CHAPTER 5

PHASE II-UNINTENTIONAL INSIDER MITIGATION (UIM)

CHAPTER 5

PHASE II - UNINTENTIONAL INSIDER MITIGATION (UIM)

5.1 INTRODUCTION

This chapter discusses the methodology of phase II – Unintentional Insider Mitigation (UIM) which consists of two layers namely Feature Engineering, and Core Behavior Identification. The detected unintentional insiders (UI) in the P&ID (Phase I) are mitigated in this phase to reduce the likelihood of impact using biometric based keystroke dynamics which is one of the user authentication mechanism. UI classifies into genuine and intentional insiders based on behavioral pattern. The keystroke dataset that contains session-based keystroke population with constant keystroke features should require an abstract of intricate keystroke behavior. The Feature Engineering is done to select the best keystroke population from all sessions, convert the keystroke feature into a high dimension, and to obtain the abstract of critical keystroke features. The Core Behavior Identification layer, authenticates and classifies unintentional insiders into genuine, and intentional insiders.

The following are the major contributions of the UIM phase:

- A novel Clonal Kernel Principal Component Analysis (CKPCA) technique has been proposed to obtain the intricate keystroke behavior of unintentional insiders.
- A Deep Belief Network (DBN) has been developed to classify unintentional insiders into genuine and intentional insider with increased detection rate and minimized error rate.

The following section discusses the methodology of the UIM phase elaborately along with the results obtained.

5.2 UIM METHODOLOGY OVERVIEW

The following Figure 5.1 depicts the overall methodology of the UIM phase. The UIM phase consists of two layers namely Feature Engineering and Core Behavior Identification. In Feature Engineering, a novel Feature Engineering technique is proposed to select the best keystroke population that map the population features into high dimensions,

and to obtain an abstract of critical keystroke behavior. In Core Behavior Identification, a deep learning technique is trained to authenticate unintentional insiders and classify them as either genuine or intentional insiders.

