Formal Accuracy Analysis of a Biometric Data Transformation and Its Application to Secure Template Generation

Shoukat Ali[1], Koray Karabina[2,3] and Emrah Karagoz[2]
[1]*University of Calgary, Canada*
[2]*Florida Atlantic University, U.S.A.*
[3]*National Research Council, Canada*

Keywords: Secure Biometric Templates, Face Recognition, Keystroke Dynamics.

Abstract: Many of the known secure template constructions transform real-valued feature vectors to integer-valued vectors, and then apply cryptographic transformations. Throughout this two-step transformation, the original biometric data is distorted, whence it is natural to expect some loss in the accuracy. As a result, the accuracy and security of the whole system should be analyzed carefully. In this paper, we provide a formal accuracy analysis of a generic and intuitive method to transform real-valued feature vectors to integer-valued vectors. We carefully parametrize the transformation, and prove some accuracy-preserving properties of the transformation. Second, we modify a recently proposed noise-tolerant template protection algorithm and combine it with our transformation. As a result, we obtain a secure biometric authentication method that works with real-valued feature vectors. A key feature of our scheme is that a second factor (e.g., user password, or public/private key) is not required, and therefore, it offers certain advantages over cancelable biometrics or homomorphic encryption methods. Finally, we verify our theoretical findings through implementations over public face and keystroke dynamics datasets and provide some comparisons.

1 INTRODUCTION

Convenience and fraud prevention requirements in systems create a growing demand for biometric authentication. A biometric authentication system consists of two phases: enrollment and verification. In the enrollment phase, a user's biometric sample is collected via a sensor, and distinctive characteristics are derived using a feature extraction algorithm. A digital representation of these characteristics (the feature vector or the template) is stored in the system database. In the verification phase, a matching algorithm takes a pair of templates as input, and outputs a score. A decision (accept or reject) is made based on the matching score. Therefore, biometric templates should be stored in some protected form to guard against adversarial attacks. Since 1994 (Bodo, 1994; Schmidt et al., 1996), there have been tremendous research and development efforts for creating secure biometric schemes. In the most general terms, we can classify biometric template protection methods under four main categories: biometric cryptosystems (BC) cancelable biometrics (CB), secure multiparty computation based biometrics (SC), also known as keyed biometrics (KB), and hybrid biometrics (HB) We refer the reader to (Natgunanathan et al., 2016) for more details on biometric template protection methods.

In BC and SC (whence in HB), cryptographic functions and transformations are the main tools to create secure templates. By construction, the underlying cryptographic primitives are defined over some particular discrete domains, and therefore, feature vectors are supposed to be some binary, or integer-valued vectors. For example, the BC- and SC-based secure fingerprint and iris identification algorithms in (Blanton and Gasti, 2011; Karabina and Canpolat, 2016; Tuyls et al., 2005) assume that feature vectors are represented as fixed length binary vectors, and the Hamming distance (and some variants of the Hamming distance) is used as a way of measuring the similarity between feature vectors. More generally, a large class of template protection algorithms assume that feature vectors are integer-valued, and the similarity scores are calculated based on Hamming distance, set difference distance, or edit distance; see (Natgunanathan et al., 2016). On the other hand, biometric data, in general, is represented through real-valued feature vectors as in the case of

Ali, S., Karabina, K. and Karagoz, E.
Formal Accuracy Analysis of a Biometric Data Transformation and Its Application to Secure Template Generation.
DOI: 10.5220/0009888604850496
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 485-496
ISBN: 978-989-758-446-6

face recognition (Huang et al., 2007; Kanade et al., 2009; Rathgeb et al., 2014) and keystroke dynamics (Banerjee and Woodard, 2012; Bours and Barghouthi, 2009; Fairhurst and Da Costa-Abreu, 2011; Killourhy and Maxion, 2009). Therefore, many of the known secure template constructions, including the examples given above, would not be immediately applicable when feature vectors are composed of non-integer, real numbers.

Contributions and Organization of the Paper.

1. **Provable Accuracy.** In Section 2, we recall a generic method to transform real-valued feature vectors to integer-valued vectors. This method is derived from a simple and intuitive transformation. However, the actual challenge is to carefully parametrize the transformation and rigorously prove that the method is accuracy-preserving. In summary, given a (non-cryptographic) biometric authentication system that takes real-valued vectors as input, our construction yields a new system that now takes integer-valued vectors as input. Our key result Corollary 1 proves that the rates of the new system can be made arbitrarily close to the rates of the original system.
2. **Accuracy Evaluation.** For practical purposes, we evaluate our theoretical findings over two publicly available biometric datasets: the LFW face dataset (Huang et al., 2007) and the keystroke-dynamics dataset (Killourhy and Maxion, 2009). Our results are competetive with previously reported accuracy results derived from the same datasets using some state-of-the-art biometric recognition algorithms. For more details, please refer to Section 3.
3. **Cryptographic Implementation.** As stated previously, a major advantage of transforming real-valued feature vectors to integer-valued vectors is the ability to cryptographically secure biometric templates. In order to evaluate the practical impact of our results, we modify a recently proposed noise tolerant secure template generation and comparison algorithm NTT-Sec (Karabina and Canpolat, 2016) and combine it with our transformation. As a result, we obtain a secure biometric authentication method that works with real-valued feature vectors. A key feature of our scheme is that a second factor (e.g., user password, or public and private key) is not required, and therefore, it offers certain advantages over cancelable biometrics or homomorphic encryption methods. We verify our theoretical findings through implementations over the LFW dataset (Huang et al., 2007) and the keystrokes-dynamics dataset (Killourhy and Maxion, 2009). For more details, please refer to Section 4 and Table 4 for comparison.

As a result, we expect that our new construction and its explicit accuracy analysis will enable cryptographic techniques to secure biometrics at a larger scale.

2 AN ACCURACY-PRESERVING TRANSFORMATION

Let s be a positive real number called *scaling factor*. We define the following *scale-then-round* transformation StR_s that maps a real-valued vector of length n to an integer-valued vector of the same length.

Definition 1 (The Scale-then-Round transformation StR_s)**.** *For a real-valued vector $x = (x_1, x_2, \ldots, x_n)$, the map $\mathsf{StR}_s : \mathbb{R}^n \to \mathbb{Z}^n$ is defined as*

$$\mathsf{StR}_s(x) = (\lfloor sx_1 \rceil, \lfloor sx_2 \rceil, \ldots, \lfloor sx_n \rceil)$$

where $\lfloor \cdot \rceil$ is the nearest integer function.

Now, let $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a distance function that satisfies the homogeneity and translation properties: for any $x, y \in \mathbb{R}^n$ and $u \in \mathbb{R}$, $d(ux, uy) = |u| d(x, y)$ and $d(x, y) = d(x + u, y + u)$. We have the following lemma:

Lemma 1. *Let the transformation $\mathsf{StR}_s : \mathbb{R}^n \to \mathbb{Z}^n$ and the distance function d be defined as above. Let $x, y \in \mathbb{R}^n$ be any real-valued vectors and denote their transformations as the integer valued vectors $X = \mathsf{StR}_s(x)$ and $Y = \mathsf{StR}_s(y)$ in $\mathbb{Z}^n$. Then*

$$|d(X,Y) - sd(x,y)| \leq 2\varepsilon_{max},$$

where $\varepsilon_{max} = \max_{u \in \mathbb{R}^n} d(su, \mathsf{StR}_s(u))$. Equivalently,

$$sd(x,y) - 2\varepsilon_{max} \leq d(X,Y) \leq sd(x,y) + 2\varepsilon_{max} \quad \text{and}$$
$$\frac{d(X,Y) - 2\varepsilon_{max}}{s} \leq d(x,y) \leq \frac{d(X,Y) + 2\varepsilon_{max}}{s}.$$

Proof. Using the triangular inequality on both $d(sx, sy)$ and $d(X, Y)$ we have

$$d(X,Y) \leq d(X,sx) + d(Y,sy) + d(sx,sy) \quad \text{and}$$
$$d(sx,sy) \leq d(X,sx) + d(Y,sy) + d(X,Y).$$

Since $d(sx, sy) = sd(x, y)$ and both $d(X, sx)$ and $d(Y, sy)$ are bounded above by $\varepsilon_{\max}$, we have the desired results. □

Remark 1. *Lemma 1 shows that given a pair of vectors $x, y \in \mathbb{R}^n$, $d(X,Y)/s$ lies in the neighborhood of the distance $d(x,y)$ up to an error margin of $2\varepsilon_{max}/s$.*

In the next theorem, we observe that if the Minkowski distance $d_p(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$ is deployed, then $d_p(X,Y)/s$ converges to $d_p(x,y)$. This result will later be used in Theorem 2 where we compare the error rates of our new system (with integer valued vectors) and the original system (with real valued vectors).

