Lattice Reduction Using K-Means Algorithm

Shaurya Pratap Singh[1], Brijesh Kumar Chaurasia[2]

shauryapratap2114@gmail.com[1], brijeshchaurasia@ieee.org[2]

[1,2]Department of Computer Science & Engineering,
[1,2]Pranveer Singh Institute of Technology, Kanpur, Uttar Pradesh, India

Abstract: In post-quantum encryption, lattice-based cryptography has become the most powerful and adaptable subfield. Traditional cryptographic systems, including RSA and DSA, are based on the notion that discrete and prime number logarithms are intractable. However, quantum computing threatens to undermine these assumptions. Cryptography based on lattices has grown in popularity as a reliable solution to overcome this obstacle. There are several mathematical problems used in cryptography, including the well-known Shortest Vector Problem (SVP) of lattices. So far, this work focuses on approaches to lattice issues, particularly in two-dimensional and four-dimensional spaces. Unsupervised ML technique K-Means Clustering splits the unlabeled dataset into various clusters. In order to ensure that the data points within each group are more comparable to one another than the data points within the other groups, clustering aims to divide the population into a number of groups. In a nutshell, it's a collection of items with both similarities and differences .K-means to reduce the size of Lattice. K-means machine learning (ML) is used to accomplish this. On datasets that we prepared ourselves, our findings and analyses show a 60% accuracy rate for the strategy we discussed in this paper. A contribution of this study is to enhance lattice-based cryptography's security against emerging threats, particularly in the context of quantum computing.

Keywords: Lattice, Shortest Vector Problem (SVP), Rivest-Shamir-Adleman (RSA), Digital Signature Algorithm (DSA), K-Means Algorithm, Post-Quantum Cryptography (PQC).

1. Introduction

Lattices, a mathematical concept utilized in computer science, play a crucial role in developing novel encryption techniques by addressing problems such as the shortest vector problem (SVP) and closest vector problem (CVP). In Euclidean space, a lattice is a regular arrangement of points [1], [2]. Lattices can be applied to various mathematical and computational tasks, including solving integer programming problems, Diophantine approximation, cryptanalysis, creating integer-based programming problems, and designing error-correcting codes for multiple antenna systems [5]. Recently, the interest in lattice structures has increased significantly.

ICISML 2024, January 05-06, Pune, India

DOI 10.4108/eai.5-1-2024.2342526

The concept of concept decomposition (CD) facilitates the application of cluster analysis as a data reduction technique. Fuzzy K-means (FKM) clustering has been demonstrated as an effective alternative to singular value decomposition (SVD) for reducing the size of term-document matrices in information retrieval (IR) applications, owing to its lower computational complexity [3]. Inspired by the studies in [4], [1], and this paper proposes using K-Means to reduce the size of the lattice.
This paper is organized as follows: Section 2 provides a detailed definition of lattices and concept lattices. Section 3 explores the Shortest Vector Problem (SVP) and its complexity. Section 4 discusses the K-means algorithm's operational mechanics. The proposed methodology is presented in Section 5. Section 6 analyzes the results. Finally, Section 7 concludes the paper and suggests future research directions [23].

2. Lattice

A Lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$, A basis $B \in \mathbb{R}^{n\times n}$, define a; lattice $\mathcal{L}(B)$ in the following way.

An n-dimensional full rank lattice is the set of all integer combinations.

$$\mathcal{L}(B)=\{\sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \cdot for\ 1 \leq i \leq n\}$$

of n linearly independent vector $b_1, \dots b_n\ in\ \mathbb{R}^n$.

i^{th}successive minima (B):

The smallest radius r such that there are i linearly independent vectors $\{v_1 \dots, v_i\}$ of length at most r. (Fig. 1)

Shortest vector: (1, 2)

$$\lambda_1=\sqrt{5}$$

Shortest Basis: $\begin{matrix} 3 & 1 \\ 1 & 2 \end{matrix}$

$$\lambda_2=\sqrt{10}$$

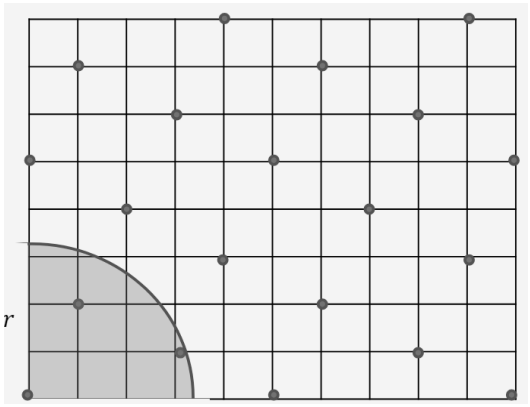


Fig.1: Two Dimensional Lattices

3. Shortest Vector Problem (SVP)

The SVP is focused on locating the shortest non-zero vector in lattice $\mathcal{L}$ [7], as shown in Fig. 2. In order to preserve generality, we restrict the exact form of SVP to integer lattices (and thus integral bases) [8], [21].

3.1 Definition of SVP:

Given a basis *B*, find a non-zero vector $\boldsymbol{v} \in \mathcal{L}\ (B)$ whose length is at most $\gamma \cdot \lambda_1\ (\mathcal{L}\ (B))$.

The exact form of SVP has three frequent variants [17], [19]:

- Determine if the instances $\lambda_1\big(\mathcal{L}(B)\big) \leq d$ and $\lambda_1\big(\mathcal{L}(B)\big) > d$ exist given a lattice basis B and a real $d > 0$.
- Calculation: Locate $\lambda_1\big(\mathcal{L}(B)\big)$ given a lattice basis B.
- Look over a (nonzero) $v \in \mathcal{L}(B)$ such that $\vee v \vee = \lambda_1\big(\mathcal{L}(B)\big)$ given a lattice basis B [20].

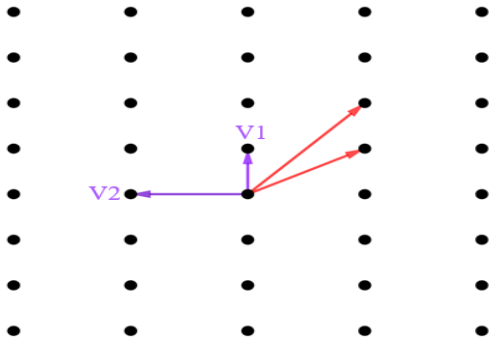


Fig. 2: Shortest Vector v_1 in $\mathcal{L}$

3.2 Hardness of SVP and CVP

Lattice problems, such as the shortest vector problem (SVP) and the closest vector problem (CVP), hold significant importance in number theory and cryptology. Continued fractions are used in number theory to achieve Diophantine approximations, while in cryptography, these problems have recently been employed to develop cryptosystems. The additive and parallelizable nature of lattices makes lattice-based cryptography a promising area of research. Both SVP and CVP are NP-hard to solve exactly and are also NP-hard to approximate within constant factors. Algorithms that find exact solutions to these problems have at least exponential time complexities relative to the lattice dimension. These algorithms also serve as subroutines in strong polynomial-time approximation algorithms, aiding in parameter selection [1], [22], [24], [29].
The shortest vector can be determined using methods such as enumeration [6], sieving [9], [20], or the Voronoi-cell algorithm. Enumeration uses minimal memory, with a running time dependent on the amount and quality of pre-processing. Probabilistic sieving algorithms and the deterministic Voronoi-cell algorithm have exponential time complexities. While enumeration and the Voronoi-cell algorithm can solve CVP, no sieve algorithm has been proven to solve CVP instances effectively. Further research into sieve algorithms that work for CVP would be valuable [24]. Table 1 presents the complexities of currently known SVP and CVP algorithms [16], [18].

