

# A Quasilinear-Time CVP Algorithm for Triangular Lattice Based Fuzzy Extractors and Fuzzy Signatures

Kenta Takahashi\* and Wataru Nakamura†

\*Hitachi, Ltd., Yokohama, Japan

E-mail: kenta.takahashi.bw@hitachi.com

†Hitachi, Ltd., Yokohama, Japan

E-mail: wataru.nakamura.va@hitachi.com

**Abstract— Fuzzy Extractor (FE) and Fuzzy Signature (FS) are useful schemes for generating cryptographic keys from fuzzy data such as biometric features. To implement FE and FS for fuzzy data in a Euclidean space, such as facial feature vectors, several techniques have been proposed that use lattice-based error correction. In these techniques, it is necessary to find the closest vector in the lattice to a given fuzzy data at the time of key reproduction or signing. However, the closest vector problem (CVP) in a general lattice is known to be NP-hard. Therefore, it is necessary to use a special lattice in which an efficient CVP algorithm can be constructed. In this paper, we propose a faster algorithm for CVP in triangular lattices with  $O(n \log n)$ -time compared to the conventional  $O(n^2)$ -time algorithm.**

## I. INTRODUCTION

Fuzzy Extractor (FE) [1] is a general method to extract the same random numbers from fuzzy data within a certain error margin. FE enables the realization of cryptographic systems using biometric information or PUFs (Physically Unclonable Function) [2] as key management mediums. Similarly, Fuzzy Signature (FS) [3] is a digital signature scheme that uses fuzzy data itself as a signing key.

The specific algorithms of FE and FS depend on the metric space to which the fuzzy data belong and the error range to be tolerated. For example, if the fuzzy data is in  $n$ -bit Hamming space and you want to tolerate up to  $t$ -bit errors, there are known FE algorithms using  $n$ -bit error correcting codes that can correct  $t$ -bit errors [1]. With respect to the strength of the keys to be extracted, the ECC should have as many codewords as possible for given  $(n, t)$ . In this sense, it is ideal for the ECC to be a *perfect code*<sup>1</sup>, which divides the  $n$ -bit space completely with Hamming spheres of radius  $t$ .

On the other hand, in biometric authentication methods such as face recognition, features are often extracted as  $n$ -dimensional vectors by machine learning or other methods, and errors are often evaluated as Euclidean distance<sup>2</sup>. Applying FE

and FS here requires an error correction mechanism in Euclidean space, but since  $n$ -dimensional space cannot be divided by  $n$ -dimensional hyperspheres, an approximate division must be used instead. As a technique for this purpose, a method using an  $n$ -dimensional lattice structure has been proposed [6,7,8]. The desirable properties of the lattice to be applied to FE and FS are as follows.

- (1) The lattice gives a dense sphere packing.
- (2) The closest lattice point to a given arbitrary vector can be computed efficiently.

The first property is important to ensure the strength of the key to be extracted. Such lattice structures have been studied for many years as the *sphere packing problem*, but for the densest structures it is an open problem except in  $n = 1, 2, 3, 8$  and  $24$  dimensions. The second requirement is also important to optimize the balance between security and performance. However, the CVP problem in general lattices is known to be NP-hard.

As a lattice structure that satisfies these properties to some extent, Yoneyama et al. proposed the use of triangular lattices and constructed an efficient CVP algorithm [6]. Their algorithm has a computational complexity of  $O(n^2)$ -time for  $n$ -dimensions by exploiting the high symmetry of the triangular lattice.

However, when applied to real biometric systems, this algorithm may take too much time to compute. For example, the typical number of feature dimensions in face recognition is around  $n = 128 \sim 512$ , and when applied to a 1:N authentication system with large  $N$ , the algorithm needs to be faster.

In this paper, we propose a new algorithm that accelerates the existing CVP algorithm for triangular lattices with a computational complexity of  $O(n \log n)$ .

## II. PRELIMINARIES AND RELATED WORKS

### A. Triangular Lattice

is defined by the cosine similarity  $S_c(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \mathbf{x} \cdot \mathbf{y}$ , are often used for face recognition. This is equivalent to using Euclidean distance, since  $\|\mathbf{x} - \mathbf{y}\|^2 = 2(1 - S_c(\mathbf{x}, \mathbf{y}))$ .

<sup>1</sup> Although there are a limited number of  $(n, t)$  pairs for which a perfect code exists.