Theorem 1. *Let d_p be the Minkowski distanced defined on $\mathbb{R}^n \times \mathbb{R}^n$, and let $X = \mathsf{StR}_s(x)$, $Y = \mathsf{StR}_s(y)$, as defined before. For a given $\varepsilon > 0$, if a scalar s is chosen such that $s \geq n^{1/p}/\varepsilon$, then*

$$|d(X,Y)/s - d(x,y)| \leq \varepsilon \qquad \forall x,y \in \mathbb{R}^n$$

Proof. Note that

$$\begin{aligned} \varepsilon_{\max} &= \max_{u \in \mathbb{R}^n} d_p(su, \mathsf{StR}_s(u)) \qquad (1) \\ &\leq \left(\sum_{i=1}^{n} (1/2)^p \right)^{1/p} = \frac{n^{1/p}}{2}, \end{aligned}$$

where the last inequality follows because $|su_i - \lfloor su_i \rceil| \leq 1/2$ for all i. Now, for a given $\varepsilon > 0$, choose s such that $s \geq n^{1/p}/\varepsilon$. This implies $n^{1/p}/s \leq \varepsilon$, and it follows from Lemma 1 and (1) that $|d(X,Y)/s - d(x,y)| \leq 2\varepsilon_{\max}/s \leq n^{1/p}/s \leq \varepsilon$, as required. □

Next, we provide some theoretical estimates on the new system's False Accept Rate (FAR) and False Reject Rate (FRR) as a function of the original system's error rates. Let GenP and ImpP denote the list of genuine pairs and the list of impostor pairs, respectively. Corresponding to these lists, let GenP' and ImpP' denote the lists of transformed version of GenP and ImpP respectively, defined as

$$\mathsf{GenP}' = \{(\mathsf{StR}_s(x), \mathsf{StR}_s(y)) : (x,y) \in \mathsf{GenP}\}$$

$$\mathsf{ImpP}' = \{(\mathsf{StR}_s(x), \mathsf{StR}_s(y)) : (x,y) \in \mathsf{ImpP}\}$$

Thus, $\#\mathsf{GenP} = \#\mathsf{GenP}'$ and $\#\mathsf{ImpP} = \#\mathsf{ImpP}'$ where the symbol # represents the number of pairs. Note that all of them are lists, not sets. Therefore it is possible to see identical pairs, especially in the lists GenP' and ImpP' because there may be several identical pairs which are the transformation of different pair of vectors, i.e. there may exists $(x_1,y_1),(x_2,y_2)$ such that $(x_1,y_1) \neq (x_2,y_2)$ but $(\mathsf{StR}_s(x_1), \mathsf{StR}_s(y_1)) = (\mathsf{StR}_s(x_2), \mathsf{StR}_s(y_2))$.

For a distance function d on $\mathbb{R}^n$ and $t \in \mathbb{R}^+$, the $\mathsf{FAR}(t)$ and $\mathsf{FRR}(t)$ are defined as follows:

$$\mathsf{FAR}(t) = \frac{\#\{(x,y) \in \mathsf{ImpP} : d(x,y) \leq t\}}{\#\mathsf{ImpP}},$$

$$\mathsf{FRR}(t) = \frac{\#\{(x,y) \in \mathsf{GenP} : d(x,y) > t\}}{\#\mathsf{GenP}}.$$

Now we can define the corresponding rates for $T \in \mathbb{R}^+$:

$$\mathsf{FAR}'(T) = \frac{\#\{(X,Y) \in \mathsf{ImpP}' : d(X,Y) \leq T\}}{\#\mathsf{ImpP}'},$$

$$\mathsf{FRR}'(T) = \frac{\#\{(X,Y) \in \mathsf{GenP} : d(X,Y) > T\}}{\#\mathsf{GenP}}.$$

We have the following lemma:

Lemma 2. *Let s be the scaling factor in transformation StR_s and define $\varepsilon_{max} = \max_{u \in \mathbb{R}^n} d(su, \mathsf{StR}_s(u))$. Then $\mathsf{FAR}\left(t - \frac{2\varepsilon_{max}}{s}\right) \leq \mathsf{FAR}'(st) \leq \mathsf{FAR}\left(t + \frac{2\varepsilon_{max}}{s}\right)$, $\mathsf{FRR}\left(t + \frac{2\varepsilon_{max}}{s}\right) \leq \mathsf{FRR}'(st) \leq \mathsf{FRR}\left(t - \frac{2\varepsilon_{max}}{s}\right)$.*

Proof. For the first inequality, define $(X,Y) = (\mathsf{StR}_s(x), \mathsf{StR}_s(y))$ for an impostor pair (x,y) in the list ImpP. Then by using the inequalities in Lemma 1, we have

$$\begin{aligned} & d(x,y) \leq t - \frac{2\varepsilon_{\max}}{s} \\ \implies & d(X,Y) \leq s\big(t - \frac{2\varepsilon_{\max}}{s}\big) + 2\varepsilon_{\max} = st \\ \implies & d(x,y) \leq \frac{st + 2\varepsilon_{\max}}{s} = t + \frac{2\varepsilon_{\max}}{s}. \end{aligned}$$

These inequalities mean that

- Any impostor pair (x,y) having distance less than or equal to $t - \frac{2\varepsilon_{\max}}{s}$, which is already counted in the rate $\mathsf{FAR}\left(t - \frac{2\varepsilon_{\max}}{s}\right)$, has its transformed pair (X,Y) with a distance less than or equal to st. Therefore, this transformed pair (X,Y) is needed to be counted in the rate $\mathsf{FAR}'(st)$. Thus,

$$\mathsf{FAR}\left(t - \frac{2\varepsilon_{\max}}{s}\right) \leq \mathsf{FAR}'(st).$$

- Any pair (X,Y) in the list ImpP' having distance less than or equal to st, which is already counted in the rate $\mathsf{FAR}'(st)$, has its pre-transformed impostor pair (x,y) in the list ImpP with a distance less than or equal to $t + \frac{2\varepsilon_{\max}}{s}$. Therefore, this imposter pair (x,y) is needed to be counted in the rate $\mathsf{FAR}\left(t + \frac{2\varepsilon_{\max}}{s}\right)$. Thus,

$$\mathsf{FAR}'(st) \leq \mathsf{FAR}\left(t + \frac{2\varepsilon_{\max}}{s}\right).$$

For the second inequality, now let (X,Y) denote the transformation $(\mathsf{StR}_s(x), \mathsf{StR}_s(y))$ for a genuine pair

(x,y) in the list GenP. Then by using the inequalities in Lemma 1, we have

$$d(x,y) > t + \frac{2\varepsilon_{\max}}{s}$$
$$\implies d(X,Y) > s\big(t + \frac{2\varepsilon_{\max}}{s}\big) - 2\varepsilon_{\max} = st$$
$$\implies d(x,y) > \frac{st - 2\varepsilon_{\max}}{s} = t - \frac{2\varepsilon_{\max}}{s}.$$

These inequalities mean that

- Any genuine pair (x,y) in the list GenP having distance greater than $t + \frac{2\varepsilon_{\max}}{s}$, which is already counted in the rate $\mathsf{FRR}\left(t + \frac{2\varepsilon_{\max}}{s}\right)$, has its transformed pair (X,Y) with a distance greater than st. Therefore, this transformed pair (X,Y) is needed to be counted in the rate $\mathsf{FRR}'(st)$. Thus,
$$\mathsf{FRR}\left(t + \frac{2\varepsilon_{\max}}{s}\right) \leq \mathsf{FRR}'(st).$$
- Any pair (X,Y) in the list GenP$'$ having distance greater than st, which is already counted in the rate $\mathsf{FRR}'(st)$, has a pre-transformed genuine pair (x,y) in the list GenP with a distance greater than $t - \frac{2\varepsilon_{\max}}{s}$. Therefore, this genuine pair (x,y) is needed to be counted in the rate $\mathsf{FRR}\left(t - \frac{2\varepsilon_{\max}}{s}\right)$. Thus,
$$\mathsf{FRR}'(st) \leq \mathsf{FRR}\left(t - \frac{2\varepsilon_{\max}}{s}\right).$$

