A Privacy-preserving Biometric Authentication Protocol Revisited

Aysajan Abidin and Katerina Mitrokotsa

Department of Computer Science & Engineering
Chalmers University of Technology

YACC 2014,
Porquerolles, France, 9-13 June 2014

CHALMERS
UNIVERSITY OF TECHNOLOGY

Outline

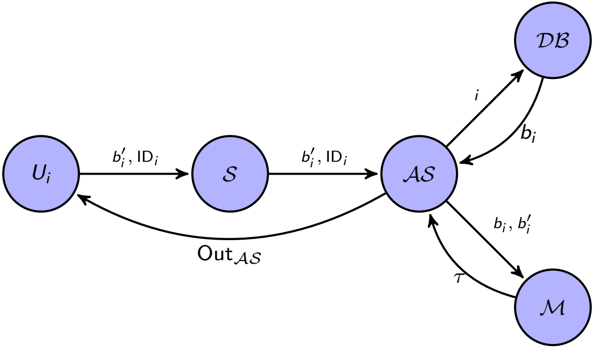
- Biometric authentication ⇒ **Privacy issues**
- Bringer *et al.** privacy-preserving biometric authentication protocol
- Secure & privacy-preserving in the *honest-but-curious* (or passive) attack model
- Algorithm: to mount a number of attacks on the protocol
- Propose *improved version* that is secure in the *malicious* (insider) but *non-colluding* insiders

* Bringer et al. "An application of the Goldwasser-Micali cryptosystem to biometric authentication", ACISP 2007.

Biometric-based cryptosystems

- Natural variability $\Rightarrow$ Biometric feature is rarely the same twice
- Traditional cryptographic handling (*e.g.* hash) not suitable
- Authentication: comparison of two vectors (binary for iris biometrics)
- Distributed architecture: lower the *level of trust* on the involved parties
- Example: relationship between a biometric feature & relevant identities
- Privacy issues $\Rightarrow$ Use secure multi-party computation

A distributed biometric authentication protocol



Schematic description of a distributed biometric authentication system

Threats

- *Biometric reference privacy:* $\mathcal{A}$ tries to recover the reference biometric template b_i.
- *Biometric sample privacy:* $\mathcal{A}$ tries to recover the fresh biometric template b'_i.
- *Identity privacy:* $\mathcal{A}$ tries to link a biometric template b_i to the user identity ID_i.
- *User indistinguishability:* $\mathcal{A}$ should not be able to *distinguish* if two authentication attempts are from the same user.

Definition (Privacy-preserving biometric authentication protocol)

A biometric authentication protocol is said to be **privacy-preserving** if no probabilistic polynomial-time (PPT) adversary can recover any of the following information, if they are not already known: a fresh biometric trait b'_i, a stored biometric template b_i and/or the correspondence between the identity ID_i and the stored template b_i.

GM Cryprosystem (KeyGen, Enc, Dec)

- **KeyGen**(1^ℓ): Upon an input 1^ℓ, where ℓ is the security parameter, outputs two distinct large prime numbers p and q, $n = pq$ and a non-residue x for which the Jacobi symbol is 1. The public key pk is (x, n), and the secret key sk is (p, q).
- **Enc**$(m, \mathbf{pk})$: Takes a message $m \in \{0, 1\}$ and the public key pk $= (x, n)$ as input, and outputs the ciphertext $c = y^2 x^m \mod n$, where y is randomly chosen from $Z_n^\star$.
- **Dec**$(c, \mathbf{sk})$: Takes a ciphertext c and the private key sk $= (p, q)$ as input, and outputs the message m, which is 0 if c is a quadratic residue, 1 otherwise.

It holds that:

- $\text{Enc}(m) \times \text{Enc}(m') = \text{Enc}(m \oplus m')$ and,
- $\text{Enc}(m)^y = \text{Enc}(ym)$

Bringer *et al.* privacy-preserving biometric authentication protocol

PHASE 1

Sensor $\mathcal{S}$		Authentication Server $\mathcal{AS}$
public key K_p		public key K_p
get b_i' from $\mathcal{U}_i$	$\xrightarrow{\text{Enc}(b_i'),\text{ID}_i}$	