<sup>2</sup> Feature extraction methods such as CosFace [4] and ArcFace [5], where a feature vector is  $L_2$ -normalized and the 'closeness' between two vectors  $\mathbf{x}, \mathbf{y}$

In an  $n$ -dimensional real Euclidean space  $\mathbb{R}^n$ , take  $n$  independent vectors  $\mathbf{b}^1, \dots, \mathbf{b}^n \in \mathbb{R}^n$  (basis vectors) and let  $B = (\mathbf{b}^1, \dots, \mathbf{b}^n)$  be a matrix (basis matrix). Then the lattice  $L(B)$  is defined as

$$L(B) := \{B\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}.$$

That is,  $L(B)$  is the set of all vectors  $\mathbf{b}^1, \dots, \mathbf{b}^n$  linearly combined with integer coefficients.

A lattice  $L$  is called a triangular lattice if it has a basis  $B = (\mathbf{b}^1, \dots, \mathbf{b}^n)$  satisfying the following conditions.

$$\forall_i \|\mathbf{b}^i\| = c \text{ (constant)}, \quad \forall_{i \neq j} \mathbf{b}^i \cdot \mathbf{b}^j = \frac{c}{2}$$

The above conditions imply that an angle between any two basis vectors is  $\pi/3$  (Figure 1). Triangular lattices have been proven to give the densest sphere-packing in two and three dimensions. It has also been shown experimentally to have relatively good performance in higher dimensions when applied to FE [6].

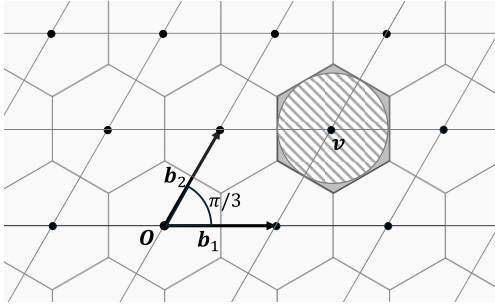


Figure 1: Black dots are the points of a triangular lattice in  $\mathbb{R}^2$  spanned by  $B = (\mathbf{b}^1, \mathbf{b}^2)$ . The gray hexagon represents a Voronoi cell centered at lattice point  $\mathbf{v}$ . The shaded disk represents a sphere with center  $\mathbf{v}$  whose radius is half the minimum distance.

For simplicity, and without loss of generality, the following discussion deals with the triangular lattice with  $c = 1$  and a basis matrix expressed as

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & \beta_1 & \beta_1 & \dots & \beta_1 \\ 0 & \alpha_2 & \beta_2 & \beta_2 & \dots & \beta_2 \\ 0 & 0 & \alpha_3 & \beta_3 & \dots & \beta_3 \\ 0 & 0 & 0 & \alpha_4 & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \beta_{n-1} \\ 0 & 0 & 0 & \dots & 0 & \alpha_n \end{pmatrix} \quad (1)$$

where  $\alpha_k$  and  $\beta_k$  are constants computed by the following simultaneous difference equations:

$$\alpha_k = \sqrt{1 - \sum_{i=1}^{k-1} \beta_i^2}, \quad \beta_k = \frac{\frac{1}{2} - \sum_{i=1}^{k-1} \beta_i^2}{\alpha_k} \quad (k \in [n]), \quad (2)$$

where  $[n]$  denotes the set  $\{1, \dots, n\}$ . The first few values are as follows.

$$\begin{aligned} \alpha_1 &= 1, & \alpha_2 &= \frac{\sqrt{3}}{2}, & \alpha_3 &= \frac{\sqrt{6}}{3}, & \dots \\ \beta_1 &= \frac{1}{2}, & \beta_2 &= \frac{\sqrt{3}}{6}, & \beta_3 &= \frac{\sqrt{6}}{12}, & \dots \end{aligned}$$

### B. Conventional CVP Algorithm for Triangular Lattices

A CVP algorithm takes as input a target vector  $\mathbf{y} \in \mathbb{R}^n$  and outputs the lattice point  $CVP(\mathbf{y})$  closest to  $\mathbf{y}$ . In the following, we outline the conventional CVP algorithm for triangular lattices proposed by Yoneyama et al. See reference [6] for details.