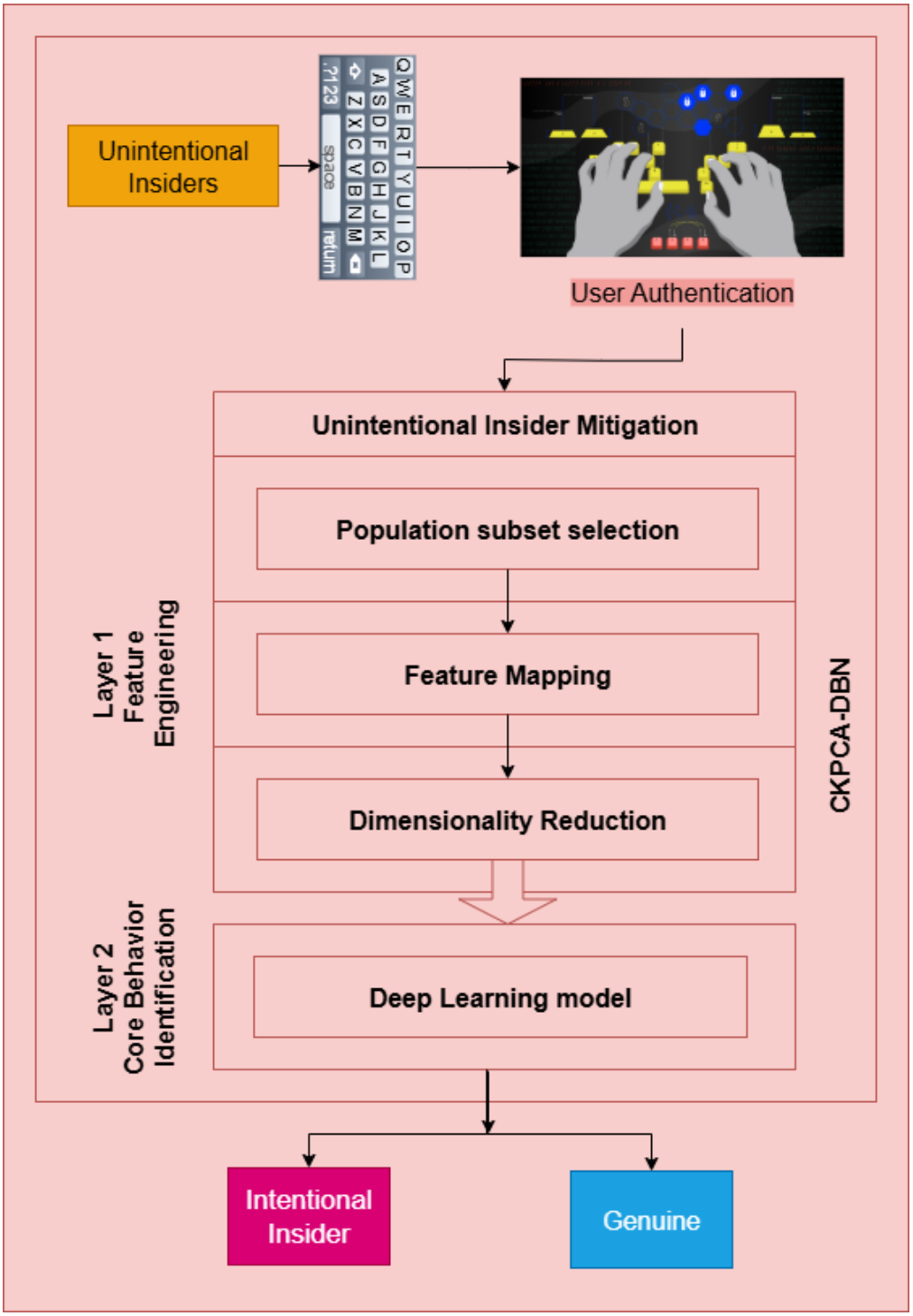


Figure 5.1: Methodology Overview of UIM Phase

5.2.1 DATASET

Two datasets are used for evaluating the methodology of the UIM phase.

1. CMU keystroke dataset for evaluating the methodology of UIM phase (Killourhy & Maxion, 2009).
2. Collected keystroke dataset for validating the methodology of UIM phase

A. CMU Keystroke Dataset

The keystroke behavior is necessary to authenticate the unintentional insider to mitigate them. The benchmark dataset used in this research is generated at Carnegie Mellon University which contains the timely sequence of keystroke behavior for a predefined static password, '.tie5Roanl' (Killourhy & Maxion, 2009). The dataset contains temporal characteristics collected from 51 distinct subjects, each participating in eight sessions over a period. During each session, participants were required to type the predefined password for 50 times, resulting in a total of 400 password samples per subject. In total, this dataset contains 20400 records of keystroke temporal information. Two key typing events for every password character are collected. They are

(i) The time when the keystroke event is pressed is denoted as a Key-Down event and

(ii) The time when the keystroke event is finished is denoted as a Key-Up event.

These events are combined into di-graphs and calculated for each password character. The temporal events of the CMU keystroke dataset are listed in Table 5.1.

Table 5.1: Temporal Events of the CMU Keystroke Dataset

Attribute	**Definition**
Hold Time	It denotes the time interval of holding the individual character during typing.
Flight Time	It denotes the time interval among pressing one character and the subsequent character.
Dwell Time	It denotes the time interval among releasing one character and pressing subsequent character.

[1]

The dataset comprises of 34 parameters that contains insights into the number of sessions and the frequency of password typing within each session to analyze the inter-

session and intra-session variations. The dataset description of the CMU keystroke dataset is listed in table 5.2.

Table 5.2: Dataset Description of CMU Keystroke Dataset

Attribute	**Definition**
subject	It denotes the unique user ID (e.g., 1 to 51)
sessionIndex	It denotes the index of the session in which this keystroke behavior was recorded
rep	It denotes the repetition number of the fixed password in that session
H.period	It denotes the time interval in holding the period (.) character.
DD.period.t	It denotes the pressing time between (.) and (t) characters.
UD.period.t	It denotes the duration among releasing (.) character and pressing (t) character.
H.t	It denotes the time interval in holding (t) character.
DD.t.i	It denotes the pressing time between (t) and (i) characters.
UD.t.i	It denotes the duration among releasing (t) character and pressing (i) character.
H.i	It denotes the time interval in holding (i) character.
DD.i.e	It denotes the pressing time between (i) and (e) characters.
UD.i.e	It denotes the duration among releasing (i) character and pressing (e) character.
H.e	It denotes the time interval in holding (e) character.
DD.e.five	It denotes the pressing time between (e) and (5) characters.
UD.e.five	It denotes the duration among releasing (e) character and pressing (5) character.
H.five	It denotes the time interval in holding (5) character.
DD.five.Shift.r	It denotes the pressing time between (5) character and (Shift + r) combination.
UD.five.Shift.r	It denotes the duration among releasing (5) character and pressing (Shift + r) combination.
H.Shift.r	It denotes the time interval in holding (Shift + r) or (capital R) character.