□

Theorem 2. *Let d_p be the Minkowski distance defined on $\mathbb{R}^n \times \mathbb{R}^n$, and let $X = \mathsf{StR}_s(x)$, $Y = \mathsf{StR}_s(y)$, as defined before. For a given $\varepsilon > 0$, if a scalar s is chosen such that $s \geq n^{1/p}/\varepsilon$, then*

$$\mathsf{FAR}(t-\varepsilon) \leq \mathsf{FAR}'(st) \leq \mathsf{FAR}(t+\varepsilon), \text{ and}$$
$$\mathsf{FRR}(t+\varepsilon) \leq \mathsf{FRR}'(st) \leq \mathsf{FRR}(t-\varepsilon).$$

Proof. Let $\varepsilon > 0$ be given and choose s such that $s \geq n^{1/p}/\varepsilon$. We already observed in the proof of Theorem 1 that $2\varepsilon_{\max}/s \leq n^{1/p}/s \leq \varepsilon$. Using this inequality together with the inequality

$$\mathsf{FAR}'(st) \leq \mathsf{FAR}\left(t + \frac{2\varepsilon_{\max}}{s}\right)$$

from Lemma 2, and the fact that $\mathsf{FAR}(t_2) \geq \mathsf{FAR}(t_1)$ for $t_2 \geq t_1$, we obtain

$$\mathsf{FAR}'(st) \leq \mathsf{FAR}\left(t + \frac{2\varepsilon_{\max}}{s}\right) \leq \mathsf{FAR}(t+\varepsilon).$$

This proves one of the four inequalities in the statement, and the other three inequalities can be proved similarly. □

Corollary 1. *For any given $\bar{\varepsilon} > 0$, there exists $T \geq 0 \in \mathbb{R}$ such that*

$$\mathsf{FAR}(t) - \bar{\varepsilon} \leq \mathsf{FAR}'(T) \leq \mathsf{FAR}(t) + \bar{\varepsilon}, \text{ and} \quad (2)$$
$$\mathsf{FRR}(t) - \bar{\varepsilon} \leq \mathsf{FRR}'(T) \leq \mathsf{FRR}(t) + \bar{\varepsilon} \quad (3)$$

Proof. Given $\bar{\varepsilon} > 0$ as in the statement of the corollary, one can choose $\varepsilon > 0$ such that

$$\mathsf{FAR}(t+\varepsilon) \leq \mathsf{FAR}(t) + \bar{\varepsilon},$$
$$\mathsf{FAR}(t) - \bar{\varepsilon} \leq \mathsf{FAR}(t-\varepsilon),$$
$$\mathsf{FRR}(t-\varepsilon) \leq \mathsf{FRR}(t) + \bar{\varepsilon},$$
$$\mathsf{FRR}(t) - \bar{\varepsilon} \leq \mathsf{FRR}(t+\varepsilon),$$

because $\mathsf{FAR}(t)$ and $\mathsf{FRR}(t)$ can be modelled as a continuously increasing and decreasing function parameterized by t, respectively. Now, choosing s as suggested by Theorem 2, and combining the inequalities of Theorem 2 with the inequalities above, we can write

$$\begin{aligned}\mathsf{FAR}(t) - \bar{\varepsilon} &\leq \mathsf{FAR}(t-\varepsilon) \leq \mathsf{FAR}'(st) \\ &\leq \mathsf{FAR}(t+\varepsilon) \leq \mathsf{FAR}(t) + \bar{\varepsilon}, \quad \text{and} \\ \mathsf{FRR}(t) - \bar{\varepsilon} &\leq \mathsf{FAR}(t+\varepsilon) \leq \mathsf{FRR}'(st) \\ &\leq \mathsf{FRR}(t-\varepsilon) \leq \mathsf{FRR}(t) + \bar{\varepsilon}.\end{aligned}$$

Finally, setting $T = st$, we conclude

$$\mathsf{FAR}(t) - \bar{\varepsilon} \leq \mathsf{FAR}'(T) \leq \mathsf{FAR}(t) + \bar{\varepsilon}, \quad \text{and} \quad (4)$$
$$\mathsf{FRR}(t) - \bar{\varepsilon} \leq \mathsf{FRR}'(T) \leq \mathsf{FRR}(t) + \bar{\varepsilon} \quad (5)$$

□

Remark 2. *Given a biometric authentication system that takes real-valued feature vectors as input, deploys Minkowski distance d_p in its matching algorithm, and runs at false accept rate $\mathsf{FAR}(t)$ and false reject rate $\mathsf{FRR}(t)$, Theorem 2 and its Corollary 1 assure the existence of a scalar s (and $T = st$) that can be used to transform the system to integer-valued vectors, deploys the same d_p in its matching algorithm, and runs at false accept rate $\mathsf{FAR}'(T)$ and false reject rate $\mathsf{FRR}'(T)$ that are arbitrarily close to $\mathsf{FAR}(t)$ and $\mathsf{FRR}(t)$ of the original system.*

Remark 3. *Smaller values of s would be preferred in cryptographic secure template generation algorithms due to the smaller size of the resulting feature vectors and the smaller threshold values. However, it seems challenging to find a tight lower bound for s in Theorem 2, one can address this gap between theory and practice as we explain in the following remark. One should also be careful to choose s sufficiently large to prevent dictionary attacks. Therefore, in the light of Theorem 2, we outline a procedure in Algorithm 1 to determine a suitable scalar s_0 and a threshold T_0 from a given original system. We assume*

Algorithm 1: To determine suitable parameters s and T.

Input: $t_0, n, p, \bar{\varepsilon}, I_{\mathsf{FAR}}, I_{\mathsf{FRR}}, \mathsf{DS}, \mathsf{MinScalar}$
Output: $s_0 \geq \mathsf{MinScalar}$ and T_0 such that $\mathsf{FAR}'(T_0) \in I_{\mathsf{FAR}}$ and $\mathsf{FRR}'(T_0) \in I_{\mathsf{FRR}}$

1: Set a and b as the corresponding thresholds of $\big(\mathsf{FAR}(t_0) - \bar{\varepsilon}\big)$ and $\big(\mathsf{FRR}(t_0) + \bar{\varepsilon}\big)$, respectively
2: $t_1 \leftarrow \mathrm{Max}\big(a, b\big)$
3: Set c and d as the corresponding thresholds of $\big(\mathsf{FAR}(t_0) + \bar{\varepsilon}\big)$ and $\big(\mathsf{FRR}(t_0) - \bar{\varepsilon}\big)$, respectively
4: $t_2 \leftarrow \mathrm{Min}\big(c, d\big)$
5: $\varepsilon \leftarrow \mathrm{Min}\big((t_0 - t_1), (t_2 - t_0)\big)$
6: $s_0 \leftarrow \lfloor n^{1/p}/\varepsilon \rceil$
7: **while** $s_0 >$ MinScalar and $\Big(\mathsf{FAR}'((s_0-1)t_0) \in I_{\mathsf{FAR}}$ and $\mathsf{FRR}'((s_0-1)t_0) \in I_{\mathsf{FRR}}$ over $\mathsf{DS}\Big)$ **do**
8: $\quad s_0 \leftarrow s_0 - 1$
9: **end while**