Table 1: Complexity of CVP and SVP solutions

Algorithm	Time	Memory	CVP	SVP	Property
Voronoi-cell	2^{2n}	2^{n}	Yes	Yes	Proven
ListSieve-Birthday	$2^{2.465n+O(n)}$	$2^{1.23n+O(n)}$	No	Yes	Proven
GaussSieve	$2^{20.415n+o(n)}$?	$2^{0.2075n+o(n}$	?	Yes	Proven

In the above table question mark(?) is represent the complexity still not clear.

4. *K*-Means Algorithm

K-Means Clustering is an unsupervised machine learning technique that partitions unlabeled datasets into various clusters. The aim of clustering is to ensure that data points within each group are more similar to each other than to those in other groups, essentially forming groups of items based on similarities and differences [9], [10].
K-Means is a popular and scalable algorithm known for its complexity. It is a centroid-based clustering method, where each data point is assigned to a cluster based on its distance from the centroid. A crucial step involves determining the appropriate number of clusters, K, within the

dataset [11]. Data points are iteratively assigned to the nearest cluster, progressively organizing them according to their characteristics. The objective is to minimize the total distance between data points and their respective cluster centroids, ensuring each data point is assigned to the most suitable cluster [12]. This process involves dividing the dataset into K clusters and assigning a mean value to each cluster [26].
Clusters with data points closest to their centroid are selected, and various distance metrics can be used to calculate these distances. Figure 3 illustrates the dataset before and after applying the K-Means algorithm.

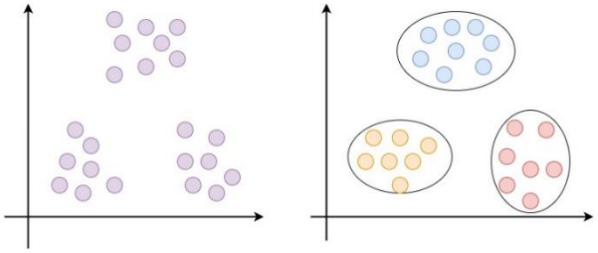


Fig. 3. Clustering of Data

4.1 Flow-chart of K-means Algorithm

The flowchart shown in Fig. 4 represents the k-means clustering process, an unsupervised machine learning algorithm used to group similar data points. The algorithm operates by iteratively assigning data points to the nearest centroid (mean) and updating the centroids after each iteration to reflect the new cluster assignments. The process continues until the cluster assignments stabilize and no longer change [10], [1].
K-means clustering is known for its simplicity and effectiveness. It is easy to implement and applicable to data of any dimensionality. However, the algorithm's sensitivity to the initialization of centroids must be noted, as improper initialization can lead to convergence at a local optimum rather than the global optimum [15]. The detailed explanation of the flowchart is as follows:

1. **Start:** The algorithm starts by initializing the centroids of the clusters. This can be done randomly or using a heuristic method.

2. **Selection of Centroids:** The algorithm then assigns each data point to the cluster with the nearest centroid.

3. **Calculation of distance:** The algorithm then calculates the distance between each data point and the centroid of its assigned cluster.
4. **Minimization of Distance:** The algorithm then checks if the distance between the data point and the centroid of its assigned cluster is the minimum distance between the data point and all centroids. If so, the algorithm moves to the next data point. Otherwise, the algorithm moves the data point to the cluster with the nearest centroid.

5. **Move to the same cluster:** If the algorithm has moved a data point to a new cluster, it updates the centroid of the new cluster to reflect the new cluster assignment.
6. **Repeat the process from step 2:** The algorithm then repeats steps 2-5 until the cluster assignments no longer change.
7. **Stop:** The algorithm terminates when the cluster assignments no longer change [25], [26].

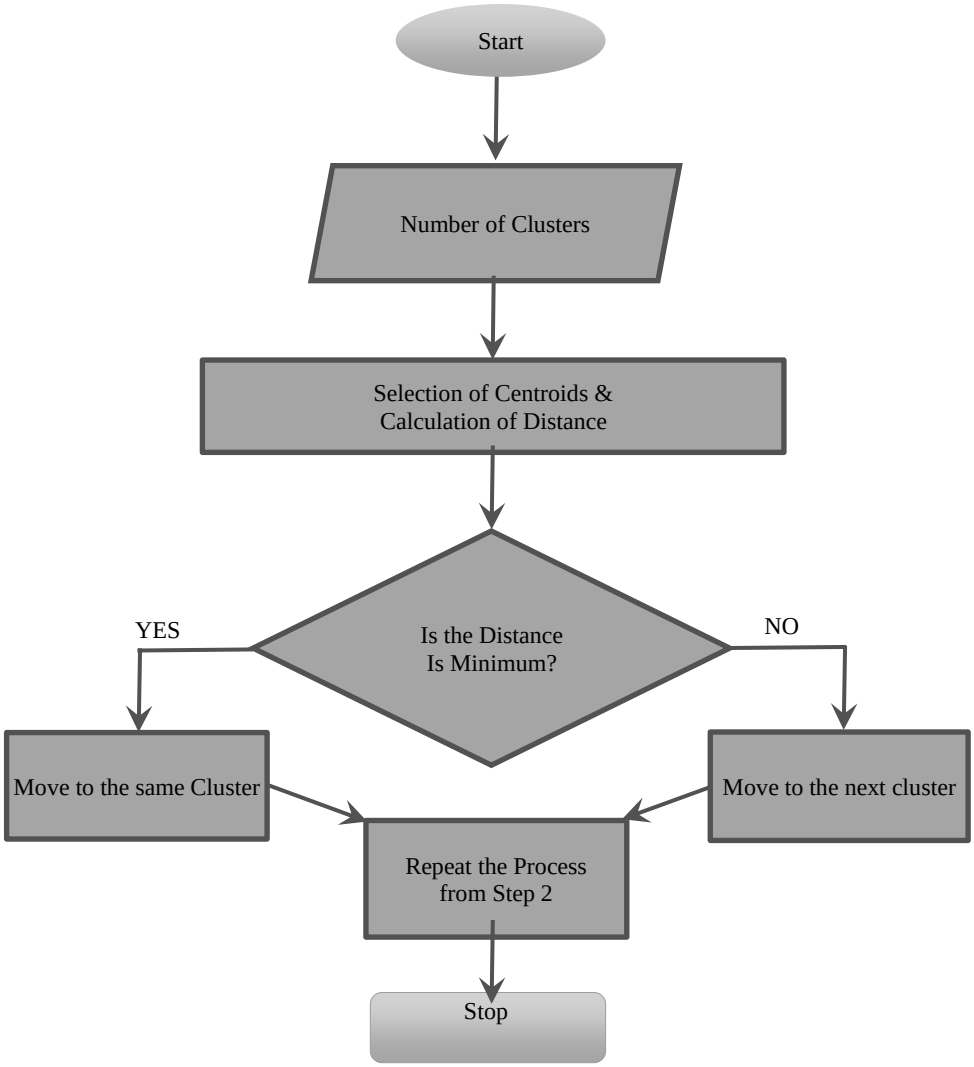


Fig. 4. Flowchart of K-Means Algorithm

4.2 Algorithm and Working of K-means

This sub-section describes the functional process of the suggested method for using machine learning as well as *K*-means to target SVMs. We present a clustering approach for datasets. The dataset is self-generated with 350 points to evaluate the efficacy of the proposed algorithm [1].

