Security Attack on CloudBI: Practical privacy-preserving outsourcing of biometric identification in the cloud

Jiawei Yuan, Embry-Riddle Aeronautical University

I. INTRODUCTION

In ESORICS 2015 [1], Wang et al. proposed a privacy-preserving outsourcing design for biometric identification using public cloud platforms, namely CloudBI. CloudBI introduces two designs: CloudBI-I and CloudBI-II. CloudBI-I is more efficient and CloudBI-II has stronger privacy protection. Based on the threat model of CloudBI, CloudBI-II is claimed to be secure even when the cloud service provider can act as a user to submit fingerprint information for identification. However, this security argument is not hold and CloudBI-II can be completely broken when the cloud service provider submit a small number of identification requests. In this technical report, we will review the design of CloudBI-II and introduce the security attack that can efficiently break it.

II. BRIEF REVIEW OF CloudBI-II

In the data encryption phase of CloudBI-II, each FingerCode $b_i = [b_{i1}, b_{i2}, \cdots, b_{in}]$ are extended as $B_i'$

$$B_i' = \begin{bmatrix}
    b_{i1} & 0 & \cdots & 0 & 0 & 0 \\
    0 & b_{i2} & \cdots & 0 & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
    0 & \cdots & 0 & b_{in} & 0 & 0 \\
    0 & \cdots & 0 & 0 & -0.5 \sum_{j=1}^{n} b_{ij} & 0 \\
    0 & \cdots & 0 & 0 & 0 & 1
\end{bmatrix}$$

Each $B_i'$ is encrypted as

$$C_i = M_1 Q_i B_i' M_2$$

where $M_1, M_2$ are two random $(n + 2) \times (n + 2)$ invertible matrices, and $Q_i$ is a random $(n + 2) \times (n + 2)$ lower triangular matrix with diagonal entries set as 1. All $C_i$ will be outsourced to cloud servers.

$$Q_i = \begin{bmatrix}
    1 & 0 & \cdots & 0 & 0 \\
    r_{i1} & 1 & \cdots & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    r_{i(n+1)} & \cdots & r_{i(n+1)n} & 1 & 0 \\
    r_{i(n+2)} & \cdots & r_{i(n+2)n} & r_{i(n+2)(n+1)} & 1
\end{bmatrix}$$

When the user submit a candidate FingerCode $b_c = [b_{c1}, b_{c2}, \cdots, b_{cn}]$ for identification, the biometric database owner extends it as $B_c'$

$$B_c' = \begin{bmatrix}
    b_{c1} & 0 & \cdots & 0 & 0 & 0 \\
    0 & b_{c2} & \cdots & 0 & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
    0 & \cdots & 0 & b_{cn} & 0 & 0 \\
    0 & \cdots & 0 & 0 & 1 & 0 \\
    0 & \cdots & 0 & 0 & 0 & r_c
\end{bmatrix}$$

where $r_c$ is a random number generated for each identification request. The owner then encrypts $B_c'$ as

$$C_F = M_2^{-1} B_c' Q_c M_1^{-1}$$

where $M_1^{-1}$ and $M_2^{-1}$ are inverse matrices of $M_1$ and $M_2$ respectively, $Q_c$ is a random $(n + 2) \times (n + 2)$ lower triangular matrix with diagonal entries set as 1. $C_F$ is finally submitted to cloud servers for identification.
We now show that the cloud server only needs to submit more than 3 identification requests to break the ciphertext $C_i$ of any FingerCode $b_i$ in the owner's database. For expression simplicity, we use $n'$ to denote $n+2$ in the rest part of this section.

After submitting an identification request, the cloud server has access to $C_i$ of any FingerCode $b_i$ and $C_F$ of the submitted FingerCode $b_c$. Then, the cloud server can compute

$$P_i = C_i C_F = M_1 Q_i B'_1 M_2 M_2^{-1} B'_c Q_c M_1^{-1} = M_1 Q_i B'_1 B'_c Q_c M_1^{-1}$$

We now use $P_1$ of FingerCode $b_1$ as an example to show our attack, which can also be applied to any other FingerCode $b_i$ in the same manner. In $P_1$, there are $n'^2$ unknowns in $M_1$, $n'$ unknowns in $B'_1$, $\frac{n'^2-n'}{2}$ unknowns in $Q_1$, $\frac{n'^2-n'}{2}$ unknowns in $Q_c$. As $b_c$ is submitted by the cloud server, there is only one unknown $r_c$ in $B'_c$. $M_1^{-1}$ can be expressed with elements in $M_1$ since it is the inverse matrix of $M_1$. Among these unknowns, $M_1$, $Q_1$, $B'_1$ are fixed for all identification requests, $B'_c$ and $Q_c$ are randomly generated for each identification request. Therefore, after the first identification request, each new identification request only introduces $\frac{n'^2-n'}{2}+1$ unknowns to the computation of $P_1$. However, as $M_1$, $Q_1$, $B'_1$, $B'_c$, $Q_c$, $M_1^{-1}$ are all $n' \times n'$ matrices, it is easy to see that the cloud server can construct $n'^2$ equations for $P_1$ from each new identification request. As shown in Table III, when the cloud server submits more than 3 identification requests, it can construct more equations than the number of unknowns in $P_1$. Thus, all unknowns in $P_1$ decrypted by solving their corresponding equations. Once unknowns in $B'_c$ are decrypted, the cloud can easily extract the actual FingerCode $b_1$. To decrypt any other FingerCode $b_i$, the cloud server just needs to perform the same attack as that for $b_1$.