that the original system deploys the distance function $d_p(x,y) = (\sum (x_i - y_i)^p)^{1/p}$*, and has some some desired rates* $\mathsf{FAR}(t_0)$*,* $\mathsf{FRR}(t_0)$*, where* $\mathsf{FAR}(t)$*,* $\mathsf{FRR}(t)$ *are measured over some dataset* DS*. For example, one may fix* t_0 *so that the system runs at the equal error rate* $\mathsf{EER} = \mathsf{FAR}(t_0) = \mathsf{FRR}(t_0)$*. Our procedure outputs a value of* $s_0 \geq \mathsf{MinScalar}$ *and a threshold value* T_0 *for which the new system's accuracy is in a close neighborhood of the original system's accuracy. More particularly, new parameters will assure that* $\mathsf{FAR}'(T_0) \in I_{\mathsf{FAR}} = [\mathsf{FAR}(t_0) - \bar{\varepsilon}, \mathsf{FAR}(t_0) + \bar{\varepsilon}]$ *and* $\mathsf{FRR}'(T_0) \in I_{\mathsf{FRR}} = [\mathsf{FRR}(t_0) - \bar{\varepsilon}, \mathsf{FRR}(t_0) + \bar{\varepsilon}]$ *for a given* $\bar{\varepsilon} > 0$*, where* $d_p(X,Y)$ *is used to compute the distance between integer-valued vectors* $X = \mathsf{StR}_s(x)$ *and* $Y = \mathsf{StR}_s(y)$*. In practice,* $\bar{\varepsilon}$ *should be chosen so that the new rates* $\mathsf{FAR}'(T_0)$ *and* $\mathsf{FRR}'(T_0)$ *are close to* $\mathsf{FAR}(t_0)$ *and* $\mathsf{FRR}(t_0)$*, respectively. The correctness of Algorithm 1 follows from Theorem 2.*

3 APPLICATIONS OF THE NEW TRANSFORMATION

In this section, we evaluate our theoretical findings over two publicly available biometric datasets: the LFW dataset (Huang et al., 2007) for face recognition, and the keystrokes-dynamics dataset (Killourhy and Maxion, 2009). Our reasoning for choosing these datasets is that they are widely referenced in the literature, and the biometric features in both of these datasets are represented as real-valued vectors. As an application of our construction, we propose some concrete system parameters to convert these feature vectors into integer-valued vectors, and verify its accuracy preserving property.

3.1 Labeled Faces in the Wild

We use one of the most popular public face datasets that was presented by Gary B. Huang et. al. (Huang et al., 2007) and named "Labeled Faces in the Wild" (LFW). The dataset comprises more than $13,000$ face images of $5,749$ people collected from the web and $1,680$ of them have two or more images. Among the four different available versions of the datasets, we use the original version of the LFW in our experiment.

In our implementation, we have used the face recognition (Python) module of Adam Geitgey (Geitgey, 2017) which he built using the face recognition model in the Dlib library of Davis E. King (King, 2011) where the model was trained on a dataset of about 3 million face images (King, 2017). The Histogram of Oriented Gradients (HOG) and the Convolutional Neural Network (CNN) are the two methods that we used for face detection in our experiment. The HOG is faster than the CNN method but less accurate in detecting faces from the images. For example, we found that CNN only failed to detect the face in *Jeff_Feldman_0001.jpg* while the HOG failed to detect faces in 57 images in the LFW. Therefore, we utilize CNN detector in our experiments and report the results that we found.

In the pre-trained model of Davis E. King, the Euclidean Distance (ED) is measured between two 128-dimensional facial vectors. If the distance is less than or equal to 0.6, then two images are considered a match otherwise, it is a mis-match. The match and mis-match are returned as "True" and "False", respectively, by Adam Geitgey in his face recognition module. In other words, False implies correct identification of impostors in the set of ImpP. Here, the accuracy is measured as follows

$$\begin{aligned} \text{Accuracy} &= \frac{\#\text{True in GenP} + \#\text{False in ImpP}}{\#\text{GenP} + \#\text{ImpP}} \\ &= \frac{\#\text{GenP} \times (1 - \text{FRR}) + \#\text{ImpP} \times (1 - \text{FAR})}{\#\text{GenP} + \#\text{ImpP}} \end{aligned} \tag{6}$$

Each image in the dataset is labelled with a person's name and contains that person's face image. In addition, some images contain faces of people other than the person in the label. In our experiments, we assume that the first detected face is the face of the labelled person. Under this assumption, for GenP, we find #True and #False as $231,752$ and $10,505$, respectively. On the other hand, for ImpP, we find #True and #False as $515,817$ and $86,778,222$, respectively. So the sizes of GenP and ImpP are $242,257$ and $87,294,039$, respectively. Hence, our evaluation

yields 99.40% accuracy using the CNN method and threshold $t = 0.6$ with ED; see Table 1. Note that our accuracy evaluation confirms the results reported by Davis E. King and Adam Geitgey (King, 2017; Geitgey, 2017), and also it is comparable to other state-of-the-art models; see (Huang et al., 2007) for an extensive list of results and comparisons.

3.1.1 Transforming LFW Feature Vectors

As mentioned before, a (detected) face in the image is represented by a 128-dimensional real-valued vector, and the accuracy evaluations are performed using the ED function. In this section, we apply our proposed transformation to obtain 128-dimensional integer-valued feature vectors. We further replace the ED by the Manhattan distance (MD) function. This latter modification allows us to simplify the quadratic distance formula to a linear one, which eventually yields better efficiency in crpytographic computations for secure template comparison.

In our analysis, we focus on three critical threshold values $t = 0.54$, $t = 0.6$, and $t = 0.66$ as shown in Table 1. These three thresholds capture FAR values near 0.001, FAR values near 0.005, and the equal error rate $\mathsf{FAR} = \mathsf{FRR} \approx 0.03$; with respect to the use of the ED function. The threshold $t = 0.6$ provides a basis for comparing our results to previously reported results in (King, 2017; Geitgey, 2017). We should first emphasize that switching from the ED to MD has almost negligible impact on FAR and FRR as shown in the first two rows of Table 1. The critical part is

Table 1: ED = Euclidean Distance, MD = Manhattan Distance, MD_{100}, MD_{1376} = MD where the feature vector components are scaled-then-rounded by integer 100 and 1376, respectively.

Method	FAR ≈ 0.001	FAR ≈ 0.005	EER
ED	$t = 0.54$ FRR $= 0.091159$ FAR $= 0.000897$ Accuracy $= 0.9989$	$t = 0.6$ FRR $= 0.043363$ FAR $= 0.005909$ Accuracy 0.99399	$t = 0.66$ FRR $= 0.033427$ FAR $= 0.033630$ Accuracy $= 0.96637$
MD	$t = 4.846$ FRR $= 0.100096$ FAR $= 0.00087$ Accuracy $= 0.99885$	$t = 5.393$ FRR $= 0.044989$ FAR $= 0.005896$ Accuracy $= 0.993995$	$t = 5.941$ FRR $= 0.03358$ FAR $= 0.03365$ Accuracy $= 0.96635$
MD_{100}	$t = 485$ FRR $= 0.099105$ FAR $= 0.00091$ Accuracy $= 0.99882$	$t = 539$ FRR $= 0.04497$ FAR $= 0.006007$ Accuracy $= 0.993886$	$t = 594$ FRR $= 0.033617$ FAR $= 0.03428$ Accuracy $= 0.965718$
MD_{1376}	$t = 6668$ FRR $= 0.100088$ FAR $= 0.000873$ Accuracy $= 0.99885$	$t = 7421$ FRR $= 0.044973$ FAR $= 0.005908$ Accuracy $= 0.99398$	$t = 8175$ FRR $= 0.033555$ FAR $= 0.033702$ Accuracy $= 0.966299$

to determine a suitable scalar s using Algorithm 1 so as to preserve the accuracy after the StR_s transformation. We explain the process in detail for $t_0 = 5.941$ (i.e. $\mathsf{FAR}(t_0) = 0.03365$ and $\mathsf{FRR}(t_0) = 0.03358$) over the dataset $\mathsf{DS} = \mathsf{LFW}$. First, we choose $\bar{\varepsilon} = 0.01$ so that the new system's error rates would satisfy $\mathsf{FAR}'(T_0) \in I_{\mathsf{FAR}} = [\mathsf{FAR}(5.941) - \bar{\varepsilon}, \mathsf{FAR}(5.941) + \bar{\varepsilon}] = [0.02365, 0.04365]$, and $\mathsf{FRR}'(T_0) \in I_{\mathsf{FRR}} = [\mathsf{FRR}(5.941) - \bar{\varepsilon}, \mathsf{FRR}(5.941) + \bar{\varepsilon}] = [0.02358, 0.04358]$.

