Supplementary Material of Security Analysis on Locality-Sensitive Hashing-Based Biometric Template Protection Schemes

Seunghun Paik
whitesoonguh@hanyang.ac.kr
Sunpill Kim
ksp0352@hanyang.ac.kr
Jae Hong Seo*
jaehongseo@hanyang.ac.kr

Department of Mathematics
Research Institute for Natural Sciences
Hanyang University
Seoul, Republic of Korea

In this supplementary material, we provide the followings:

- Formal definitions related to BTP and LSH.
- Formal descriptions of the target LSHs (IoM-type LSHs [6] and ABH [7]).
- Detailed reformulation processes into LSPs on each LSH.
- Detailed experimental settings.

Notation. Throughout this supplementary material, we denote a vector in $\mathbb{R}^n$ as a boldface lowercase letter $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, and the inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ as $\mathbf{x} \cdot \mathbf{y}$. We denote a matrix as a boldface uppercase letter, *e.g.*, $\mathbf{M}$. For a set of all integers between 1 and n, we use the notation $[n]$. An algorithm is denoted by $H(\mathbf{x}; \mathbf{r})$, where $\mathbf{x}$ is the input, and $\mathbf{r}$ is an additional parameter such as random source for probabilistic algorithms. $\mathcal{N}(\mu, \mathbf{C})$ denotes a Gaussian distribution with a mean μ and a covariance matrix $\mathbf{C}$. For a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ denotes that x is uniformly sampled from $\mathcal{S}$. A metric space is denoted by a pair $(\mathcal{X}, d_{\mathcal{X}})$ of a set $\mathcal{X}$ and the corresponding metric $d_{\mathcal{X}}$ defined on $\mathcal{X}$. We denote $sgn : \mathbb{R} \to \{-1, 1\}$ as a function that returns a sign of the given real-valued number. We shall define $sgn(0) = 1$ for the sake of correctness.

* Corresponding author.

A Formal Definitions

In this section, we provide the formal definitions omitted in the main paper.

A.1 Biometric Template Protection

We first give a formal definition of biometric authentication systems. The biometric authentication system is defined as a pair of algorithms (Ext, V), each of which stands for feature extraction algorithm and verification algorithm, respectively. The formal description is given below.

Definition 1 (Biometric Authentication System)**.** *Let $\mathcal{I}$ be a set of biometrics and $(\mathcal{X}, d_{\mathcal{X}})$ a metric space. A biometric authentication system defined over $(\mathcal{I}, (\mathcal{X}, d_{\mathcal{X}}))$ with a threshold parameter τ is a pair of deterministic algorithms (Ext, V), where*

- *Ext is the extraction algorithm mapping from $\mathcal{I}$ to $\mathcal{X}$.*
- *V is the verification algorithm that takes a pair of vectors $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ and outputs 1 if $d_{\mathcal{X}}(\mathbf{x}, \mathbf{y}) \leq \tau$, and 0 otherwise.*

Note that the definition of a biometric authentication system does not capture the privacy threats of the biometric information. The following scheme named Biometric Template Protection (BTP) is an extension of a biometric authentication system, which has an additional process called *transformation* to protect the biometric templates generated by Ext. The BTP consists of two deterministic algorithms (Ext, V) and a probabilistic algorithm T satisfying the following definition.

Definition 2 (Biometric Template Protection)**.** *A biometric template protection scheme over $(\mathcal{I}, (\mathcal{X}, d_{\mathcal{X}}), \mathcal{R}, (\mathcal{S}, d_{\mathcal{S}}))$ is a triple of algorithms (Ext, V, T), where Ext and V are defined as in Definition 1 and T is a transformation algorithm defined as follows:*

- *T takes $\mathbf{x} \in \mathcal{X}$ (transformation object) and $R \in \mathcal{R}$ (randomness) as inputs and outputs an element in the metric space $(\mathcal{S}, d_{\mathcal{S}})$.*

Recall that in the main text, we expected additional properties to be satisfied: First, T should be difficult to invert, even when randomness R is disclosed to the public. Second, the transformation T should almost preserve the distance for each randomness R such that close vectors map to close vectors. The first property enables us to use a pair $(T(Ext(img); R), R)$ as a new (protected) template for $img \in \mathcal{I}$ instead of $Ext(img)$. The second property enables the closeness between original templates to be tested by testing the (protected) templates because, for a fixed R, the closeness between $(T(Ext(img); R), R)$ and $(T(Ext(\overline{img}); R), R)$ is equivalent to the closeness between $Ext(\overline{img})$ and $Ext(img)$[1].