PHASE 2

Authentication Server $\mathcal{AS}$		Database $\mathcal{DB}$
public key K_p		public key K_p
retrieves i from ID_i		
$t_j := \begin{cases} 1, & \textit{if } j = i \\ 0, & \textit{if } j \neq i \end{cases}$		
	for $j = 1$ to N $\xrightarrow{\text{Enc}(t_j)}$	for $k = 1$ to M
		$\prod_{j=1}^{N} \text{Enc}(t_j)^{b_{j,k}} = \text{Enc}(b_{i,k})$
	for $k = 1$ to M $\xleftarrow{\text{Enc}(b_{i,k})}$	

Bringer *et al.* biometric authentication protocol

PHASE 3

Authentication Server $\mathcal{AS}$		**Matcher $\mathcal{M}$**
public key K_p		public key K_p
Compute $\mathrm{Enc}(b'_{i,k})\mathrm{Enc}(b_{i,k})$ $= \mathrm{Enc}(b'_{i,k} \oplus b_{i,k}) = v_k$		
Take a permutation σ		
$\lambda_k = v_{\sigma(k)}$	$\xrightarrow{\lambda_1,\ldots,\lambda_M}$	$\left(\mathrm{Dec}(\lambda_1), \ldots, \mathrm{Dec}(\lambda_M)\right)$ for $k = 1$ to M Check if
	$\xleftarrow{\mathrm{HW}\left(\mathrm{Dec}(\lambda_k)\right) \leqslant \tau}$	$\mathrm{HW}\left(\mathrm{Dec}(\lambda_k)\right) \leqslant \tau$

PHASE 4

User $\mathcal{U}_i$		**Authentication Server $\mathcal{AS}$**
	$\xleftarrow{\mathrm{Out}_{\mathcal{AS}}}$	

The attack idea

- $\mathcal{AS}$ sets $\lambda := \mathsf{Enc}(b_i) = (\mathsf{Enc}(b_{i1}), \mathsf{Enc}(b_{i2}), \cdots, \mathsf{Enc}(b_{iM})) = c_1, c_2, \ldots, c_M$
- and b=$\underbrace{(\mathsf{Enc}(0), \cdots, \mathsf{Enc}(0))}_{M \ bits}$
- Replaces components of b with components of λ
- Sends **repeatedly** b to the $\mathcal{M}$ (check if it is accepted or rejected)
- Finds a bit-string whose Hamming weight is **equal** to the threshold $\tau + 1$

The attack idea

The matcher $\mathcal{M}$ is checking if $\mathrm{HW}(b_i \oplus b_i') \leqslant \tau$

The attack idea

$$\textbf{Step 1: } \overbrace{\mathsf{Enc}(0), \mathsf{Enc}(0), \mathsf{Enc}(0), \mathsf{Enc}(0), \cdots, \mathsf{Enc}(0)}^{M \;\; bits}$$

$$\Downarrow$$

Always Accepted

The attack idea

Step 2: $c_1, \mathsf{Enc}(0), \mathsf{Enc}(0), \mathsf{Enc}(0), \cdots, \mathsf{Enc}(0)$

$$\Downarrow$$

Accepted

The attack idea

Step 3: $c_1, c_2, \mathsf{Enc}(0), \mathsf{Enc}(0), \cdots, \mathsf{Enc}(0)$

$\Downarrow$

Accepted

The attack idea

Step 4: $c_1, c_2, c_3, \ldots, c_k, \mathrm{Enc}(0) \cdots, \mathrm{Enc}(0)$

$\Downarrow$

Rejected? then $b_k = 1$

The attack idea

Step 5: $c_1, c_2, c_3, \ldots, c_{k-1}, \color{blue}{Enc(1)}, \mathrm{Enc}(0), \cdots, \mathrm{Enc}(0)$

$\Downarrow$

k-th bit revealed!

The attack idea

$$\textbf{Step 6: } \text{Enc}(0), c_2, c_3, \ldots, c_{k-1}, Enc(1), \cdots, \text{Enc}(0)$$
$$\Downarrow$$
$$\text{Accepted? then } b_1 = 1$$