4.2.1 *Deciding how many clusters to use:*

Choosing K, the total number of clusters in which the data is going to be arranged, is the initial step. K = 3 will be used in this case.

4.2.2 *Setting up centroid*

The centroid stands for a cluster's focal point. However, since the exact centre of the data points is initially unknown, we choose certain data points at random to serve as the very first centroid for each of the clusters. Fig. 5 (a) initialize three centroid for our dataset.

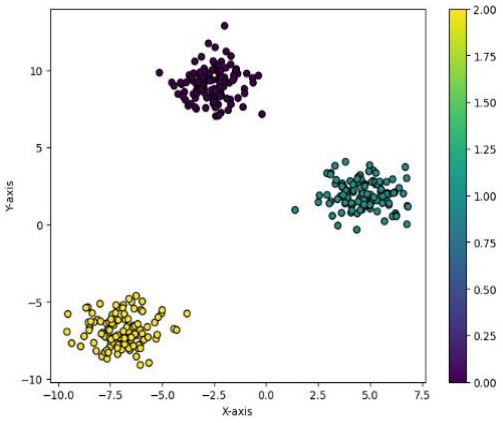


Fig. 5(a). Clustering dataset with K=3

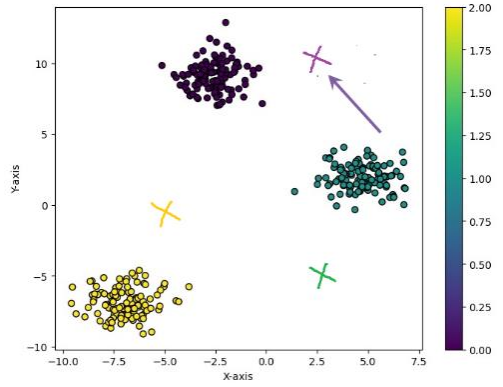


Fig. 5(b). Closest Clustering dataset with K=3

4.2.3 *Align the closest cluster with the data points*

After initializing the centroid, the next step is to connect data points X_n with the nearest cluster centroid C_k as illustrated by Fig. 5 (b). The first objective in this phase is to use the Euclidean distance metrics to calculate the distance between data point X and the centroid.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Choose the data point cluster in which the distance between the data point and the centroid is the shortest.

4.2.4 *Re- Initialize centroid*

By averaging all the data points in that cluster, the centroid will then be reset.

$$C_i = \frac{1}{|N_i|}\sum X_i$$

4.2.5. *Repeat steps three and four*

Phases 3 and 4 will continue to be performed until we obtain the best centroid and reliable data point assignments to the appropriate clusters as represented in Fig. 6.

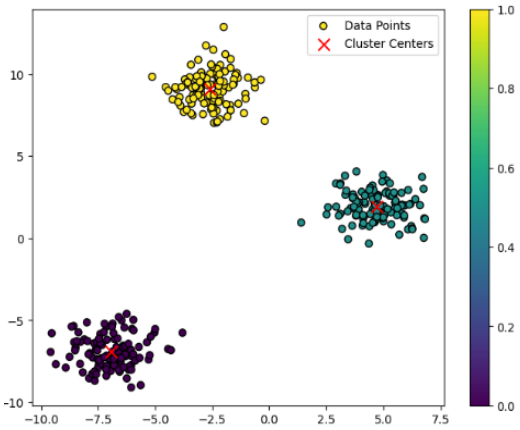


Fig. 6 Clustering dataset with K=3

5. Proposed Methodology

This section introduces a K-Mean-based attack on the Shortest Vector Problem (SVP). SVP is characterized by a large, multi-dimensional lattice that can extend across numerous dimensions. Identifying clusters within a large lattice is crucial. Currently, no algorithm exists that can solve SVP in polynomial time. To the best of my knowledge, this is the first attempt

at applying this approach. The attack was executed on self-generated datasets using lattice attributes.

- **Vector Lattice Property**: Addressing the vector lattice challenge involves a focus on the lattice's inherent vector nature. Lattice vectors can be expressed through coordinate representations, providing a comprehensive understanding of their structural properties. Multidimensional lattice [14] is depicted in Fig. 7, provides a thorough description of the 2- D honeycomb lattice constructed from the vertices of the regular triangles spanning the plane.

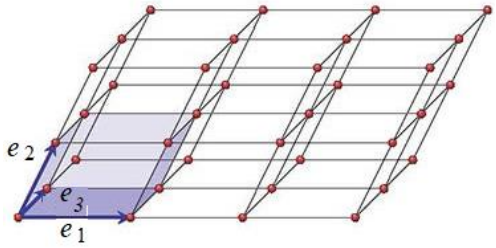


Fig. 7: Multi-dimensional Lattice where e_1, e_2, e_3 are vectors [13]

- **Cryptographic Lattice Size**: As detailed in the 'Basis' section, the finite memory capacity of computers underscores the impracticality of employing an infinitely large lattice. Therefore, the dimensions of the lattice are tailored to the specific capacity constraints of the system in use.

- **K-Mean Algorithm Approach for SVP:** The *K*-Mean algorithm operates on a $2D$ or $4D$ dataset. By inputting the dataset values, the algorithm calculates the silhouette score, enabling the determination of the optimal number of clusters. Subsequently, the algorithm generates clusters based on this determined count, facilitating the creation of a heat-map that visually represents the dataset values.

The following is a full summary of the suggested methodology's steps: As seen in Fig. 8, the procedure begins by providing the entrance points (*X* and *Y*) and the number of groups (*K*). The K-group centers will be chosen in the first step. After that, the procedure moves into an iterative phase that entails processing (such as matching each data point to the closest centroid) and updating (recalculating cluster centroid). A check is done to see how much the cluster's mean has changed after each iteration. The program then goes back to the allocation phase in this situation. If not, the algorithm moves on to the last output stage. Each data point's cluster function and the coordinates of the cluster center are provided in the output's final form.