[1] For a successful test, a public R is necessary. For this reason, we expect the onewayness of T even when R is disclosed to the public.

A.2 Locality-Sensitive Hashing

We now give a detailed explanation of locality-sensitive hashing (LSH) including its formal definition. LSH, first discussed in [4], is a collection of "similarity preserving" functions. The formal definition of LSH is as follows:

Definition 3 (Locality Sensitive Hashing). *Let $\mathcal{H}$ be a collection of functions from a metric space $(\mathcal{X}, d_X)$ to a finite set $\mathcal{S}$. We assume that $\mathcal{H}$ is $(\tau_1, \tau_2, p_1, p_2)$-sensitive if it satisfies the following two properties: For any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$,*

$$(1) \text{ if } d_X(\mathbf{x}, \mathbf{y}) < \tau_1, \text{ then } Pr[h(\mathbf{x}) = h(\mathbf{y})] > p_1$$
$$(2) \text{ if } d_X(\mathbf{x}, \mathbf{y}) > \tau_2, \text{ then } Pr[h(\mathbf{x}) = h(\mathbf{y})] < p_2.$$

The probability space is determined by the choice of $h \in \mathcal{H}$.

The $(\tau_1, \tau_2, p_1, p_2)$-sensitive collection $\mathcal{H}$ can be naturally extended to the $(\tau_1, \tau_2, p_1^n, p_2^n)$-sensitive collection $\mathcal{G}$, including $g(\mathbf{x})$ defined as

$$g(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_n(\mathbf{x})),$$

where $h_1, h_2, \cdots, h_n$ are distinct functions independently chosen from $\mathcal{H}$.

From this, we define a metric on $\mathcal{S}^n$ as the Hamming distance $d_{\mathcal{S}}$[2]. If we assume that $h_1, h_2, \cdots, h_n$ are independently sampled, then the distance between the two outputs is closely related to τ_1 and τ_2. That is, we can consider g as a similarity-preserving map because it can distinguish whether the two given vectors are sufficiently close or not. By the convention of the literature on LSH-based BTP proposals, we denote the collection $\mathcal{G}$ as LSH rather than that appearing in the original definition. We note that the random sampling of $g \in \mathcal{G}$ in the definition of LSH can be achieved by introducing an additional random parameter on g.

B Formal Descriptions of Previous LSHs

We give concrete descriptions of the target LSHs mentioned in the main text: GRP-IoM, URP-IoM [6], and ABH [7]. Prior to the explanation, we first introduce the first LSH-based BTP called BioHashing [5]. BioHashing consists of two algorithms, *Setup* and *Hashing*. Given two positive integers, d and m, as system parameters, BioHashing is defined as follows.

- *Setup*, denoted by S_{BH}, is a probabilistic algorithm that outputs a $m \times d$ matrix $\mathbf{R}$. Row vectors $\{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m\}$ of $\mathbf{R}$ are independently sampled from $\mathcal{N}(\vec{0}, I_d)$.

- *Hashing*, denoted by H_{BH}, is a deterministic algorithm that takes a real-valued vector $\mathbf{x} \in \mathbb{R}^d$ and internal parameter $\mathbf{R}$ generated by *Setup*. This algorithm outputs an integer vector $\alpha = (\alpha_1, \cdots, \alpha_n) \in [m]^n$, where $\alpha_i = sgn(\mathbf{r}_i \cdot \mathbf{x})$.

[2] The distance between two given strings of the same length is defined as the number of different entries for each position.

B.1 GRP-IoM

GRP-IoM consists of two algorithms, *Setup* and *Hashing*. Given three positive integers, d, m and n, as system parameters, GRP-IoM is defined as follows.

- *Setup*, denoted by S_{GRP}, is a probabilistic algorithm that outputs a set of matrices $\mathbf{R} = \left\{\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(n)}\right\}$ such that $\mathbf{R}^{(i)} \in \mathbb{R}^{m \times d}$ for $i \in [n]$. In addition, row vectors $\left\{\mathbf{r}_1^{(i)}, \mathbf{r}_2^{(i)}, \cdots, \mathbf{r}_m^{(i)}\right\}$ of $\mathbf{R}^{(i)}$ are independently sampled from $\mathcal{N}(\vec{0}, I_d)$.
- *Hashing*, denoted by H_{GRP}, is a deterministic algorithm that takes a real-valued vector $\mathbf{x} \in \mathbb{R}^d$ and internal parameter $\mathbf{R}$ generated by *Setup*. This algorithm outputs an integer vector $\alpha = (\alpha_1, \cdots, \alpha_n) \in [m]^n$, where $\alpha_i = \mathrm{argmax}_{k \in [m]} \mathbf{r}_k^{(i)} \cdot \mathbf{x}$

The formal description of H_{GRP} is provided in Algorithm 1.