To this end, we have demonstrated that CloudBI-II can be completely broken when the cloud server can submit more than 3 identification requests.

**TABLE I**

<table>
<thead>
<tr>
<th># of Requests</th>
<th># of Unknowns in $P_1$</th>
<th># of Equations from $P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2n'^2$</td>
<td>$n'^2$</td>
</tr>
<tr>
<td>2</td>
<td>$2n'^2 - \frac{n'^2}{2} + 1$</td>
<td>$2n'^2$</td>
</tr>
<tr>
<td>3</td>
<td>$3n'^2 - n' + 2$</td>
<td>$3n'^2$</td>
</tr>
<tr>
<td>4</td>
<td>$3n'^2 - \frac{n'^2}{2} + 3$</td>
<td>$4n'^2$</td>
</tr>
</tbody>
</table>

**IV. EXAMPLE OF SECURITY ATTACK ON CLOUDBI-II**

In this example, we set $n=2$ and $n'=n+2=4$. For $b_1 = [2, 2]$, the owner extends it as

$$B'_1 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The owner randomly generates $M_1, M_2, Q_1$ as

$$M_1 = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 3 & 4 & 1 & 1 \\ 1 & 3 & 3 & 0 \\ 1 & 4 & 2 & 2 \\ 2 & 2 & 0 & 1 \end{bmatrix}, \quad Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 4 & 1 & 0 & 1 \end{bmatrix}$$

$B'_1$ is encrypted as $C_1 = M_1 Q_1 B'_1 M_2$ and outsourced to cloud servers. Now the cloud server selects $b_c = (1, 3)$ for identification and submits it 3 times. We denote the extended $B'_c$ for 3 identification requests as $B'_{c1}, B'_{c2}, B'_{c3}$ respectively.

$$B'_{c1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}, \quad B'_{c2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}, \quad B'_{c3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$$

The owner encrypts $B'_{c1}, B'_{c2}$ and $B'_{c3}$ as $C_{F1} = M_2^{-1} B'_{c1} Q_{c1} M_1^{-1}, C_{F2} = M_2^{-1} B'_{c2} Q_{c2} M_1^{-1}$ and $C_{F3} = M_2^{-1} B'_{c3} Q_{c3} M_1^{-1}$ respectively, where $Q_{c1}, Q_{c2}$ and $Q_{c3}$ are randomly generated as

$$Q_{c1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 8 & 11 & 2 & 1 \end{bmatrix}, \quad Q_{c2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 5 & 12 & 1 & 0 \\ 2 & 8 & 3 & 1 \end{bmatrix}, \quad Q_{c3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 12 & 1 & 0 & 0 \\ 10 & 0 & 1 & 0 \\ 3 & 2 & 1 & 1 \end{bmatrix}$$
After $C_{F1}$, $C_{F2}$ and $C_{F3}$ are sent to the cloud, the cloud computes

\[ P_{11}M_1 = C_1C_{F1}M_1 = M_1Q_1B_1^tB_{c1}^tQ_{c1}M_1^{-1}M_1 = M_1Q_1B_1^tB_{c1}^tQ_{c1} \]  
\[ P_{12}M_1 = C_1C_{F2}M_1 = M_1Q_1B_1^tB_{c2}^tQ_{c2}M_1^{-1}M_1 = M_1Q_1B_1^tB_{c2}^tQ_{c2} \]  
\[ P_{13}M_1 = C_1C_{F3}M_1 = M_1Q_1B_1^tB_{c3}^tQ_{c3}M_1^{-1}M_1 = M_1Q_1B_1^tB_{c3}^tQ_{c3} \]

Based on Eq. 1-3, the cloud can construct the following equations to solve all unknowns in $M_1$, $Q_1$, $B_1^t$, $B_{c1}^t$, $B_{c2}^t$, $B_{c3}^t$, $Q_{c1}$, $Q_{c2}$ and $Q_{c3}$.

Based on above matrix multiplications, it is clear that the cloud server can construct 16 equations for $P_{11}M_1$, 16 equations for $P_{12}M_1$, and 16 equations for $P_{13}M_1$. Meanwhile, there are 46 total unknowns in $P_{11}M_1$, $P_{12}M_1$ and $P_{13}M_1$. Thus, when the cloud server submit 3 identification requests, it will have sufficient information to solve all unknowns in $M_1$, $Q_1$, $B_1^t$, $B_{c1}^t$, $B_{c2}^t$, $B_{c3}^t$, $Q_{c1}$, $Q_{c2}$ and $Q_{c3}$. Once the cloud server gets $B_1^t$, it can easily recover $b_1 = [2, 2]$.

REFERENCES