Following the steps 1 to 4 in Algorithm 1, we find the smallest $\varepsilon = 0.093$ such that $[\mathsf{FAR}(5.941 - \varepsilon), \mathsf{FAR}(5.941 + \varepsilon)] \subseteq I_{\mathsf{FAR}}$, and $[\mathsf{FRR}(5.941 + \varepsilon), \mathsf{FRR}(5.941 - \varepsilon)] \subseteq I_{\mathsf{FRR}}$. The step 5 in Algorithm 1 initializes s_0 as shown below. Note that $p = 1$ in the MD function.

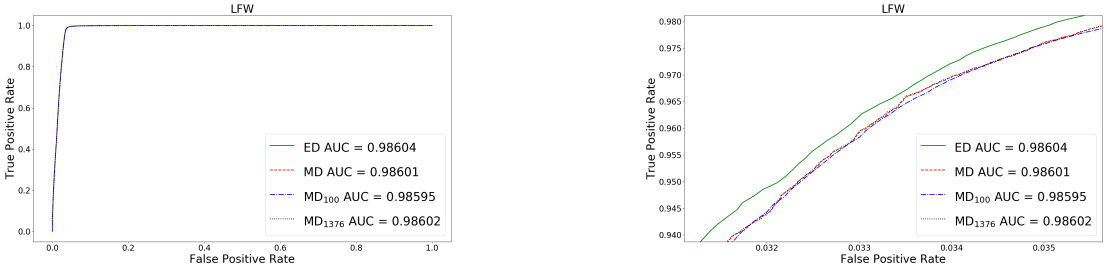
$$s_0 = \lfloor n^{1/p}/\varepsilon \rceil = \lfloor 128/0.093 \rceil = 1376.$$

Next, we need to choose a suitable value for $\mathsf{MinScalar}$. For this, we compute the average feature vector $a = [a_1, ..., a_{128}]$ over all 13233 feature vectors in the LFW dataset, where a_i is the average of the absolute values of the i'th components of the feature vectors. We find that $\min(a) = \min(\{a_i\}) = 0.035, \max(a) = \max(\{a_i\}) = 0.37$, with an average of $\mathrm{Mean}(a) = \sum a_i/128 = 0.098$. We choose $s = \mathsf{MinScalar} = 100$, and obtain $\min(\mathsf{StR}_s(a)) = 4, \max(\mathsf{StR}_s(a)) = 37$, with an average of $\mathrm{Mean}(\mathsf{StR}_s(a)) = 10$. This ensures that creating a dictionary for the set of transformed feature vectors is an infeasible task for an attacker because 10^{128} feature vectors are expected on average. Finally, the Algorithm 1 returns

$$s_0 = 100 \text{ and } T_0 = \lfloor s_0 t_0 \rceil = 594,$$

and the new system's rates become $\mathsf{FAR}'(594) = 0.03428$ and $\mathsf{FRR}'(594) = 0.033617$, which are extremely close to the original system's rates. Please see the results in the last two rows of Table 1 for the new system derived from the original system with the threshold values of $t_0 = 4.846$, $t_0 = 5.393$, and $t_0 = 5.941$. As expected, the new system's rates are close to the original system's rates.

In Figure 1, we show the Receiver Operating Characteristic (ROC) curve and the Area Under Curve (AUC) for ED, MD, MD_{100} and MD_{1376}. In all the following figures, the False Positive Rate and True Positive Rate represent FRR and $(1 - \mathsf{FAR})$, respectively. The curves in Figure 1a depict that the differences among the used techniques are very small which is further supported by AUC. It shows that the AUC of ED and MD are 0.98604 and 0.98601, respectively. In other words, the area differ by 0.00003 only. Furthermore, we find the AUC of MD_{100} and MD_{1376} as 0.98595 and 0.98602, respectively. Clearly, MD_{1376} is a better choice than MD_{100} in terms of accuracy. However, note that the loss of 0.00006 in accuracy, if MD_{100} is used, is actually very small, and it may be tolerable in practice given the efficiency gains in choosing smaller scalars. The ROC



(a) Normal ROC. (b) Zoomed in near EER.

Figure 1: ROC curve and the area underneath called Area under curve (AUC).

curves differences among our used techniques near the EER neighborhood are shown in Figure 1b, which confirm that relative differences are not significant.

3.2 Keystroke-dynamics

The public keystroke-dynamics dataset of Killourhy and Maxion (Killourhy and Maxion, 2009) contains the keystroke-timing of 51 subjects typing the same password in 8 different sessions where each session consists of 50-repetition and only one session per day was performed. From each (password) typing event, 31 timing features were extracted. The authors implemented 14 anomaly-detection algorithms using the R statistical programming language. The performance of each detector was measured by generating an receiver operating characteristic (ROC) curve using the anomaly scores. The authors have reported 0.153 as the average equal error rate (EER) using MD function. Note that the subject identifiers are not in the range of *s001* to *s051* and for further details please see (Killourhy and Maxion, 2009).

Using the MD, we compute the error rates and select the two subjects that exhibit minimum and maximum equal error rate (EER). Actually, these two subjects are tantamount to the best- and worst-case which we believe are the best candidates to show the impact of our transformation. If the two extreme error rates satisfy the conditions, then do all the other values because they lie in the range of the two extremes. Using the Python programming language, we find the average EER to be 0.153, that matches the rate as reported in (Killourhy and Maxion, 2009).

3.2.1 Transforming Keystroke Feature Vectors

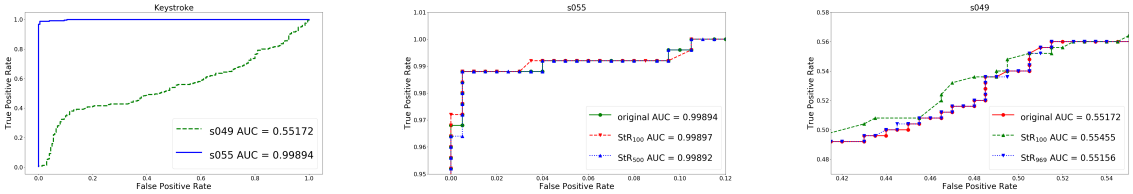
After computing the error rates for each of the 51-subject, our implementation results show that subjects *s055* and *s049* have minimum and maximum EER, respectively. In Table 2, the error rates of both *s055* and *s049* are provided at EER threshold points. In this context the length of the feature vector is 31. We find $t_0 = 1.509$ and $t_0 = 6.718$ as the EER threshold for *s055* and *s049*, respectively. Using the FRR and FAR values at EER threshold, for *s055*, we choose $\bar{\varepsilon} = 0.005$ such that $\mathsf{FAR}'(T_0) \in I_{\mathsf{FAR}} = [\mathsf{FAR}(1.509) - \bar{\varepsilon}, \mathsf{FAR}(1.509) + \bar{\varepsilon}] = [0.007, 0.017]$, and $\mathsf{FRR}'(T_0) \in I_{\mathsf{FRR}} = [\mathsf{FRR}(1.509) - \bar{\varepsilon}, \mathsf{FRR}(1.509) + \bar{\varepsilon}] = [0.005, 0.015]$. Similarly, for *s049*, we choose $\bar{\varepsilon} = 0.02$ such that $\mathsf{FAR}'(T_0) \in I_{\mathsf{FAR}} = [\mathsf{FAR}(6.718) - \bar{\varepsilon}, \mathsf{FAR}(6.718) + \bar{\varepsilon}] = [0.46, 0.50]$, and $\mathsf{FRR}'(T_0) \in I_{\mathsf{FRR}} = [\mathsf{FRR}(6.718) - \bar{\varepsilon}, \mathsf{FRR}(6.718) + \bar{\varepsilon}] = [0.46, 0.50]$.

Following the steps 1 to 4 in Algorithm 1, we find the smallest $\varepsilon = 0.061$ and $\varepsilon = 0.032$ for *s055* and *s049*, respectively, such that FAR' and FRR' lie in the range of the error rates of the corresponding subjects. The step 5 in Algorithm 1 initializes the corresponding s_0 for *s055* and *s049* as $\lfloor 31/0.061 \rceil = 508$, and $\lfloor 31/0.032 \rceil = 969$, respectively.