Algorithm 1 GRP-IoM.H_{GRP}

Require: $\mathbf{x} \in \mathbb{R}^d, n \in \mathbb{N}$, and $m \in \mathbb{N}$
1: **for** $i \in [n]$ **do**
2: Set $\mathbf{r}_j^{(i)} \leftarrow \mathcal{N}(\vec{0}, I_d)$ for $j \in [m]$
3: Set $\alpha_i \leftarrow \mathrm{argmax}_{j \in [m]} \mathbf{r}_j^{(i)} \cdot \mathbf{x}$ for $j \in [m]$
4: Set $\mathbf{R}^{(i)} \leftarrow \left[\mathbf{r}_1^{(i)} \| \cdots \| \mathbf{r}_m^{(i)}\right]^T$
5: **end for**
6: Set $\mathbf{R} \leftarrow \left\{\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \cdots, \mathbf{R}^{(n)}\right\}$
7: Set $\alpha \leftarrow (\alpha_1, \cdots, \alpha_n)$
8: **return** $(\alpha, \mathbf{R})$

B.2 URP-IoM

Another IoM-type LSH called URP-IoM takes four positive integers, d, n, p, and w, as system parameters, where d must be greater than w.

- *Setup*, denoted by S_{URP}, is a probabilistic algorithm that outputs a set of permutations

$$\Sigma = \left\{\sigma^{(i,j)} : i \in [n], j \in [p]\right\}$$

  from $[d]$ to itself such that each element is uniformly and independently sampled from the set of all permutations from $[d]$ to itself.
- *Hashing*, denoted by H_{URP}, is a deterministic algorithm that takes a real-valued vector $\mathbf{x} \in \mathbb{R}^d$ and internal parameter Σ generated by *Setup*. This algorithm outputs an integer vector $\alpha = (\alpha_1, \cdots, \alpha_n) \in [w]^n$, where $\alpha_i = \mathrm{argmax}_{k \in [w]} \left(\prod_{j=1}^{p} \sigma^{(i,j)}(\mathbf{x})\right)_k$.

The formal description of H_{URP} is provided in Algorithm 2.

Algorithm 2 URP-IoM.H_{URP}

Require: $\mathbf{x} \in \mathbb{R}^d, n \in \mathbb{N}, w \in \mathbb{N}$, and $p \in \mathbb{N}$
1: Set $\Sigma \leftarrow \{\sigma : [d] \to [d] | \sigma \text{ is bijective}\}$
2: **for** $i \in [n]$ **do**
3: Set $\sigma^{(i,j)} \xleftarrow{\$} \Sigma$ for $j \in [p]$
4: Set $\alpha_i \leftarrow \mathrm{argmax}_{k \in [w]} \left(\prod_{j=1}^{p} \sigma^{(i,j)}(\mathbf{x}) \right)_k$
5: Set $\mathbf{R}^{(i)} \leftarrow \left\{ \sigma^{(i,1)}, \cdots, \sigma^{(i,p)} \right\}$
6: **end for**
7: Set $\mathbf{R} \leftarrow \left\{ \mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \cdots, \mathbf{R}^{(n)} \right\}$
8: Set $\alpha \leftarrow (\alpha_1, \cdots, \alpha_n)$
9: **return** $(\alpha, \mathbf{R})$

B.3 ABH

Recently, another type of LSH was proposed in [2] that is different from IoM. Remark that we call it Advanced Biohashing (ABH) throughout the main paper because this scheme is not named by the authors, though its form originates from and revises the weakness of Biohashing [5]. ABH takes four positive integers, d, p, q, and r, as system parameters.