1.  $(x_1, \dots, x_n)^t \leftarrow B^{-1}\mathbf{y}$
2. for  $i = 1, \dots, n$  do
  - $w_i \leftarrow \lfloor x_i \rfloor, r_i \leftarrow x_i - w_i$
- $\mathbf{w} \leftarrow (w_1, \dots, w_n)^t, \mathbf{r} \leftarrow (r_1, \dots, r_n)^t$
3.  $(r_{\sigma(1)}, \dots, r_{\sigma(n)}) \leftarrow (r_1, \dots, r_n)$  sorted in descending order.
4. for  $k \leftarrow 0, \dots, n$  do
  - $\mathbf{z}^k \leftarrow (z_1^k, \dots, z_n^k)^t$  s. t.
    - $z_{\sigma(j)}^k = 1 \quad (j \leq k)$
    - $z_{\sigma(j)}^k = 0 \quad (j > k)$
5.  $\mathbf{z} \leftarrow \operatorname{argmin}_{\mathbf{z}^k \in \{z^0, \dots, z^n\}} \|B\mathbf{r} - B\mathbf{z}^k\|$
6. return  $B(\mathbf{w} + \mathbf{z})$

The computational time complexity of each step is  $O(n^2)$  for step 1, 4 and 6,  $O(n)$  for step 2 and  $O(n \log n)$  for step 3. Step 5 appears to take  $O(n^3)$ -time at first glance, but can be computed in  $O(n^2)$ -time by using the following difference equation:

$$B\mathbf{z}^n = 0, \quad B\mathbf{z}^{k-1} = B\mathbf{z}^k + \mathbf{b}_{\sigma(k)}.$$

Therefore, the total time complexity is  $O(n^2)$ .

### III. PROPOSED ALGORITHM

Based on the conventional algorithm, let us consider speeding up steps 1, 4, 5, and 6, which take  $O(n^2)$ -time. Specifically, the coordinate transformation process (product of

basis matrix and vector) in steps 1 and 6 and the closest vector search (enumeration of candidate vectors and distance calculation) in steps 4 and 5 are accelerated as subroutines, respectively.

#### A. Coordinate Transformation Subroutine

Let us consider algorithms for mutual transformation between  $\mathbf{y} = (y_1, \dots, y_n)^t$  (Cartesian coordinate system) and  $\mathbf{x} = (x_1, \dots, x_n)^t$  (triangular lattice basis coordinate system) that satisfies the following relation:

$$\mathbf{y} = B\mathbf{x} \Leftrightarrow \mathbf{x} = B^{-1}\mathbf{y}.$$

In general, the product of a matrix and a vector requires  $O(n^2)$ -time computations, but by utilizing the basis matrix representation in Eq (1), a linear-time algorithm can be constructed as follows.

Defining  $s_k$  ( $k \in [n]$ ) as

$$s_k := \sum_{i=k+1}^n x_i,$$

we have

$$s_n = 0, \quad (3)$$

$$s_{k-1} = s_k + x_k \quad (k = 2, \dots, n). \quad (4)$$

Furthermore, from Eq (1) the following equation holds:

$$y_k = \alpha_k x_k + \beta_k s_k \quad (k = 1, \dots, n). \quad (5)$$

Therefore, given  $(x_1, \dots, x_n)$ ,  $(y_1, \dots, y_n)$  can be obtained by solving equation (3) and (4) sequentially in the following order:  $y_n \rightarrow s_{n-1} \rightarrow y_{n-1} \rightarrow s_{n-2} \rightarrow y_{n-2} \rightarrow \dots \rightarrow s_1 \rightarrow y_1$ .

Similarly, given  $(y_1, \dots, y_n)$ , we can obtain  $(x_1, \dots, x_n)$  by solving equation (3) and (4) sequentially in the following order:  $x_n \rightarrow s_{n-1} \rightarrow x_{n-1} \rightarrow s_{n-2} \rightarrow x_{n-2} \rightarrow \dots \rightarrow s_1 \rightarrow x_1$ .

Computational time complexity of this algorithm is  $O(n)$ .

#### B. Closest Vector Search Subroutine

Here we describe an idea to speed up the steps 4 and 5 of the conventional CVP algorithm.