The attack idea

$$\textbf{Step 7:}\ \mathrm{Enc}(1), c_2, c_3, \ldots, c_{k-1}, Enc(1), \cdots, \mathrm{Enc}(0),$$

$$\Downarrow$$

start recovering bits 1 to k-1

Algorithm 1

Input: $\mathsf{Enc}(b_i) = c_1, \cdots, c_M$
Output: b_i
Initialise: $b_i = 00\cdots0$
For $k = 1$ to M:
    Set $\lambda = c_1, \ldots, c_k, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0)$
    Send λ to the matcher $\mathcal{M}$
    If λ is rejected **Then**
        break
    EndIf
    If $k == M$ **Then**
        Return centerSearch(b_i)
    EndIf
EndFor
Set $k^* = k$
Set $b_{i,k^*} = 1$
If $k^* \geq 2$ **Then**
    For $k = 1$ to $k^* - 1$:
        Set $\lambda = c_1, \ldots, c_{k-1}, \mathsf{Enc}(0), c_{k+1} \ldots, c_{k^*}, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0)$
            Send λ to the matcher $\mathcal{M}$
            If λ is accepted **Then**
                $b_{i,k} = 1$
            EndIf
    EndFor
EndIf
For $k = k^* + 1$ to M:
    Set $\lambda = c_1, \ldots, c_{k^*-1}, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0), c_k, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0)$
    Send λ to the matcher $\mathcal{M}$
    If λ is rejected **Then**
        $b_{i,k} = 1$
    EndIf
EndFor
Return b_i

Algorithm 1

Input: $\mathsf{Enc}(b_i) = c_1, \cdots, c_M$
Output: b_i
Initialise: $b_i = 00\cdots0$
For $k = 1$ to M:
Set $\lambda = c_1, \ldots, c_k, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0)$
Send λ to the matcher $\mathcal{M}$
If λ is rejected **Then**
break
EndIf
Send λ to the matcher $\mathcal{M}$
If λ is accepted **Then**
$b_{i,k} = 1$
EndIf
EndFor
EndIf
For $k = k^* + 1$ to M:
Set $\lambda = c_1, \ldots, c_{k^*-1}, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0), c_k, \mathsf{Enc}(0), \ldots, \mathsf{Enc}(0)$
Send λ to the matcher $\mathcal{M}$
If λ is rejected **Then**
$b_{i,k} = 1$
EndIf
EndFor
Return b_i

Attacks based on the attack algorithm

Attack 1 - $\mathcal{AS}$ Compromised

- Use Algorithm 1 and input $Enc(b_i) = c_1, \ldots, c_M$
- Deduce all bits of b_i, $O\big(\max(2(\tau + M), 4\tau + M)\big)$

Attack 2 - $\mathcal{AS}$ Compromised

- $\mathcal{AS}$ has access to $\text{Enc}(b_i)$ and $\text{Enc}(b_i \oplus b_i')$ thus deduce $\text{Enc}(b_i')$
- Use $\text{Enc}(b_i')$ as input to Algorithm 1, deduce b_i'

Attack 3 - Compromised $\mathcal{DB}$

- $\mathcal{DB}$ simulates $\mathcal{AS}$ queries $\mathcal{M}$
- Uses Algorithm 1 with input $\text{Enc}(t_j)$

Improved Protocol

PHASE 1

Sensor $\mathcal{S}$ — **Authentication Server** $\mathcal{AS}$

Get $\mathcal{M}$'s public key: pk
Secret key: K
Shared keys: K_1, K_2, $K_{\mathcal{S}\leftrightarrow\mathcal{DB}}$
Derive from $K_{\mathcal{S}\leftrightarrow\mathcal{DB}}$: π
Generate: S
$\omega = \mathrm{Enc}_{K_1}(S)$
$\sigma = h_{K_1}(\omega)$
Get b'_i and ID_i from $\mathcal{U}_i$
$\mathrm{id}_i = \mathrm{Enc}_K(\mathrm{ID}_i)$
Compute, for $k = 1, \cdots, M$:

$$a_k = \mathrm{Enc}\big((b'_{i,k})_\pi \oplus S_k)\big) \xrightarrow{a,\ \mathrm{id}_i,\ (\omega,\sigma)}$$

Phase 2

Authentication Server $\mathcal{AS}$

Get $\mathcal{M}$'s public key: pk
Shared key: K_3
Retrieve i from id_i

$$t_j := \begin{cases} 1, & \text{if } j = i \\ 0, & \text{if } j \neq i \end{cases}$$

Compute, for $j = 1, \cdots, N$: $d_j = \text{Enc}(t_j)$

$\xrightarrow{d}$

Database $\mathcal{DB}$

Get $\mathcal{M}$'s public key: pk
Shared keys: K_4, K_5, $K_{\mathcal{S} \leftrightarrow \mathcal{DB}}$