- *Setup*, denoted by S_{ABH}, is a probabilistic algorithm that outputs a set of matrices $\mathbf{R} = \left\{ \mathbf{R}^{(1)}, \cdots, \mathbf{R}^{(pq)} \right\}$ such that $\mathbf{R}^{(i)} \in \mathbb{R}^{r \times d}$ for $i \in [pq]$. The row vectors $\left\{ \mathbf{r}_1^{(i)}, \cdots, \mathbf{r}_r^{(i)} \right\}$ of $\mathbf{R}^{(i)}$ are independently sampled from $\mathcal{N}(\vec{0}, I_d)$.
- *Hashing*, denoted by H_{ABH}, is a deterministic algorithm that takes a real-valued vector $\mathbf{x} \in \mathbb{R}^d$ and internal parameter $\mathbf{R}$ generated by *Setup*. This algorithm outputs a matrix $\mathrm{A} = (\hat{\alpha}_{jk}) \in [2^r]^{p \times q}$ from the following process.
  1. Set $\alpha_i = (\alpha_{i1}, \cdots, \alpha_{ir})$ such that $\alpha_{il} = sgn(\mathbf{r}_l^{(i)} \cdot \mathbf{x})$ for $i \in [pq]$.
  2. Set α_i' to a decimal expression of $(\alpha_{i1}\alpha_{i2}\cdots\alpha_{ir})_2$.
  3. Set $j = i \pmod p + 1$ and k such that $i = jp + k$.
  4. Set $\hat{\alpha}_{jk}$ using α_i' for $j \in [p]$ and $k \in [q]$.

The formal description of H_{ABH} is provided in Algorithm 3.

Algorithm 3 ABH.H_{ABH}

Require: $\mathbf{x} \in \mathbb{R}^d, p \in \mathbb{N}, q \in \mathbb{N}$, and $r \in \mathbb{N}$

Set $n \leftarrow p \times q \times r$
for $i \in [pq]$ **do**
  for $j \in [r]$ **do**
    Set $\mathbf{r}_j^{(i)} \leftarrow \mathcal{N}(\vec{0}, I_d)$
    Set $\alpha_{ij} \leftarrow sgn(\mathbf{r}_j^{(i)} \cdot \mathbf{x})$
  end for
  Set $\mathbf{R}^{(i)} \leftarrow \left[\mathbf{r}_1^{(i)} || \cdots || \mathbf{r}_r^{(i)}\right]^T$
end for
Set $\mathbf{R} \leftarrow \left\{\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \cdots, \mathbf{R}^{(pq)}\right\}$
Set α_i' as a decimal expression of $(\alpha_{i1}\alpha_{i2}\cdots,\alpha_{ir})_2$.
Set $j \leftarrow i \pmod{p} + 1$ and $k \leftarrow i - jp$
Set $\hat{\alpha}_{jk} \leftarrow \alpha_i'$
Set $\mathrm{A} \leftarrow (\hat{\alpha}_{ij}) \in [2^r]^{p \times q}$.
return $(\mathrm{A}, \mathbf{R})$

C Reformulation of LSH into LSP

Now we demonstrate that GRP-IoM, URP-IoM, and ABH can be converted to our LSP notation. We first give an example of how to convert the BioHashing into our LSPs. For a $m \times d$ matrix $\mathbf{R}$ whose row vectors are $\mathbf{r}_i$, $H_{\mathrm{BH}}(\mathbf{x};\mathbf{R}) = (sgn(\mathbf{r}_i \cdot \mathbf{x}))_{i\in[m]}$. If the ith component of the hashed value is a_i, then we have that $a_i = sgn(\mathbf{r}_i \cdot \mathbf{x})$. Thus, for $\mathbb{P}_i^{\mathrm{BH}}(\mathbf{x};\mathbf{R}) = (-1)^{a_i} sgn(\mathbf{r}_i \cdot \mathbf{x})$ for $i \in [m]$, we obtain $H_{\mathrm{BH}}(\mathbf{x};\mathbf{R}) = (a_1, \ldots, a_m)$ if and only if $\mathbb{P}_i^{\mathrm{BH}}(\mathbf{x};\mathbf{R}) \leq 0$ for all $i \in [m]$.