In step 5, if we let  $d_k := \|\mathbf{B}\mathbf{r} - \mathbf{B}\mathbf{z}^k\|^2$ , then

$$\begin{aligned} d_k &= \|\mathbf{B}\mathbf{r}\|^2 + \|\mathbf{B}\mathbf{z}^k\|^2 - 2\mathbf{B}\mathbf{r} \cdot \mathbf{B}\mathbf{z}^k \\ &= \|\mathbf{B}\mathbf{r}\|^2 + \|\mathbf{b}^{\sigma(1)} + \dots + \mathbf{b}^{\sigma(k)}\|^2 - 2(r_1 \mathbf{b}^1 + \dots + r_n \mathbf{b}^n) \\ &\quad \cdot (\mathbf{b}^{\sigma(1)} + \dots + \mathbf{b}^{\sigma(k)}) \end{aligned}$$

$$\begin{aligned} &= \|\mathbf{B}\mathbf{r}\|^2 + \frac{1}{2}k(k+1) \\ &\quad - k(r_1 + \dots + r_n) - (r_{\sigma(1)} + \dots + r_{\sigma(k)}) \\ &= \|\mathbf{B}\mathbf{r}\|^2 + \frac{1}{2}k^2 + k\left(\frac{1}{2} - r_{sum}\right) - \sum_{i=1}^k r_{\sigma(i)}, \end{aligned}$$

where  $r_{sum} := \sum_{i=1}^n r_i$ . Therefore the first and second order differences of  $d_k$  are:

$$\begin{aligned} \Delta_k &:= d_k - d_{k-1} = k - r_{sum} - r_{\sigma(k)}, \\ \Delta_k^2 &:= \Delta_k - \Delta_{k-1} = 1 - (r_{\sigma(k)} - r_{\sigma(k-1)}). \end{aligned}$$

Since  $0 \leq r_i < 1$  ( $i \in [n]$ ) we have  $\Delta_k^2 > 0$ . Therefore, if we regard  $d_k$  as a (discrete) function of  $k$ , it is convex downwards, and the  $k$  that gives the minimum value is:

- i. if  $\Delta_1 \geq 0$  then  $k = 0$ ,
- ii. if  $\Delta_n \leq 0$  then  $k = n$ ,
- iii. otherwise the  $k \in [n-1]$  satisfying  $\Delta_k \leq 0 \leq \Delta_{k+1}$ .

In case iii,  $k$  can be found in  $O(\log n)$  time by a binary search for  $\Delta_k$ . Once  $k$  is obtained, determine  $\mathbf{z} = (z_1, \dots, z_n)^t$  as follows:

$$\begin{aligned} z_{\sigma(j)} &= 1 \quad (\text{for } j \leq k), \\ z_{\sigma(j)} &= 0 \quad (\text{for } j > k). \end{aligned}$$

#### C. Fast CVP Algorithm

The following is the proposed CVP algorithm that uses the subroutine described in sections III.A and III.B to speed up the conventional algorithm.

1.  $(x_1, \dots, x_n)^t \leftarrow B^{-1}\mathbf{y}$  (via *Coordinate Transformation Subroutine* described in III.A)
2. for  $i \leftarrow 1, \dots, n$  do
  - $w_i \leftarrow \lfloor x_i \rfloor, r_i \leftarrow x_i - w_i$
  - $\mathbf{w} \leftarrow (w_1, \dots, w_n)^t, \mathbf{r} \leftarrow (r_1, \dots, r_n)^t$
3.  $(r_{\sigma(1)}, \dots, r_{\sigma(n)}) \leftarrow (r_1, \dots, r_n)$  sorted in descending order.
4.  $r_{sum} \leftarrow \sum_{i=1}^n r_i$
5. If  $1 - r_{sum} - r_{\sigma(1)} \geq 0$  then  $k \leftarrow 0$ ,  
else if  $n - r_{sum} - r_{\sigma(n)} \leq 0$  then  $k \leftarrow n$ ,  
otherwise, determine  $k \in [n-1]$  satisfying the following inequality via binary search:
  - $k - r_{\sigma(k)} \leq r_{sum} \leq k + 1 - r_{\sigma(k+1)}$ .
6.  $\mathbf{z} \leftarrow (z_1, \dots, z_n)^t$  s.t.
  - $z_{\sigma(j)} = 1$  ( $j \leq k$ )
  - $z_{\sigma(j)} = 0$  ( $j > k$ )
7. Return  $B(\mathbf{w} + \mathbf{z})$  (via *Coordinate Transformation*)

Subroutine described in III.A)

#### IV. EVALUATION

##### A. Asymptotic Computational Complexity

Let us evaluate the time complexity of the improved CVP algorithm in III.C. Steps 1 and 7 can be computed in  $O(n)$ -time by using the subroutine in III.A. Step 3 can be computed in  $O(n \log n)$ -time. Step 5 is a subroutine of III.B and can be computed in  $O(\log n)$ -time. Since other steps are  $O(n)$ -time, the complexity of the entire algorithm is  $O(n \log n)$ -time.