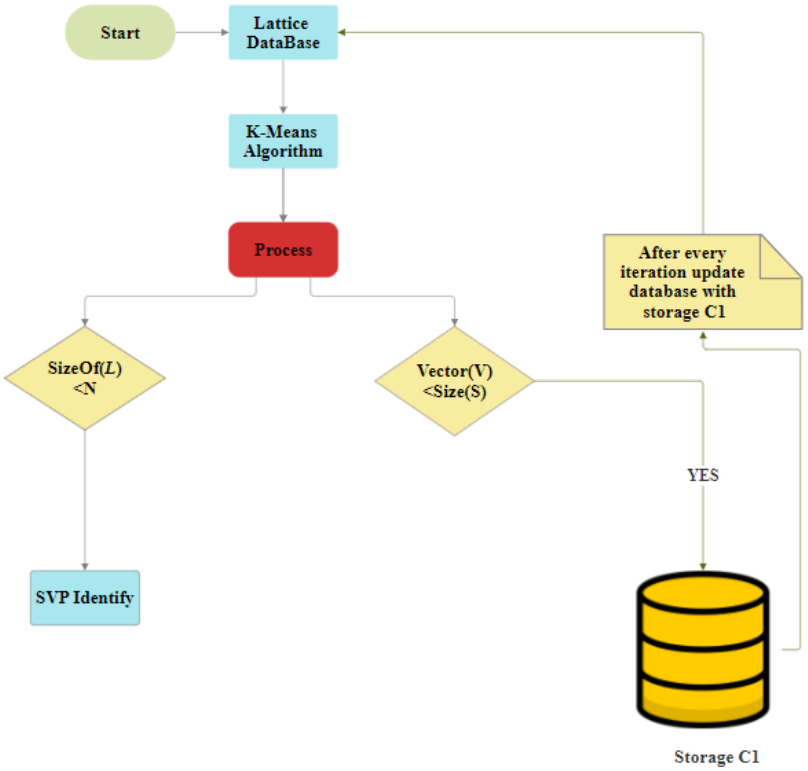


Fig. 8. Working Process of Proposed Methodology

6. Results and Analysis

The K-Means algorithm operates by defining the number of clusters, K. Initially, the centroids are set by shuffling the dataset and then randomly selecting K data points as the centroids. The process continues iteratively until the centroids no longer change. The experimental evaluation of attacks and comparisons among datasets with 2 and 4 clusters, respectively, is as follows:

6.1 Resize 2*D* Lattice using *K*-Mean

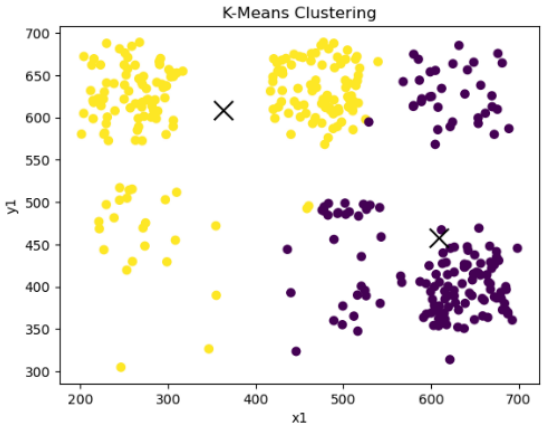
The following is an outline of the analysis of the suggested approach across a 2 *D* lattice datasets:

6.1.1 Datasets of 2 dimensional Lattice

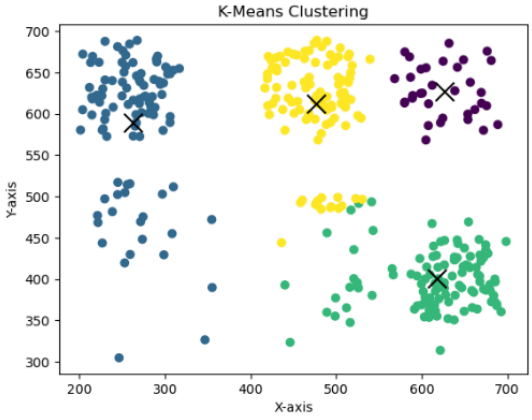
We have generated more than 350 pairs of coordinates to evaluate the efficacy of our proposed scheme. In this we implement *K*-Mean clustering as the no of clusters are given by us and it implements the clusters graph which is given below and according to the dataset values it calculate the silhouette score and the number of clusters we draw it generates its heat-map as per the dataset value.

6.1.2 Graph Representation of Clusters Drawn in $2D$ Dataset

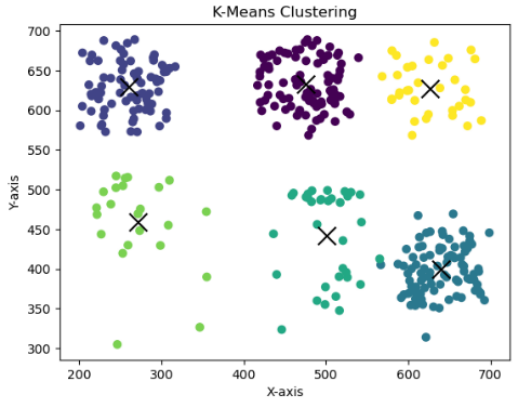
In Fig. 9, there are four graph show that clusters drawn in $2D$ lattice dataset first one is for X_1 highest values is 695.4875 in X and lowest values vary between 250.48-279.97, same in Y shows that highest value is 695.48 and lowest value is vary between 250.48-279.97. The graph is for outcome it shows the value distribution after the execution of proposed methodology.



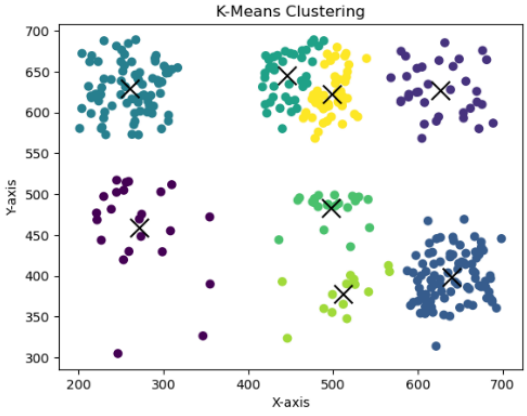
(a)



(b)



(c)



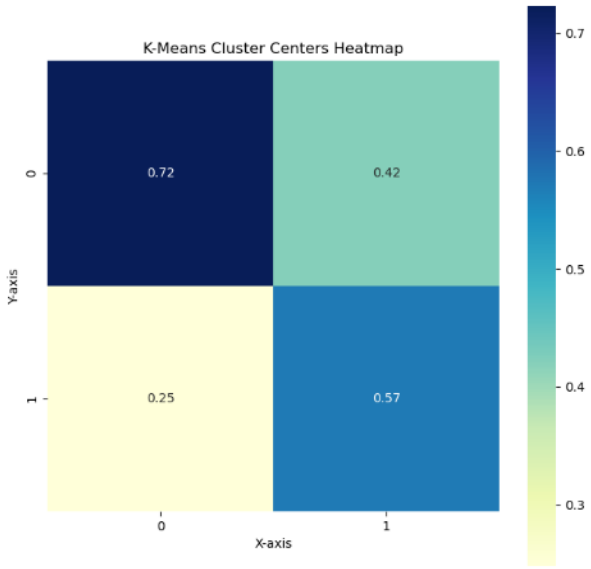
(d)