C.1 GRP-IoM

For a given $\mathbf{x} \in \mathbb{R}^d$, let $H_{\mathrm{GRP}}(\mathbf{x};\mathbf{R}) = (h_{\mathrm{GRP}}(\mathbf{x};\mathbf{R}^{(1)}), \cdots, h_{\mathrm{GRP}}(\mathbf{x};\mathbf{R}^{(n)}))$. Then, from the definition of argmax, we obtain an inequality for $a \in [m]$.

$$h_{\mathrm{GRP}}(\mathbf{x};\mathbf{R}^{(i)}) = a \Leftrightarrow (\mathbf{r}_j^{(i)} \cdot \mathbf{x} - \mathbf{r}_a^{(i)} \cdot \mathbf{x}) \leq 0, \forall j \in [m].$$

Let us define $\mathbb{P}_{j,a}^{\mathrm{GRP}}(\mathbf{x};\mathbf{R}^{(i)}) = sgn(\mathbf{r}_j^{(i)} \cdot \mathbf{x} - \mathbf{r}_a^{(i)} \cdot \mathbf{x})$ for $j \in [m]$. From this, we obtain the following relationship.

$$h_{\mathrm{GRP}}(\mathbf{x};\mathbf{R}^{(i)}) = a \Leftrightarrow \mathbb{P}_{j,a}^{\mathrm{GRP}}(\mathbf{x};\mathbf{R}^{(i)}) = 0, \forall j \in [m].$$

Thus, by converting each coordinate of $H_{GRP}(\mathbf{x};\mathbf{R})$ to the above relation, we can reformulate GRP-IoM into our LSP notation. We give a detailed algorithm in Algorithm 4

Algorithm 4 Convert GRP-IoM to LSP

Require: $(\alpha, \mathbf{R})$
Parse $\mathbf{R}$ as $\left\{\mathbf{R}^{(1)}, \cdots, \mathbf{R}^{(n)}\right\}$
Parse α as $(\alpha_1, \cdots, \alpha_n)$
for $i \in [n]$ **do**
  Parse $\mathbf{R}^{(i)}$ as $\left[\mathbf{r}_1^{(i)} \| \cdots \| \mathbf{r}_m^{(i)}\right]^T$
  for $j \in [m]$ **do**
    Set $k \leftarrow (i-1)m + j$ and $a \leftarrow \alpha_i$
    Set function $\mathbb{P}_k^{\mathrm{GRP}}(\mathbf{x}; \mathbf{R}^{(i)}) = sgn(\mathbf{r}_j^{(i)} \cdot \mathbf{x} - \mathbf{r}_a^{(i)} \cdot \mathbf{x})$ for some $\mathbf{x} \in \mathbb{R}^d$
  end for
end for
return $\mathbb{P}^{\mathrm{GRP}} = \left\{\mathbb{P}_1^{\mathrm{GRP}}, \cdots, \mathbb{P}_{nm}^{\mathrm{GRP}}\right\}$

C.2 URP-IoM

For convenience, we denote $\Sigma^{(i)} = \left\{\sigma^{(i,j)} : j \in [p]\right\}$. Suppose that $\mathbf{x} \in \mathbb{R}^d$ is given. Let us denote $H_{\mathrm{URP}}(\mathbf{x}; \Sigma) = (h_{\mathrm{URP}}(\mathbf{x}; \Sigma^{(1)}), \cdots, h_{\mathrm{URP}}(\mathbf{x}; \Sigma^{(n)}))$. For a fixed $i \in [n]$:

$$h_{\mathrm{URP}}(x; \Sigma^{(i)}) = a \Leftrightarrow \left(\prod_{j=1}^{p} \sigma^{(i,j)}(\mathbf{x})\right)_t - \left(\prod_{j=1}^{p} \sigma^{(i,j)}(\mathbf{x})\right)_a \leq 0, \forall t \in [w].$$

In this case, an LSP $\mathbb{P}_{t,a}^{\mathrm{URP}}$ for $t, a \in [w]$ can be defined as

$$\mathbb{P}_{t,a}^{\mathrm{URP}}(\mathbf{x}; \Sigma^{(i)}) = sgn\left(\left(\prod_{j=1}^{p} \sigma^{(i,j)}(\mathbf{x})\right)_t - \left(\prod_{j=1}^{p} \sigma^{(i,j)}(\mathbf{x})\right)_a\right).$$

From this, we have

$$h_{URP}(\mathbf{x}; \Sigma^{(i)}) = a \Leftrightarrow \mathbb{P}_{t,a}^{\mathrm{URP}}(\mathbf{x}; \Sigma^{(i)}) = 0, \forall t \in [w].$$

Therefore, we conclude that URP-IoM can also be converted to our LSP notation and the precise algorithm is given in Algorithm 5.