Next, we need to choose a suitable value for $\mathsf{MinScalar}$. For this, we compute the average feature vector $a = [a_1, ..., a_{31}]$ over all 400 feature vectors of each subject, where a_i is the average of the absolute values of the i'th component of the feature vectors. For *s055*, we find $\min(a) = 0.0184$, $\max(a) = 0.2344$ and $\mathrm{Mean}(a) = \sum a_i/31 = 0.0964$. We choose $s = \mathsf{MinScalar} = 100$, and obtain $\min(\mathsf{StR}_s(a)) = 2$, $\max(\mathsf{StR}_s(a)) = 23$ and $\mathrm{Mean}(\mathsf{StR}_s(a)) = 10$. This ensures that creating a dictionary for the set of transformed feature vectors is an infeasible task for an attacker because $10^{31} \approx 2^{93}$ feature vectors are expected on average. Finally, the Algorithm 1 returns

$$s_0 = 100 \text{ and } T_0 = \lfloor s_0 t_0 \rceil = 151.$$

On the other hand, we find the new system's rates become $\mathsf{FAR}'(151) = 0.012$ and $\mathsf{FRR}'(151) = 0.010$ that are the same as the original system's EER. Please see the two-column MD_s in Table 2 for a complete list of parameters for the new system (integer-valued) derived from the original system (real-valued). As expected, the new system's rates are close to the original system's rates. Similarly, we perform the same operations for *s049* and the results are provided in



(a) ROC curve for *s055* and *s049*. (b) Variation near EER. (c) Variation near EER.

Figure 2: ROC curve, AUC and the curve variation near EER for *s055* and *s049*.

Table 2: The FRR and FAR values for the subjects *s055* and *s049* by computing the MD using both real- and integer-valued feature vectors and using the EER threshold as point of reference.

		Method		
		MD	MD_s	
EER	*s055*	$t = 1.510$ $FRR = 0.010$ $FAR = 0.012$	$s = 100$ $t = 151$ $FRR' = 0.010$ $FAR' = 0.012$	$s = 500$ $t = 755$ $FRR' = 0.010$ $FAR' = 0.012$
	s049	$t = 6.718$ $FRR = 0.480$ $FAR = 0.480$	$s = 100$ $t = 672$ $FRR' = 0.470$ $FAR' = 0.464$	$s = 969$ $t = 6510$ $FRR' = 0.480$ $FAR' = 0.480$

Table 2. Note that the use of smallest scalar s does not necessarily yield EER threshold in the new system and the result of *s049* depicts such a scenario for $s = 100$. But if one is interested in EER, then any scalar greater than the smallest scalar can be used and Algorithm 1 guarantees that those scalars satisfy the error rates range.

Like the LFW dataset, we provide the ROC curve and AUC for keystroke in Figure 2. Both the ROC curve and AUC of *s055* are much better than *s049* as shown in Figure 2a and we find the AUC of *s055* and *s049* as 0.99894 and 0.55172, respectively. It is evident that all the other subjects' curves and AUCs lie between the two curves in Figure 2a. In order to show the effects of our choice of scalars, we provide the ROC curves near the EER of *s055* and *s049* in Figure 2b and Figure 2c, respectively. In case of *s055*, we have the AUC value of 0.99897 and 0.99892 by using $s = 100$ and $s = 500$, respectively. Similarly, we find the AUC value of 0.55455 and 0.55156 by using $s = 100$ and $s = 969$, respectively. The AUC values of $s = 100$ are greater than $s = 500$ and $s = 969$ for both *s055* and *s049*. As expected, larger scalars preserve the accuracy of the original system better, however, the loss in accuracy is not significant when smaller scalars are used as shown in Figure 2b and Figure 2c.

4 A CASE ANALYSIS

We have so far proposed and analyzed a method for transforming biometric authentication systems based on real-valued feature vectors into biometric authentication systems based on integer-valued feature vectors. This allows real-valued feature vectors to be used as input to some cryptographic algorithms, whence to enhance the security of the matching algorithms while preserving the accuracy rates of the original (non-cryptographic) systems. Our proposed biometric authentication system (Enrollment and Verification Phase) is shown in Figure 3. In the following sections, we make our ideas more concrete by implementing a previously proposed algorithm NTT-Sec (Karabina and Canpolat, 2016) over the face (Huang et al., 2007) and keystroke-dynamics (Killourhy and Maxion, 2009) datasets. We choose NTT-Sec in our implementation because a second factor (e.g. user password, or public/private key) is not required in the system, and therefore it offers certain advantages over cancelable biometrics or homomorphic encryption methods. NTT-Sec also seems to be more advantageous than some of the known biometric cryptosystems (e.g. fuzzy extractors) because it is highly non-linear, which yields some built-in security against distinguisahbility and reversibility attacks. For more details about the technical details on NTT-Sec and its security analysis, we refer the reader to (Karabina and Canpolat, 2016).

4.1 Modifying NTT-Sec to NTT-Sec-$\mathbb{R}$

NTT-Sec was originaly proposed to work with binary feature vectors. On the other hand, the face and keystroke-dynamics datasets (Huang et al., 2007; Killourhy and Maxion, 2009), consists of real-valued feature vectors. Therefore, we first need to modify NTT-Sec to NTT-Sec-$\mathbb{R}$ so it can handle real-valued vectors.

The original NTT-Sec is based on two algorithms called Proj (project) and Decomp (decompose). The Proj algorithm maps (projects) a length n binary vec-

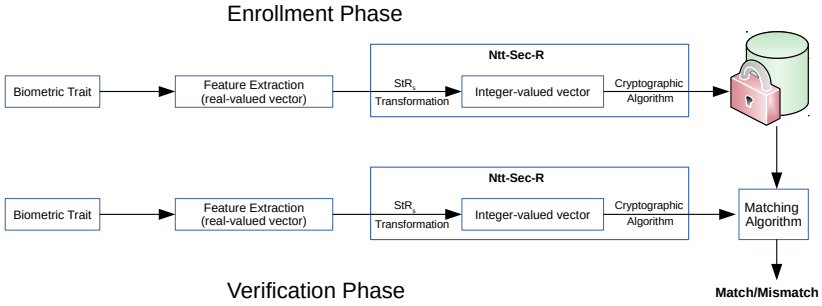


Figure 3: Block Diagram of the proposed system.

tor (considered as the feature vector) to a finite field element (considered as its secure template) using a priori-fixed set of public parameters and a factor basis. Given a pair of secure templates, the Decomp algorithm can detect whether the templates originate from a pair of binary feature vectors that differ in at most t indices for some priori-fixed error threshold value t. In Decomp, the detection is achieved by checking whether a particular finite field element can be written (decomposed) as a product of the factor basis elements in a certain form. Computations in NTT-Sec are performed in a cyclotomic subgroup $\mathbb{G}$ of the multiplicative group of a finite field. We adapt the same group structure in our modification. More specifically, let $\mathbb{F}_q$ be a finite field with q elements where $q = p^m$. Let $c \in \mathbb{F}_q$ be a non-quadratic residue with minimal polynomial of degree m over $\mathbb{F}_p$. Let $\mathbb{F}_{q^2} = \mathbb{F}_q(\sigma)$ be a degree two extension of $\mathbb{F}_q$ where σ is a root of $x^2 - c$. $\mathbb{F}_{q^2}$ has a cyclotomic subgroup $\mathbb{G}$ of order q and every non-identity element in $\mathbb{G}$ can be represented as $\frac{a+\sigma}{a-\sigma}$ for some $a \in \mathbb{F}_q$. Moreover, we say an element $a \in \mathbb{G}$ is *k-decomposable* over $\mathbb{F}_p$ if it can be written as a product $a = \prod_{i=1}^{k} \left(\frac{a_i+\sigma}{a_i-\sigma}\right)$ for some $\mathbb{F}_p$-elements $a_1, a_2, \ldots a_k$.