Fig. 9. Number of clusters drawn in 2D dataset

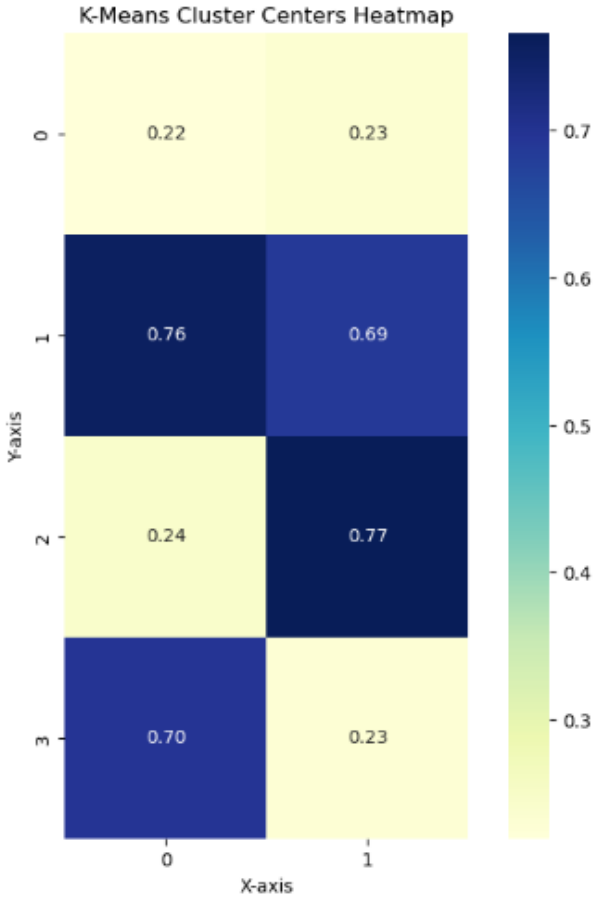
The result shows that cluster sizes are $2D$, $4D$, $6D$, and $8D$ respectively. The accuracy of $8D$ is more, however, at $2D$ accuracy is less.

6.1.3 Heat Map representation of 2 D Lattice

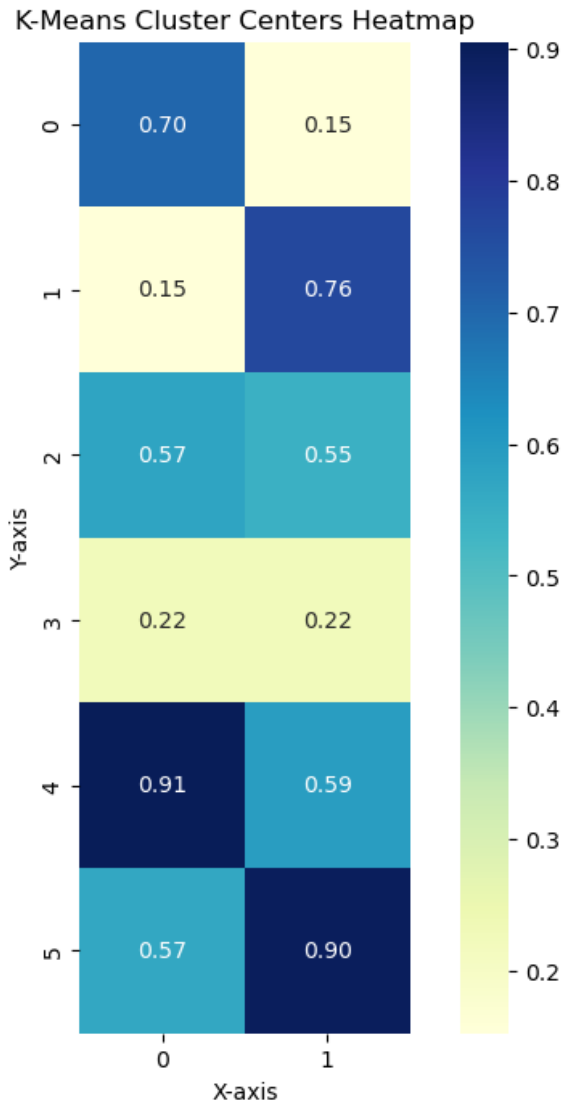
This subsection presents data classification results using a heat map and evaluates performance based on machine learning parameters. Inefficient correlations or interactions between variables in a dataset are illustrated in Fig. 10, with color classes 0 and 1 shown in orange and blue, respectively. The K-Means algorithm is employed to identify clusters among relevant characteristics, distributing the distinctive feature vectors into different classes. Correlated feature points are represented as orange and blue dots, based on the best possible separation bounds determined by K-Means.


K-Means Cluster Centers Heatmap
0.72
0.42
0.25
0.57
0
1
Y-axis
0
1
X-axis
0.7
0.6
0.5
0.4
0.3

(a)

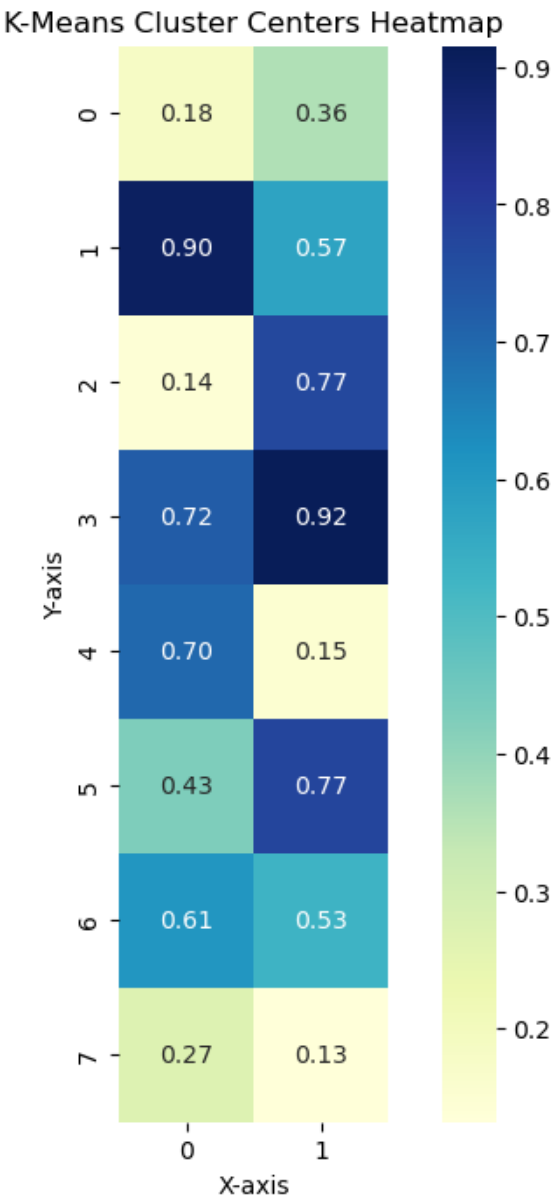

K-Means Cluster Centers Heatmap
0.22
0.23
0.76
0.69
0.24
0.77
0.70
0.23
0
1
2
3
Y-axis
0
1
X-axis
0.7
0.6
0.5
0.4
0.3

(b)



(c)

The heat map shows the number of clusters that are drawn, and the different graph represents that different numbers of clusters are drawn, and each cluster has its own accuracy. According to this, the heat map changes its parameters, and its figure changes at different numbers of clusters. The heat map cannot be the same, and the result can't be the same as compared to the result before we get it.