Algorithm 5 Convert URP-IoM to LSP

Require: $(\alpha, \mathbf{R})$ and $w \in \mathbb{N}$

Parse $\mathbf{R}$ as $\left\{\mathbf{R}^{(1)}, \cdots, \mathbf{R}^{(n)}\right\}$
Parse α as $(\alpha_1, \cdots, \alpha_n)$
for $i \in [n]$ **do**
  Parse $\mathbf{R}^{(i)}$ as $\left\{\sigma^{(i,1)}, \cdots, \sigma^{(i,p)}\right\}$
  for $j \in [w]$ **do**
    Set $k \leftarrow (i-1)w + j$ and $a \leftarrow \alpha_i$
    Set $M(\cdot; \mathbf{R}^{(i)}) \leftarrow \left(\prod_{m=1}^{p} \sigma^{(i,m)}(\cdot)\right)$
    Set $\mathbb{P}_k^{\mathrm{URP}}(\cdot; \mathbf{R}^{(i)}) \leftarrow sgn(M(\cdot; \mathbf{R}^{(i)})_j - M(\cdot; \mathbf{R}^{(i)})_a)$
  end for
end for
return $\mathbb{P}^{\mathrm{URP}} = \left\{\mathbb{P}_1^{\mathrm{URP}}, \cdots, \mathbb{P}_{nw}^{\mathrm{URP}}\right\}$

C.3 ABH

Note that for any $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^d$, $H_{\mathrm{ABH}}(\mathbf{x}; \mathbf{R}) = H_{\mathrm{ABH}}(\hat{\mathbf{x}}; \mathbf{R})$ if and only if $sgn(\mathbf{r}_j^{(i)} \cdot \mathbf{x}) = sgn(\mathbf{r}_j^{(i)} \cdot \hat{\mathbf{x}}), \forall i \in [pq]$ and $j \in [r]$. This condition is equivalent to a random projection-based LSH for $n = pqr$, *i.e.*, the LSP corresponding to ABH is $\mathbb{P}_{i,j}^{\mathrm{ABH}}(\mathbf{x}; \mathbf{R}) = sgn(\mathbf{r}_j^{(i)} \cdot \mathbf{x})$ Thus, ABH can be reformulated using our LSP notation, and this is equivalent to BioHashing in the sense of LSP. The detailed algorithm is given in Algorithm 6.

Algorithm 6 Convert ABH to LSP

Require: $(\mathrm{A}, \mathbf{R})$

Parse $\mathbf{R}$ as $\left\{\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \cdots, \mathbf{R}^{(pq)}\right\}$
Parse A as $(\hat{\alpha}_{ij}) \in [2^r]^{p \times q}$
for $i \in [p]$ **do**
  for $j \in [q]$ **do**
    Set $l \leftarrow (i-1)q + j$
    Parse $\mathbf{R}^{(l)}$ as $\left[\mathbf{r}_1^{(l)} || \cdots || \mathbf{r}_r^{(l)}\right]^T$
    Convert $\hat{\alpha}_{ij}$ to binary string $(\alpha_{l1}\alpha_{l2}\cdots\alpha_{lr})_2$
    for $k \in [r]$ **do**
      Set $n \leftarrow (l-1)r + k$ and $a \leftarrow \alpha_{lk}$
      Set function $\mathbb{P}_n^{\mathrm{ABH}}(\mathbf{x}; \mathbf{r}_k^{(l)}) = (-1)^a sgn(\mathbf{r}_k^{(l)} \cdot \mathbf{x})$ for some $\mathbf{x} \in \mathbb{R}^d$
    end for
  end for
end for
return $\mathbb{P}^{\mathrm{ABH}} = \left\{\mathbb{P}_1^{\mathrm{ABH}}, \cdots, \mathbb{P}_{pqr}^{\mathrm{ABH}}\right\}$

D Detailed Experimental Settings

In this section, we provide the detailed settings for the experiment reported in the main paper.

D.1 Formal Description of the Evaluation Metrics

Recall that we used the true accept ratio (TAR) and the false accept ratio (FAR) as evaluation metrics for describing the performance of the given biometric authentication system (Ext, V_{BA}). We give a precise definition for each of them. For simplicity, let us denote $\mathcal{D}$ as the benchmark dataset that consists of pairs of biometrics. We assume that $\mathcal{D}$ splits into $\mathcal{D}_p$ and $\mathcal{D}_n$, each of which indicates the set of positive pairs and negative pairs, respectively. From this, we define the true accept ratio (TAR) and false alarm ratio (FAR) with respect to the threshold τ_{BA} as follows:

$$\mathrm{TAR}(\tau_{\mathsf{BA}}) = \frac{|\{(x,y) \in \mathcal{D}_p : V_{\mathsf{BA}}(Ext(x), Ext(y), \tau_{\mathsf{BA}}) = 1\}|}{|\mathcal{D}_p|},$$
$$\mathrm{FAR}(\tau_{\mathsf{BA}}) = \frac{|\{(x,y) \in \mathcal{D}_n : V_{\mathsf{BA}}(Ext(x), Ext(y), \tau_{\mathsf{BA}}) = 1\}|}{|\mathcal{D}_n|}.$$