Modifications. Now, assume that n and t are some fixed values that represent the length of feature vectors and the system threshold value, respectively. We choose a scaling factor s (to be used in StR_s transformation) and let $T = \lfloor st \rceil$ be the new threshold value. As in NTT-Sec, we choose a prime number p such that $p > 2n$, an integer m such that $m \geq T$, a set $\mathsf{B} = \{g_1, g_2, \ldots, g_n\}$ such that $1 \leq g_i \leq \frac{p-1}{2}$ for each i. We further choose m to be prime in order to avoid any potential attacks exploiting subfields. We define new functions NTT-Hash-$\mathbb{R}$ and NTT-Match-$\mathbb{R}$ as a replacement of the original Proj and Decomp functions in NTT-Sec.

The algorithm NTT-Hash-$\mathbb{R}$ maps (or hashes) a given real-valued feature vector $x = (x_1, x_2, \ldots, x_n)$ to a group element in $\mathbb{G}$ as follows: It first computes $X = (X_1, \ldots, X_n) = \mathsf{StR}_s(x)$ using the StR_s transformation. Then using the *basis* $\mathsf{B} = \{g_1, g_2, \ldots, g_n\}$, it computes the hash value

$$\text{NTT-Hash-}\mathbb{R}(x) = \prod_{i=1}^{n} \left(\frac{g_i+\sigma}{g_i-\sigma}\right)^{X_i}.$$

We note that the output of NTT-Hash-$\mathbb{R}$ serves as the secure template for x. The main difference between the modified NTT-Hash-$\mathbb{R}$ and the original Proj is that NTT-Hash-$\mathbb{R}$ can handle real-valued vectors.

The algorithm NTT-Match-$\mathbb{R}$ works very similar to Decomp. Assume a hash value $h_x = \text{NTT-Hash-}\mathbb{R}(x)$ for some $x = (x_1, \ldots, x_n)$, a real-valued vector $y = (y_1, \ldots, y_n)$, and a positive real number t are given. The goal of NTT-Match-$\mathbb{R}$ is to decide whether $\sum_{i=1} |x_i - y_i| \leq t$ or not. To achieve this goal, the following process is performed: It computes $h_y = \text{NTT-Hash-}\mathbb{R}(y)$ using NTT-Hash-$\mathbb{R}$, and then it decides whether the $\mathbb{G}$-element h/h_y is $\lfloor st \rceil$-decomposable. Furthermore, if the retreived $\mathbb{F}_p$-elements belong to the basis B, NTT-Match-$\mathbb{R}$ returns Match, otherwise Mismatch.

We pack all of these parameters under the set $\mathsf{SP} = \{n, t, s, p, m, \mathsf{B}\}$, and call this as the system parameter set. Note that SP can be made public, and commonly used in the NTT-Hash-$\mathbb{R}$ and NTT-Match-$\mathbb{R}$ algorithms.

4.2 Implementation Results, Security Analysis, and Comparisons

First, we discuss the implementation details of the NTT-Sec-$\mathbb{R}$ algorithm over the LFW dataset (Huang et al., 2007) and the keystroke-dynamics dataset (Killourhy and Maxion, 2009). For the LFW dataset, we align our parameter set with the parameters from Table 1. We set $n = 128$, $t = 5.941$, and $s = 100$. Then, we set $p = 257$ (smallest prime $p \geq 2n$) and $m = 599$ (smallest prime $m \geq T = \lfloor st \rceil = 594$). The parameters for the keystroke-dynamics datasets are chosen similarly, and they are same as in Table 2; see Table 3 for a complete list of the parameters. In Ta-

Table 3: The systems parameters (SP), secure template bit size, and timing results in millisecond (ms) using the NTT-Sec-$\mathbb{R}$ algorithm on face and keystroke public datasets.

		SP					Template size (Bit)	Time (ms)	
		n	t	s	p	m		Hash	Match
Face		128	5.941	100	257	599	5391	68.17	309.07
Keystroke	*s055*	31	1.510	100	67	157	1099	3.68	9.21
	s049	31	6.718	100	67	673	4711	21.65	322.63

ble 3, we also report on the bit size of secure biometric template. Recall that a secure template (the output of NTT-Hash-$\mathbb{R}$) is represented as an $\mathbb{F}_q$ element with $m \times (\lfloor \log_2 p \rfloor + 1)$ bits. Hashing (secure template generation) and matching times are also reported in Table 3. We confirm that NTT-Sec-$\mathbb{R}$ does not alter the accuracy-preserving properties of our construction and our experimental results confirm that the FRR and FAR values of the NTT-Sec-$\mathbb{R}$ algorithm are same as that of the MD when integer-valued feature vectors are used. All the codes are written in C programming language. The results are obtained on an Intel Core i7 $-$ 7700 CPU @ 3.60GHz desktop computer that is running Ubuntu 16.04 LTS. The timings are based on a high level implementation of the algorithms and only the GCC compiler is utilized for optimization using the argument -O3.

A Security Analysis. The security of NTT-Sec-$\mathbb{R}$ should be discussed with respect to the *irreversibility* and *indistinguishability* notions as defined in (Karabina and Canpolat, 2016). They are formally modelled between a challenger and a computationally bounded adversary. For irreversibility, several attacks have been considered in (Karabina and Canpolat, 2016), including guessing attack, brute force attack, and discrete logarithm attack. It was also argued in (Karabina and Canpolat, 2016) that reversing the templates is the best strategy for an adversary in a distingushing attack. Following the security analysis in (Karabina and Canpolat, 2016), we inspect that the best strategy for an adversary to attack our modified NTT-Sec-$\mathbb{R}$ (with respect to both irreversibility and indistinguisahbility notions) is to solve the discrete logarithm problem in the underlying cyclotomic group. We provide some details in the following.

Assuming g is a generator of $\mathbb{G}$, the adversary solves $e := \log_g h$ and $e_i := \log_g g_i$ for each $i = 1, \ldots, n$ using a discrete-logarithm solver. Then the adversary gets an equation

$$e = \sum_{i=1}^{n} e_i X_i \mod p^m$$

since $|\mathbb{G}| = p^m$. Using a Knapsack-solver, the adversary finds a solution $X_1, \ldots, X_n$; and recovers X. Assuming the cost of computing the discrete logarithm of an element in $\mathbb{G}$ is C_{DLP} and the cost of solving the above modular Knapsack problem is C_{Knapsack}, then the total cost is

$$(n+1)C_{\mathsf{DLP}} + C_{\mathsf{Knapsack}}.$$

Discrete logarithms in $\mathbb{F}_{p^{2m}}$ can be computed in a time bounded by $(\max(p,m))^{O(\log_2 m)}$; see Theorem 3 in (Barbulescu et al., 2014). Ignoring the cost C_{Knapsack} of the underlying Knapsack problem, we estimate the cost of this discrete logarithm attack to be $(n+1)(\max(p,m))^{\log_2 m}$. Therefore, based on the values of n, p and m from Table 3, we estimate that cost of discrete logarithm based attacks over the LFW face dataset, the Subject *s055*, and the Subject *s049* are 2^{92}, 2^{58}, and 2^{93}, respectively.

Revocability of Templates. Another important security property is the cancellability/revocability/renewability of templates. NTT-Sec-$\mathbb{R}$ naturally allows this property as follows. Instead of using the same public system parameters for each user, one can make the system parameters user specific. This can be achieved in (at least) two different ways. First, the server can generate a public system parameter set per user. If a user's template is stolen or revoked, then a new set of parameters can be generated for that user, and a new template can be enrolled. This would also provide an extra advantage for the indistinguishability of the templates, because now template spaces ($\mathbb{F}_{q^2}$) become algebraically independent of each other. As a second method, each user can derive his own system parameter set from a secret password or a token. Then the user can generate and enroll his template in the server. At the time of authentication, the user can regenerate the parameter set, compute his template, and send them to the server as part of his query. The server proceeds as before and can authenticate the user. This second approach makes the reversibility and distinguishability problems much harder because now an attacker has to search for p and an *ordered* base elements g_i, $i = 1, ..., n$, which belong to a set of size approximately $(p/2)(p/2-1)\cdots(p/2-(n-1))$.

Comparisons. We provide a comparison between our method and four other relevant and recently proposed methods for face template protection. In (Feng et al., 2010), Feng et al. combine distance preserving dimensionality reduction transformation, a discriminality preserving transformation, and a fuzzy commitment scheme. The method in (Feng et al., 2010) requires assigning a random secret token to each user (namely, a random projection matrix) during enrollment; and users need to provide their token during authentication. Therefore, the scheme in (Feng et al., 2010) can be seen as a two factor authentication scheme.