##### B. Experimental Evaluation

We implemented the proposed CVP algorithm and the conventional algorithm to evaluate their computation times experimentally.

We randomly selected 10,000 real vectors in the range  $\mathbf{y} \in [-100,100]^n$  as target vectors, calculated  $CVP(\mathbf{y})$  using each algorithm, measured the computation time, and evaluated the average time per run. The results for  $n = 128, 256,$  and  $512$  are shown in Table 1 and Figure 2. The computing environment is as follows: OS: Windows 10 Pro, CPU: Intel(R) Core(TM) i7-8700K @ 3.70GHz, memory 32GB, storage: SSD.

Table 1: Average Computation time of CVP algorithms ( $\mu\text{s}$ )

Algorithm	$n = 128$	$n = 256$	$n = 512$
Conventional	38.2	156	623
Proposed	5.8	12.5	26.2

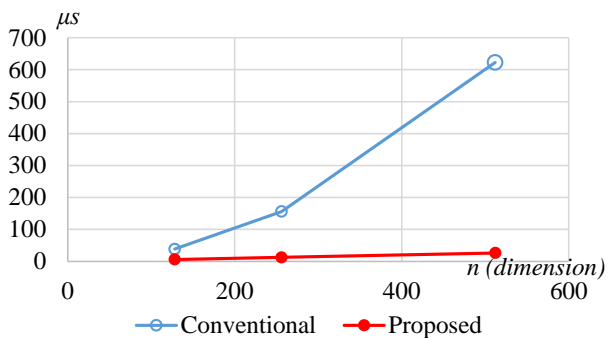


Figure 2: Graphical comparison of the computation time of the algorithms.

The proposed algorithm is about 6.6 times faster than the conventional algorithm when  $n = 128$ , about 12 times faster when  $n = 256$ , and about 24 times faster when  $n = 512$ .

The number of processes per second when  $n = 512$  is  $3.82 \times 10^4$  with the proposed algorithm while it is  $1.60 \times 10^3$  with the conventional algorithm.

#### V. CONCLUSIONS

In this paper, we proposed a fast algorithm for the closest vector problem (CVP) on high-dimensional triangular lattices. The computational complexity of the proposed algorithm is

$O(n \log n)$ , whereas that of conventional algorithms is  $O(n^2)$  for  $n$  dimensions. Experimental evaluation shows that the proposed algorithm achieves a speedup of about 24 times when  $n = 512$ .

The CVP algorithm for high-dimensional triangular lattices is useful for constructing FE and FS using fuzzy data from biometric feature information, such as face recognition. Therefore, the proposed algorithm is expected to enable real-time processing even for large-scale users when constructing template-protected biometric systems and PBI (Public Biometric Infrastructure) [3] using FE and FS. A more detailed evaluation is a subject for future work.

#### REFERENCES

- [1] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin and Adam Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," SIAM journal on computing 38, 1, 97–139. 2008.
- [2] Roel Maes, "Physically unclonable functions: Constructions, Properties and Applications," Springer. ISBN 978-3-642-41395-7, 2013.
- [3] Kenta Takahashi, Takahiro Matsuda, Takao Murakami, Goichiro Hanaoka and Masakatsu Nishigaki, "Signature schemes with a fuzzy private key," International Journal of Information Security 18 (2019), 581–617.
- [4] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li and Wei Liu, "CosFace: Large Margin Cosine Loss for Deep Face Recognition," in Proc. of CVPR, 2018.
- [5] Jiankang Deng, Jia Guo, Niannan Xue and Stefanos Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in Proc. of CVPR, 2019.
- [6] Yuta Yoneyama, Kenta Takahashi and Masakatsu Nishigaki, "Closest Vector Problem on Triangular Lattice and Its Application to Fuzzy Signature," IEICE Trans. Fundamentals (Japanese Edition), Vol.J98-A, No.6, pp.427-435, 2015.
- [7] Kaiyi Zhang, Hongrui Cui and Yu Yu, "Facial Template Protection via Lattice-based Fuzzy Extractors," IACR Cryptology ePrint Arch. 2021: 1559, 2021.
- [8] Shuichi Katsumata, Takahiro Matsuda, Wataru Nakamura, Kazuma Ohara and Kenta Takahashi, "Revisiting Fuzzy Signatures: Towards a More Risk-Free Cryptographic Authentication System based on Biometrics," CCS 2021: 2046-2065, 2021.