For the performance evaluation of the authentication scheme with BTP $(Ext, T, V_{\mathsf{BTP}})$, we follow the same definition, except for applying the T to the output Ext and running the V_{BTP} algorithm within the same randomness that was used for T. If the threshold τ_{BTP} for the BTP is specified to some value, then we will simply denote TAR@FAR.

In addition, we considered two scenarios motivated by [8]: Type-1 attack and Type-2 attack. In the Type-1 attack, the adversary attempts to impersonate the target LSH-based BTP by reconstructing the biometric from the compromised protected template. To be precise, for the adversary $\mathcal{A}$, the target authentication scheme with BTP $(Ext, T, V_{\mathsf{BTP}})$ and the threshold τ_{BTP}, we define $\mathrm{ASR}_1(\tau_{\mathsf{BTP}}) = \frac{|\mathcal{D}_1(\tau_{\mathsf{BTP}})|}{|\mathcal{D}|}$ on the the benchmark dataset $\mathcal{D}$[3], where

$$\mathcal{D}_1(\tau_{\mathsf{BTP}}) = \left\{ \begin{array}{l} x \in \mathcal{D} : \\ V_{\mathsf{BTP}}(s, T(Ext(\tilde{x}); R_x), \tau_{\mathsf{BTP}}) = 1, \\ \text{where } R_x \xleftarrow{\$} \mathcal{R},\ s \leftarrow T(Ext(x); R_x) \text{ and} \\ \tilde{x} \leftarrow \mathcal{A}(s, R_x). \end{array} \right\}$$

for the random source $\mathcal{R}$.

On the other hand, the Type-2 attack was designed to consider the scenario that the adversary attempts to impersonate another system where the identity corresponding to the compromised template is enrolled with different but the same type of biometrics. To address the above scenario, we define the Type-2 attack success rate (ASR_2) by evaluating the probability that the adversary succeeds in impersonating the unprotected biometric authentication system (Ext, V_{BA}) via retrieved biometrics from a stolen protected template. The precise definition is as follows: under the same notation on defining ASR_1 with the threshold τ_{BA}

[3] Here, we consider $\mathcal{D}$ as the set of biometrics, rather than the set of pairs of them.

for unprotected biometric authentication system, $\mathrm{ASR}_2(\tau) = \frac{|\mathcal{D}_2(\tau_{\mathsf{BA}})|}{|\mathcal{D}_p|}$, where

$$\mathcal{D}_2(\tau_{\mathsf{BA}}) = \left\{ \begin{array}{l} (x,y) \in \mathcal{D}_p : \\ V_{\mathsf{BA}}(Ext(x), Ext(\tilde{y}), \tau_{\mathsf{BA}}) = 1, \\ \text{where } R_y \xleftarrow{\$} \mathcal{R},\ s \leftarrow T(Ext(y); R_y) \text{ and} \\ \tilde{y} \leftarrow \mathcal{A}(s, R_y). \end{array} \right\}.$$

D.2 Implementation of Target LSHs

We selected three well-known proposals, GRP-IoM, URP-IoM [6], and ABH [7], as *transformation algorithms*. Each algorithm is implemented by following Algorithm 1-3 provided in Section B[4]. We tested each LSH-based BTP on various parameter settings, and the result is present in the Table 1. We selected the setting that yields the best benchmark result on each BTP and the precise parameter settings are as follows: $(n,m) = (300,16)$ for GRP-IoM, $(n,w,p) = (600,100,2)$ for URP-IoM, and $(p,q,r) = (50,60,2)$ for ABH. For each setting, we obtain the verification thresholds $\tau_{\mathrm{GRP}} = 0.137$, $\tau_{\mathrm{URP}} = 0.038$ and $\tau_{\mathrm{ABH}} = 0.7$, respectively.