Table 4: Before/After = No/Using cryptographic algorithm, N/P = Not provided, Enroll = Enrollment, and Auth = Authentication. For the LFW dataset, before and after results show the MD results using the respective scalar recommended by Algorithm 1.

	Dataset	GAR@FAR		EER		Secret		Multifactor Authentication
		Before	After	Before	After	Enroll.	Auth.	
Feng et al. (Feng et al., 2010)	FERET CMU-PIE FRGC	N/P	N/P	21.66% 18.18% 31.75%	3.62% 8.26% 9.13%	Yes	Yes	Yes
Pandey et al. (Pandey et al., 2016)	Extended Yale B CMU-PIE Multi-PIE	N/P	96.49%@0FAR 90.13%@0FAR 97.12%@0FAR	N/P	0.71% 1.14% 0.90%	Yes	No	No
Jindal et al. (Jindal et al., 2019)	FEI CMU-PIE Color FERET	N/P	99.98%@0FAR 99.98%@0FAR 99.24%@0FAR	N/P	0.01% 1.14% 0.38%	Yes	Yes	Yes
Boddeti (Naresh Boddeti, 2018) FaceNet and FHE (2 decimal digits)	LFW IJB-A IJB-B CASIA	94.56%@0.1FAR 45.92%@0.1FAR 48.31%@0.1FAR 84.70%@0.1FAR	94.53%@0.1FAR 45.78%@0.1FAR 48.31%@0.1FAR 84.68%@0.1FAR	N/P	N/P	Yes	Yes	No
Our method	LFW	89.99%@0.1%FAR	90.09%@0.1%FAR	3.36%	3.39%	No	No	No

The scheme in (Pandey et al., 2016) uses deep Convolutional Neural Network (CNN) to map face images to maximum entropy binary (MEB) codes. Each user is assigned a unique random MEB code during enrollment, and a deep CNN is used to learn a robust mapping of users' face images to their MEB codes. A cryptographic hash of the MEB code is stored on the server side, which represents the secure template of the underlying biometric data. Plain MEB codes are not needed during authentication, because a queried face image goes through the already trained CNN, and the hash of its output is compared to the stored hash value. It is claimed in (Pandey et al., 2016) that even if an attacker knows the CNN parameters of a user, he cannot obtain significant advantage to attack the system.

The method by Jindal et al. in (Jindal et al., 2019) also uses deep CNN similar to the method in (Pandey et al., 2016), but (Jindal et al., 2019) improves the matching performance over (Pandey et al., 2016) by using user specific random projection matrices. Similar to the method in (Feng et al., 2010), each user is assigned a random secret token during enrollment; and users need to provide their token during authentication.

The method in (Naresh Boddeti, 2018) uses fully homomorphic encryption (FHE), where each user has to generate and manage her public and private key pair. In a typical application, a user stores her private key on his device, encrypts her biometric information under her private key, and enrolls this encrypted template through a server. At the time of authentication, the server receives another encrypted template and uses the homomorphic encryption properties to compute the (encrypted) distance between two templates.

Our scheme does not require using user specific secret, or public/private keys. As a result, it can be thought as a single factor authentication scheme. User specific secrets can easily be adopted in our scheme to obtain extra security (e.g., cancelable templates); see our *revocability of templates* discussion in the *security analysis* part above for more details. This would also increase the matching accuracy of the system similar to the improvements gained in earlier work due to use of user-specific secrets.

In summary, our proposed scheme provides reasonable security levels with comparable performance with respect to previously known systems but comes with an advantage of not requiring any user specific secrets or training during enrollment and authentication. Our comparisons are summarized in Table 4.

Finally, we should note that it is challenging to make a global comparison between the matching accuracy of different methods. This is mainly because of the use of different datasets, feature extraction algorithms, and accuracy measures. For example, error rates over LFW are not reported in (Feng et al., 2010; Pandey et al., 2016; Jindal et al., 2019). And the reason for the difference between our error rates and the error rates as reported in (Naresh Boddeti, 2018) over LFW is that (Naresh Boddeti, 2018) uses FaceNet and we use ResNet for feature extraction. We chose ResNet in our implementation because ResNet has been more commonly used for comparisons over LFW, and that ResNet and FaceNet are comparable in terms of their accuracy. It should be clear from the accuracy preserving properties of our construction that deploying FaceNet in our scheme would yield error rates which are arbitrarily close to the rates in (Naresh Boddeti, 2018).

5 CONCLUSION

We presented a method to create secure biometric templates from real-valued feature vectors. We verified our theoretical findings by implementing a recently proposed secure biometric template generation algorithm over face and keystroke public data sets. We expect that our new construction and its explicit accuracy analysis will enable known cryptographic techniques to protect biometric templates at a larger scale.

ACKNOWLEDGEMENTS

This work was supported by the U.S. National Science Foundation (award number 1718109). The statements made herein are solely the responsibility of the authors.

REFERENCES

Banerjee, S. P. and Woodard, D. L. (2012). Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139.

Barbulescu, R., Gaudry, P., Joux, A., and Thomé, E. (2014). A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology – EUROCRYPT 2014*, pages 1–16.

Blanton, M. and Gasti, P. (2011). Secure and efficient protocols for iris and fingerprint identification. *ESORICS'11, European Symposium on Research in Computer Security*, pages 190–209.

Bodo, A. (1994). Method for producing a digital signature with aid of a biometric feature. *German patent DE*, 42(43):908.

Bours, P. and Barghouthi, H. (2009). Continuous authentication using biometric keystroke dynamics. In *The Norwegian Information Security Conference (NISK)*, volume 1, pages 1–12.

Fairhurst, M. and Da Costa-Abreu, M. (2011). Using keystroke dynamics for gender identification in social network environment. In *4th International Conference on Imaging for Crime Detection and Prevention 2011 (ICDP 2011)*, pages 1–6.

Feng, Y. C., Yuen, P. C., and Jain, A. K. (2010). A hybrid approach for generating secure and discriminating face template. *IEEE Transactions on Information Forensics and Security*, 5(1):103–117.

Geitgey, A. (2017). Face recognition. https://github.com/ageitgey/face_recognition, last accessed: May 18, 2020.

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst. http://vis-www.cs.umass.edu/lfw/, last accessed: May 18, 2020.

Jindal, A. K., Chalamala, S. R., and Jami, S. K. (2019). Securing face templates using deep convolutional neural network and random projection. In *IEEE International Conference on Consumer Electronics, ICCE 2019*, pages 1–6.

Kanade, S., Petrovska-Delacrétaz, D., and Dorizzi, B. (2009). Cancelable iris biometrics and using error correcting codes to reduce variability in biometric data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*, pages 120–127.

Karabina, K. and Canpolat, O. (2016). A new cryptographic primitive for noise tolerant template security. *Pattern Recognition Letters*, 80:70 – 75.

Killourhy, K. S. and Maxion, R. A. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. In *IEEE/IFIP International Conference on Dependable Systems & Networks, 2009, DSN'09*, pages 125–134.

King, D. E. (2011). Dlib library. http://dlib.net/, last accessed: May 18, 2020.

King, D. E. (2017). High quality face recognition with deep metric learning. http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html, last accessed: May 18, 2020.

Naresh Boddeti, V. (2018). Secure face matching using fully homomorphic encryption. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–10.

Natgunanathan, I., Mehmood, A., Xiang, Y., Beliakov, G., and Yearwood, J. (2016). Protection of privacy in biometric data. *IEEE Access*, 4:880–892.

Pandey, R. K., Zhou, Y., Kota, B. U., and Govindaraju, V. (2016). Deep secure encoding for face template protection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 77–83.

Rathgeb, C., Breitinger, F., Busch, C., and Baier, H. (2014). On application of bloom filters to iris biometrics. *IET Biometrics*, 3(4):207–218.

Schmidt, G., Soutar, C., and Tomko, G. (1996). Fingerprint controlled public key cryptographic system (1996). *US Patent*, 5541994.

Tuyls, P., Akkermans, A., Kevenaar, T., Schrijen, G.-J., Bazen, A., and Veldhuis, R. (2005). Practical biometric authentication with template protection. *Audio- and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science*, 3546:436–446.