(d)

Fig.10. Number of clusters drawn in 2*D* dataset

Fig. 11 shows that for different values of *k*, we get different values according to the dataset, and this is the result for 2 *D* dataset values. It is observed that the accuracy of clustering or

attack over SVP at $K = 6$ is high and at $K = 2$ is low. It is observed that the possibility of attack due to reducing lattice size and increasing the accuracy of clustering over 6 D is high up to 60.7%.

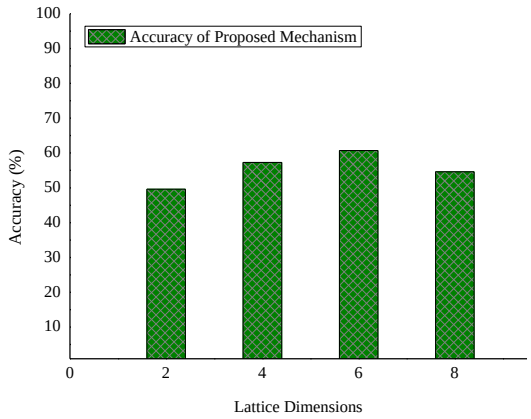


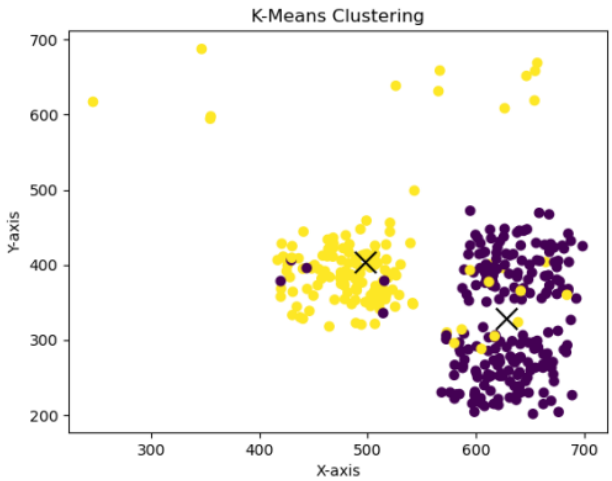
Fig. 11: Performance of proposed K-Mean scheme over 2D Lattice Dataset

6.2 Resize 4-D Lattice using KNN

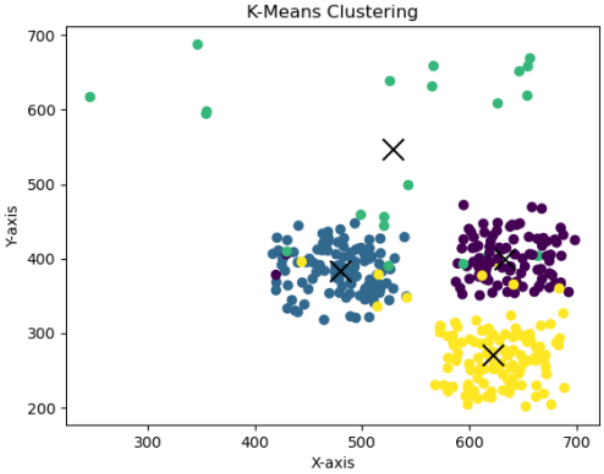
The following is an outline of the analysis of the suggested approach across a 4 D lattice:

6.2.1 Dataset of 4 dimensional Lattice

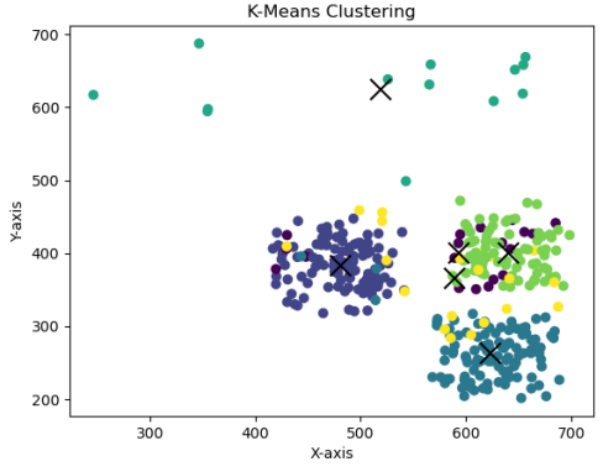
We have generated more than 350 pairs of coordinates to evaluate the efficacy of our proposed scheme. In this, we implement K-Mean clustering as the number of clusters is given by us, and it implements the cluster graph, which is given below, and according to the dataset values, it calculates the silhouette score, and the number of clusters we draw generates its heat map as per the dataset value [27].



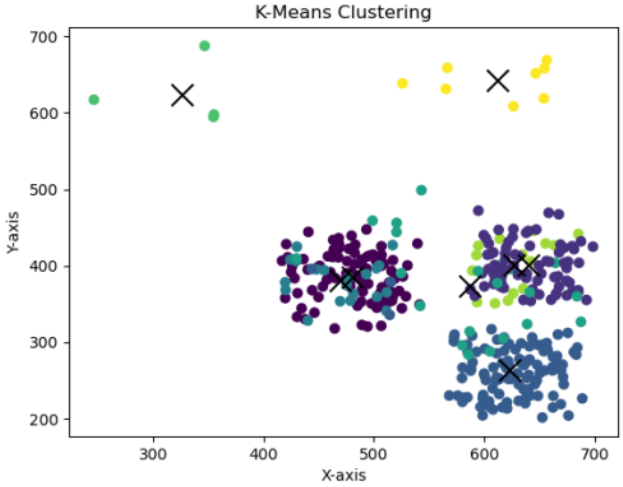
(a)


K-Means Clustering
700
600
500
400
300
200
Y-axis
300
400
500
600
700
X-axis

(b)


K-Means Clustering
700
600
500
400
300
200
Y-axis
300
400
500
600
700
X-axis

(c)



