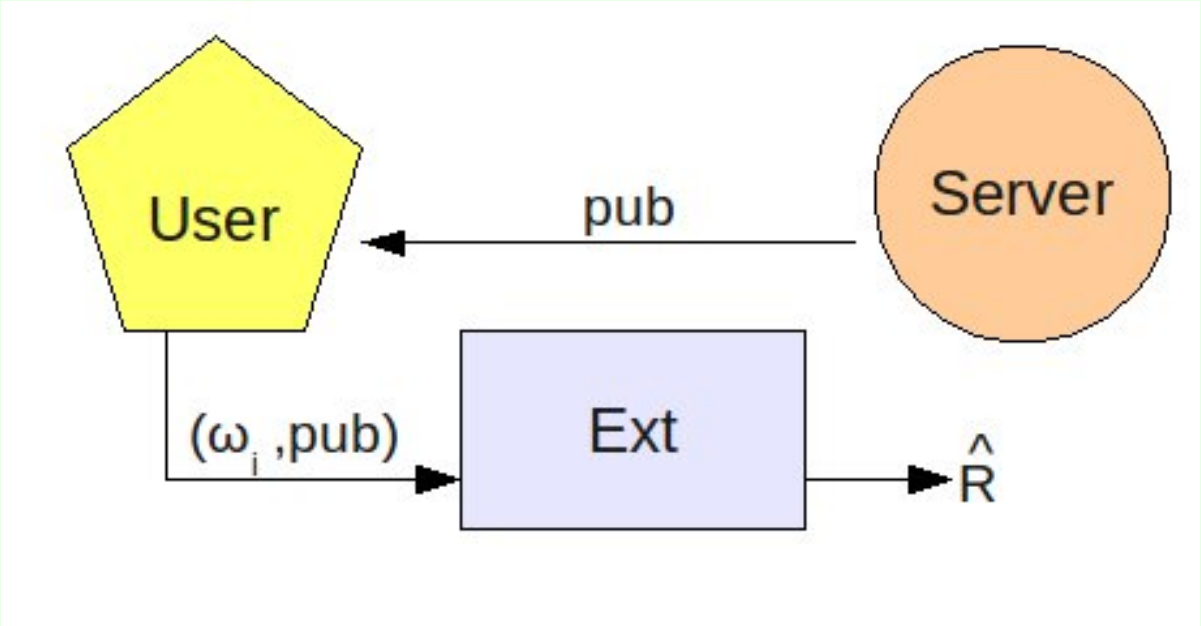
For any secure protocol Π based on a uniformly distributed key, any robust fuzzy extractor $(\mathbf{Ext}, \mathbf{Rec})$ and any source W_0, the protocol Π' is constructed as follows:

Initialization User scans W_0 to retrieve ω_0 and computes $\mathbf{Ext}(\omega_0) = (R, \mathrm{pub})$ and registers this at the server.



Execution

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



Correctness

If the user obtains correct pub from server, then user and server will use same R in procedure Π, because then $d(\omega_0, \omega_i) \leq t$.

Security

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 $\mathrm{pub}' \neq \mathrm{pub}$. These instances will then abort immediately except for a probability of at most ϵ.
- If the adversary forwards pub unchanged, the user and the server run Π. Even if the adversary succeeds in forwarding a changed pub', the protocol Π is assumed to be save and thus the adversary will not 'break in'

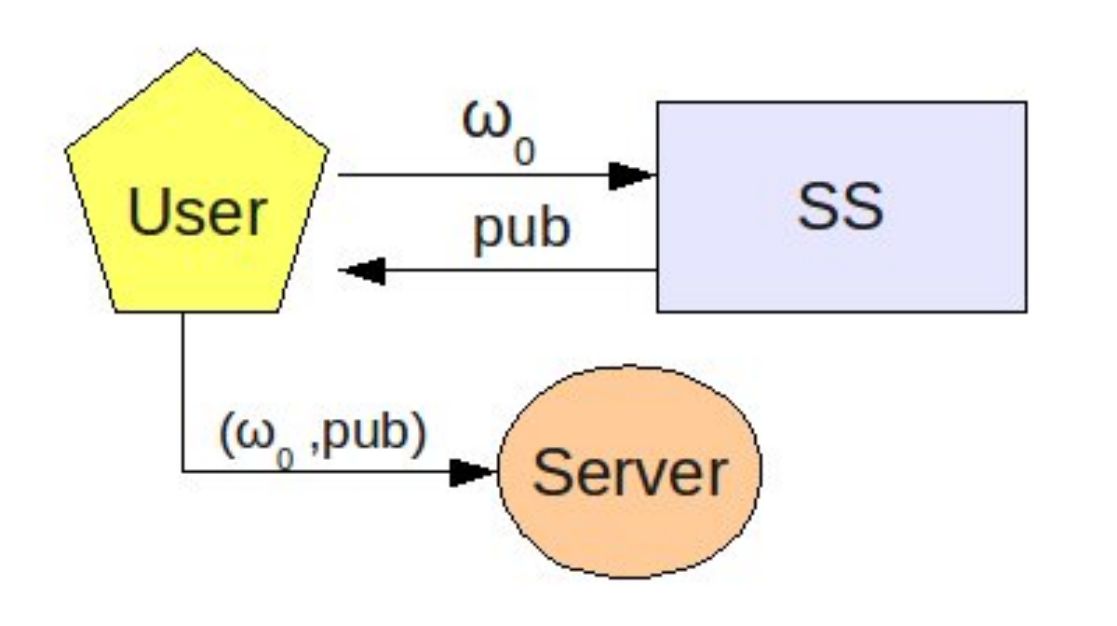
Improved Solution

Advantages of Improved Solution

- provably secure in the standard model
- improved bounds on the "effective entropy loss", as there is no randomness extraction

Initialization

Let Π be a password-only authenticated key exchange protocol and $(\mathbf{SS}, \mathbf{Rec})$ be a well-formed sketch. Construct Π' as follows: User U scans ω_0 from W_0 and computes $\mathbf{SS}(\omega_0) = \mathrm{pub}$. U registers (ω_0, pub) at server S.



Protocol Execution Server

pub is sent to user. Then the server executes Π using the following:

- identity: $S \parallel \mathrm{pub}$
- partner identity: $\mathrm{pid}_S = U \parallel \mathrm{pub}$.
- password: ω_0

Protocol Execution User

User retrieves ω_i and obtains $\mathbf{SS}(\omega_i) = \mathrm{pub}'$. With these he computes $\mathbf{Rec}(\omega_i, \mathrm{pub}') = \omega'$. If $\omega' = \perp$ he aborts, else he executes Π using the following:

- identity: $U \parallel \mathrm{pub}'$
- partner identity: $\mathrm{pid}_U = S \parallel \mathrm{pub}'$
- password: ω'

Correctness

With no interference from adversary we have:

1. $\mathrm{pid}_S = U \parallel \mathrm{pub}$
2. $\mathrm{pid}_U = S \parallel \mathrm{pub}$
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