Derive from $K_{\mathcal{S} \leftrightarrow \mathcal{DB}}$: π

Generate: S', K_4'
Compute, for $k = 1, \cdots, M$:

$$\left(\prod_{j=1}^{N} d_j^{(b_{j,k})_\pi \oplus S_k'}\right) = \text{Enc}\Big((b_{i,k})_\pi \oplus S_k'\Big) = c_k$$

$$\omega' = \text{Enc}_{K_4}(S')$$

$\xleftarrow{c, (\omega', \sigma')}$

$$\sigma' = h_{K_5}(\omega')$$

Phase 3

Authentication Server $\mathcal{AS}$ | **Matcher $\mathcal{M}$**

For $k = 1, \cdots, M$:

$$a_k c_k = \mathrm{Enc}\Big(\big(b'_{i,k} \oplus b_{i,k}\big)_\pi \oplus S_k \oplus S'_k \Big) = \lambda_k$$

$$\sigma'' = h_{K_3}(\lambda)$$

$\xrightarrow{(\omega,\sigma),\,(\omega',\sigma'),\,(\lambda,\sigma'')}$

Matcher $\mathcal{M}$:

$K_1,\ K_2,\ K_3,\ K_4,\ K_5$

sk

Check:

$$h_{K_2}(\omega) \stackrel{?}{=} \sigma,\ h_{K_5}(\omega') \stackrel{?}{=} \sigma'$$
$$h_{K_3}(\lambda) \stackrel{?}{=} \sigma''$$
$$S \leftarrow \mathrm{Dec}_{K_1}(\omega)$$
$$S' \leftarrow \mathrm{Dec}_{K_4}(\omega')$$
$$(b_i \oplus b'_i)_\pi \leftarrow \mathrm{Dec}(\lambda) \oplus S \oplus S'$$

Check:

$$\mathrm{HW}\big((b_i \oplus b'_i)_\pi\big) \leqslant \tau$$

$\xleftarrow{\text{YES or NO}}$

Phase 4

User $\mathcal{U}_i$ | **Authentic. Server $\mathcal{AS}$**

$\xleftarrow{\mathrm{Out}_{\mathcal{AS}}}$

Identity Privacy

No link between the user's identity and hir/her biometrics.

Theorem

For any ID_{i_0} and two biometric templates b'_{i_0}, b'_{i_1}, where i_0, $i_1 \geqslant 1$ and b'_{i_0} is the biometric template related to ID_{i_0}, any of the malicious, but not colluding $\mathcal{AS}$, $\mathcal{DB}$, and $\mathcal{M}$ can only **distinguish** *between (ID_{i_0}, b'_{i_0}) and (ID_{i_0}, b'_{i_1}) with a* **negligible advantage**.

User Indistinguishability

The $\mathcal{DB}$ may not distinguish the authentication attempts of two users.

Theorem

For any two users $\mathcal{U}_{i_0}$ and $\mathcal{U}_{i_1}$, where $i_0, i_1 \geqslant 1$, if $\mathcal{U}_{i_\beta}$ where $\beta \in \{0,1\}$ makes an authentication attempt, then the malicious database $\mathcal{DB}$ can only guess β with a negligible advantage. The adversary's advantage is defined as $\left|\Pr\{\beta = \beta'\} - 1/2\right|$, where β' is $\mathcal{DB}$'s guess.

Assumptions

- The sensor $\mathcal{S}$ is *honest*, has not been compromised and captures the biometric template b_i from an alive human user.
- The entities $\mathcal{AS}$, $\mathcal{DB}$, $\mathcal{M}$ may *not collude* with each other.

Assumptions

- The sensor $\mathcal{S}$ is *honest*, has not been compromised and captures the biometric template b_i from an alive human user.
- The entities $\mathcal{AS}$, $\mathcal{DB}$, $\mathcal{M}$ may *not collude* with each other.

Theorem

If the Assumptions hold and if:
(a) S and S' are generated using ϵ-secure PNGs,
(b) the symmetric encryption schemes SKE used between the sensor $\mathcal{S}$ and the matcher $\mathcal{M}$, and between the database $\mathcal{DB}$ and the matcher $\mathcal{M}$, is IND-COA-secure, and
(c) the GM scheme is IND-CPA-secure.
Then, our modified protocol is secure any against malicious authentication server $\mathcal{AS}$.

Conclusions

- Enabler of the attack: **Bit-by-bit encryption** using GM encryption scheme.
- **Question:** How to avoid the attack when multiple entities are **colluding**?
- Use another way to compare fresh and stored biometric instead of HW?

Thank you for your attention!