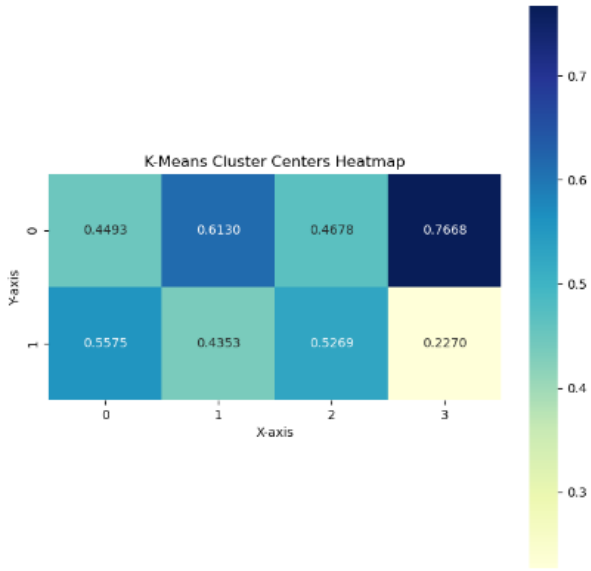
(d)

Fig. 12. Presentation of number of clusters drawn in 2*D* dataset

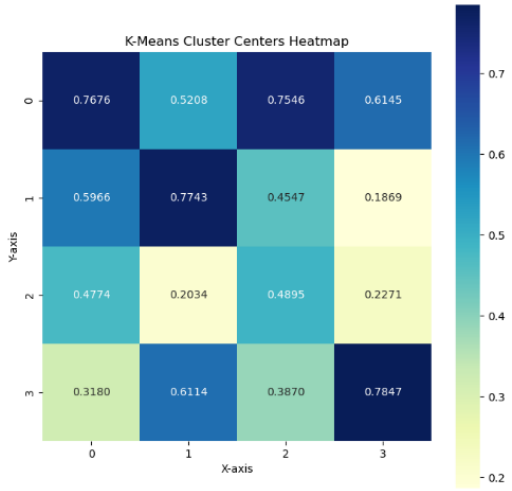
In Fig. 12, nine graphs illustrate the variations in values within a 4-lattice dataset. The first graph shows the highest values, which range between 695.48 and the lowest values, which range between 220.49 and 290.79. Additionally, this graph indicates that the highest value is between 675.48, and the lowest value is between 280.97 and 600.76. The highest value exceeds 695.4875, while the lowest value fluctuates between 250.67 and 275.94. Moreover, the highest value surpasses 695.48, and the lowest value varies between 280.87 and 340.24.

6.2.2 Results of 4-*D* Lattice

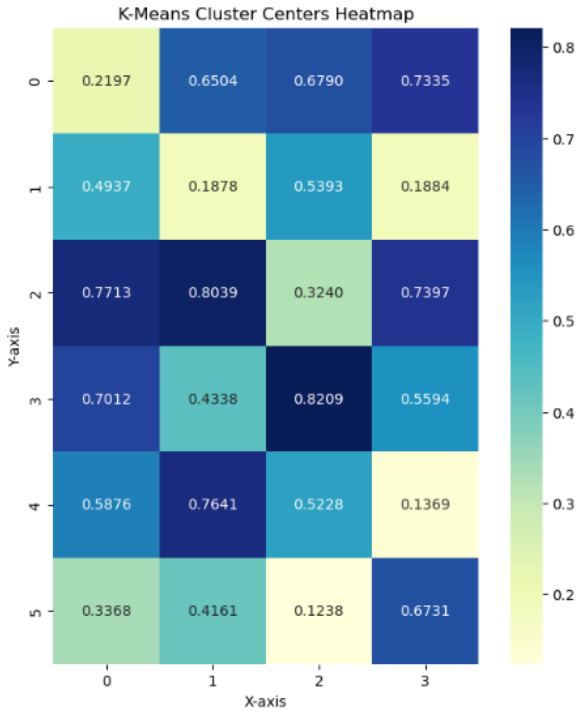
This section presents results through data classification, heat maps, and performance metrics based on machine learning parameters. It highlights the relationships or correlations between variables in a dataset. In Fig. 13, the colors orange and blue represent classes 0 and 1, respectively. The results display the distribution of distinct feature vectors across different classes using the K-Means algorithm, which determines the distance metrics among relevant features. Correlated feature points are depicted as various orange and blue dots with optimal separation boundaries determined by K-Means.


K-Means Cluster Centers Heatmap
0.4493
0.6130
0.4678
0.7668
0.5575
0.4353
0.5269
0.2270
Y-axis
X-axis
0
1
2
3
0.7
0.6
0.5
0.4
0.3

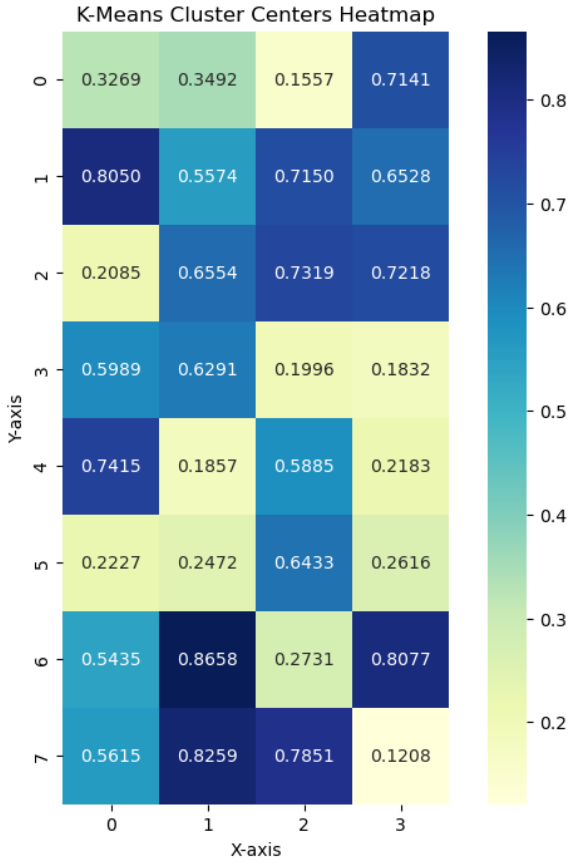
(a)


K-Means Cluster Centers Heatmap
0.7676
0.5208
0.7546
0.6145
0.5966
0.7743
0.4547
0.1869
0.4774
0.2034
0.4895
0.2271
0.3180
0.6114
0.3870
0.7847
Y-axis
X-axis
0
1
2
3
0.7
0.6
0.5
0.4
0.3
0.2

(b)


K-Means Cluster Centers Heatmap
0.2197
0.6504
0.6790
0.7335
0.4937
0.1878
0.5393
0.1884
0.7713
0.8039
0.3240
0.7397
0.7012
0.4338
0.8209
0.5594
0.5876
0.7641
0.5228
0.1369
0.3368
0.4161
0.1238
0.6731
Y-axis
X-axis
0
1
2
3
4
5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

(c)



(d)

Fig. 13. Heat Map representation of 4*D* Lattice Dataset

Fig. 14 shows that over 4 *D*, reducing the size of lattice accuracy is high; however, at 2 *D*, it is high when cluster size is 6. It is also observed that reducing the size of the lattice at cluster size 4 is the same over a 2- *D* and 4-*D* dimensional lattice. In addition, attack over four dimensions is hard and exponentially decreases accuracy at values of K up to 5 to 10. Moreover, over two-dimensional accuracy is ups and downs at values of *K* from 2 to 10 [27], [28].