(n,m)	GRP-IoM	(n,w,p)	URP-IoM
(100,16)	99.00%@3e-3	(500,100,2)	98.67%@1e-3
(200,16)	99.47%@2e-3	(600,100,2)	**99.03%@7e-4**
(300,16)	**99.63%@3e-4**	(600,100,3)	98.73%@3e-3
(400,16)	99.60%@1e-3	(700,100,2)	99.27%@4e-3

(p,q)	$r=2$	$r=3$	$r=4$
(50, 40)	99.63%@3e-4	99.67%@3e-4	99.67%@3e-4
(50, 60)	**99.70%@1e-3**	99.67%@1e-3	99.57%@0
(50, 80)	99.67%@1e-3	99.67%@1e-3	**99.70%@1e-3**
(50, 100)	99.63%@3e-4	**99.70%@1e-3**	99.63%@0

Table 1: The TAR@FAR evaluation results for GRP-IoM, URP-IoM (left) and ABH (right) with various parameters. The setting that gives the best benchmark performance is emphasized in **bold**.

D.3 Implementation of Each Attack Method

Proposed Method For the choice of a root finding algorithm, we adopted the conjugate gradient method-based algorithm [9] for GRP-IoM and ABH, and quasi-Newton's method for URP-IoM. For URP-IoM, we limited the maximum update step to 50 for the balance of attack performance and execution time. Note that other sophisticated root-finding algorithms may be exploited, but in our experiment, we utilized such basic algorithms to validate our attack methodology.

Genetic Algorithm-based Method [2] On the implementation of the genetic algorithm, we conduct the following procedure with the setting. The algorithm starts by sampling initial vectors from the standard normal distribution. We select the size of the group per generation as 200, and the selection and mutation processes are followed. We set the objective function to be maximized as the matching score between the hashed value in the target template and the hashed value that was made from vectors in the current generation. For the mutation phase, we add noise vectors sampled from a normal distribution with a mean of 0 and a standard deviation of 0.1. The total number of iterations is set to 1,000.

Optimization-based Method [3] We applied the same technique to attack URP-IoM as introduced in their original paper. In addition, when solving the linear programming problem, we used the CVXOPT [1] library as [3] did. We conducted the same analysis as previous

[4]For more details of our implementation, please refer to our code at github.

attack methods we investigated in the same experimental setting as ours. During the experiment, we observed that the algorithm failed to solve the given constraint system for the following reasons: (1) the corresponding Karush-Kuhn-Tucker matrix becomes singular during optimization, or (2) the corresponding constraint system does not have a sufficient rank. For the former case, the algorithm returns some solutions, whereas the latter case returns nothing. Empirically speaking, the latter case happens in nearly 50% of the total attempts. We only analyzed the result when the second case did not occur.

References

[1] Martin Andersen, Joachim Dahl, and Lieven Vandenberghe. Python software for conved optimization (cvxopt). URL http://cvxopt.org/#. Accessed: 2023-04-08.

[2] Xingbo Dong, Jaewoo Park, Zhe Jin, Andrew Beng Jin Teoh, Massimo Tistarelli, and KokSheik Wong. On the risk of cancelable biometrics. *arXiv preprint arXiv:1910.07770*, 2019.

[3] Loubna Ghammam, Koray Karabina, Patrick Lacharme, and Kevin Thiry-Atighehchi. A cryptanalysis of two cancelable biometric schemes based on index-of-max hashing. *IEEE Transactions on Information Forensics and Security*, 15:2869–2880, 2020.

[4] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[5] Andrew Teoh Beng Jin, David Ngo Chek Ling, and Alwyn Goh. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern recognition*, 37(11):2245–2255, 2004.

[6] Zhe Jin, Jung Yeon Hwang, Yen-Lung Lai, Soohyung Kim, and Andrew Beng Jin Teoh. Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing. *IEEE Transactions on Information Forensics and Security*, 13(2):393–407, 2017.

[7] Yenlung Lai, Zhe Jin, KokSheik Wong, and Massimo Tistarelli. Efficient known-sample attack for distance-preserving hashing biometric template protection schemes. *IEEE Transactions on Information Forensics and Security*, 16:3170–3185, 2021.

[8] Guangcan Mai, Kai Cao, Pong C Yuen, and Anil K Jain. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1188–1202, 2018.

[9] G Nagaraja and Gopalrao Krishna. An algorithm for the solution of linear inequalities. *IEEE Transactions on Computers*, 100(4):421–427, 1974.