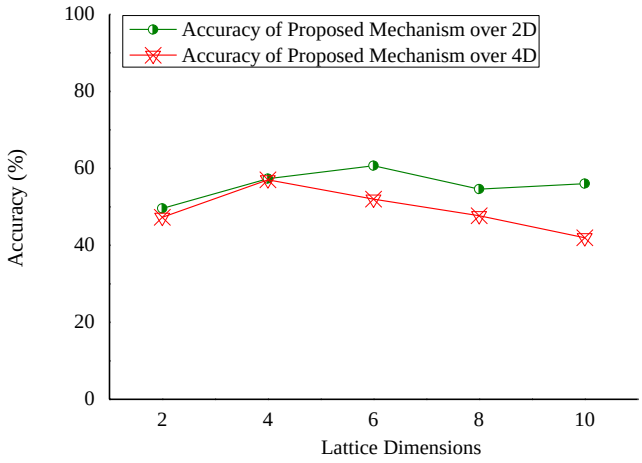


Fig.14. Comparative Analysis of proposed K-Mean scheme over 2 D and 4D Lattice Dataset

7. Conclusion and Future Work

In this work, reducing the size of the lattice using *K*-means clustering is presented. This method has demonstrated its effectiveness in simplifying complicated structures, enabling more effective calculations, and improving the understanding of big datasets. However, over a large dataset, reducing the size of the lattice is hard. The proposed mechanism achieves an accuracy of up to 60%. In the future, we will explore the attack instead of clustering, and KNN will also be applied to explore the possibilities of hybrid approaches over large dataset.

Acknowledgments. We would like to express our gratitude to Dr. Bhupendra Singh, DRDO, Bengaluru, Karnataka, India, for his valuable contributions and insights that enriched this research.

References

[1] Kumar Ch. A. and Srinivas S. Concept lattice reduction using fuzzy *K*-Means clustering. In Expert Systems with Applications 37, 2696–2704 (2010).

[2] Damien S. and Ron S. Making NTRU as secure as worst-case problems over ideal lattices. In EUROCRYPT: Advances in Cryptology – EUROCRYPT 2011, 27–47 (2011).

[3] Dobsa J. and Basic, B.D. Comparison of information retrieval techniques: Latent semantic indexing and concept indexing. In Journal of Information and Organizational Sciences 28, 1–17 (2004).

[4] Cheung, K.S.K. and Vogel, D. Complexity reduction in lattice based information retrieval. Information Retrieval 8, 285–299 (2005).

[5] Forst, M. and Fukshansky, L. Counting Basis Extensions in a Lattice. In Mathematics Subject Classification, 1-15 (2010).

[6] Fukshansky, L. and Shi, Y. Positive semigroups and generalized Frobenius numbers over totally real number fields. In Moscow Journal of Combinatorics and Number Theory 9(1), 29-41 (2020).
[7] Ajtai, M. The shortest vector problem in l2 is np-hard for randomized reductions. In Proc. 13th Annu. ACM Symp. Theory Comput., 10–19 (1998).
[8] Chuang, Yu-L., Fan, Chun-I., Tseng, Yi-F. An Efficient Algorithm for the Shortest Vector Problem. In IEEE Access 6, 61478 – 61487 (2018).
[9] Bennett, H. The Complexity of the Shortest Vector Problem. In Electronic Colloquium on Computational Complexity 170, 1-25 (2022).
[10] Boukhdhir, A., Lachiheb, O., and Gouider, M. S. An improved mapReduce design of kmeans for clustering very large datasets. 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), 1-6 (2015).
[11] Fan, Y., Gongshen, L., Kui, M., Zhaoying, S. Neural Feedback Text Clustering With BiLSTM-CNN-Kmeans. In IEEE Access 6, 57460- 57469 (2018)
[12] Park, H.S., Jun, C.H. A simple and fast algorithm for K-medoids clustering. In Expert Syst. Appl. 36, 3336–3341 (2009).
[13] Daniele Micciancio Lattice Algorithms and Applications CSE 206A Winter 2010 UCSD CSE.
[14] Mundhe, P., Yadav, VK., Verma, S. Venkatesan, S. Efficient Lattice-Based Ring Signature for Message Authentication in VANETs. In IEEE Systems journal, 14(4), 5463 - 5474 (2020).
[15] Helfrich, B. Algorithms to construct Minkowski reduced and hermite reduced lattice bases. In Theor. Comput. Sci., 41, 125–139(1985).
[16] Hendrik, W. and Lenstra, Jr.: Lattices. Algorithmic Number Theory. MSRI Publications, 44, 2008
[17] Chi, D.P., Choi, J. W., Kim, J.S., Kim, T. Lattice Based Cryptography for Beginners. Online available at: https://eprint.iacr.org/2015/938.pdf, Last accessed 23 March 2023.
[18] Zhang, J. and Zhang, Z. Lattice-Based Cryptosystems-A Design Perspective.Springer Singapore, XIII, 174, 2020.
[19] M. Ajtai. The Shortest Vector Problem is NP-Hard for Randomized Reductions, In Proc. 30th Annual ACM Symposium on Theory of Computing (STOC '98), 10–19 (1998).
[20] Ajtai, M. Generating Hard Instances of the Short Basis Problem. In J. Wiedermann et al. (Eds.): ICALP'99, LNCS 1644, Springer-Verlag Berlin Heidelberg, 1−9 (1999).
[21] Bennett, H., Peikert, C. Hardness of the (Approximate) Shortest Vector Problem: A Simple Proof via Reed-Solomon Codes. APPROX/RANDOM, 37:1-37 (2023).
[22] Ajtai, M., Kumar, R., Sivakumar, D. A Sieve Algorithm for the Shortest Lattice Vector Problem. STOC'01, 610-610 (2001).
[23] Micciancio, D. The Shortest Vector in a Lattice is Hard to Approximate to Within Some Constant. In SIAM Journal on Computing, 30(6), 1-26 (2001).
[24] Aggarwal, D., Dadush, D. Regev, O. Solving the Shortest Vector Problem in 2^n Time via Discrete Gaussian Sampling. STOC'15, 733-742 (2015).
[25] Sun, J., Liu, J., Zhao, L. Clustering algorithms Research. In *Journal of Software*, 19(1), 48-61 (2008).
[26] Abdul Nazeer, K.A., Sebastian, M.P. Improving the Accuracy and Efficiency of the k-means Clustering Algorithm. In *Proceeding of the World Congress on Engineering*, 1, 1-5 (2009).
[27] Python online available at: https://www.python.org/, Last accessed on 23 March 2024.

[28] Python online available at: https://www.python.org/, Last accessed on 23 March 2024.
[29] Becker, A., Gama, N., & Joux, A. (2013). Solving shortest and closest vector problems: The decomposition approach. Cryptology ePrint Archive, Paper 2013/685. https://eprint.iacr.org/2013/685