Attribute	**Definition**
DD.Shift.r.o	It denotes the pressing time between (Shift + r) character and (o) combination.
UD.Shift.r.o	It denotes the duration among releasing (Shift + r) combination and pressing (o) character.
H.o	It denotes the time interval in holding (o) character.
DD.o.a	It denotes the pressing time between (o) and (a) characters.
UD.o.a	It denotes the duration among releasing (o) and pressing (a) characters.
H.a	It denotes the time interval in holding (a) character.
DD.a.n	It denotes the pressing time between (a) and (n) characters.
UD.a.n	It denotes the duration among releasing (a) and pressing (n) characters.
H.n	It denotes the time interval in holding (n) character.
DD.n.l	It denotes the pressing time between (n) and (l) characters.
UD.n.l	It denotes the duration among releasing (n) and pressing (l) characters.
H.l	It denotes the time interval in holding (l) character.
DD.l.Return	It denotes the pressing time between (l) character and (Return) characters.
UD.l.Return	It denotes the duration among releasing (l) and pressing (Return) characters.
H.Return	It denotes the time interval in holding (Return) character.

Since, insider dataset contains only user's log information and mitigation requires keystroke details of unintentional insiders detected in phase I. So, the unintentional insider is associated with random subject ID, and the respective keystroke behavior is used to mitigate the unintentional insider. In this research, the CMU dataset containing temporal characteristics is utilized to evaluate the UIM phase to mitigate unintentional insiders.

B. Collected Keystroke dataset

This dataset comprises of keystroke behavior of users that is collected in a live environment. The dataset comprises a predefined password '.tie5Roanl', captured using a conventional PC keyboard. The dataset comprises of temporal characteristics of keystrokes gathered from 34 subjects, with each subject undergoing eight sessions over a period. During each session, participants typed the password approximately 50 times, resulting in a total of 400 password typing samples per subject. In total, the dataset contains 13600

records of keystroke information where three temporal events such as hold time, dwell time, and flight time are derived from key-down and key-up events. These temporal events are calculated for every character of the password. This consistent and repetitive data collection process ensures the availability of adequate samples to analyze individual keystroke behaviors comprehensively. The temporal events of the collected keystroke dataset is listed in Table 5.3.

Table 5.3: Temporal Events of the Collected Keystroke Dataset

Attribute	**Definition**
Hold Time	It denotes the time between a specific key press and release.
Flight Time	It denotes the time interval among pressing one and subsequent character.
Dwell Time	It denotes the time interval among releasing one character and pressing subsequent character.

The dataset comprises of 29 parameters that contains the three temporal events for each password key to analyze the keystroke behavior from 34 subjects. The dataset description of the Collected keystroke dataset is listed in table 5.4.

Table 5.4: Dataset Description of Collected Keystroke Dataset

Attribute	**Definition**
Name	It denotes the unique user ID (e.g., 1 to 34)
H(.)	It denotes the hold time of period (.) character.
DD(.t)	It denotes the interval among pressing (.) and (t) characters.
UD(.t)	It denotes the interval among pressing (.) and releasing (t) characters.
H(t)	It denotes the hold time of (t) character.
DD(ti)	It denotes the interval among pressing (t) and (i) characters.
UD(ti)	It denotes the interval among pressing (i) and releasing (i) characters.
H(i)	It denotes the hold time of (i) character.
DD(ie)	It denotes the interval among pressing (i) and (e) characters.
UD(ie)	It denotes the interval among pressing (i) and releasing (e) characters.
H(e)	It denotes the hold time of (e) character.
DD(e5)	It denotes the interval among pressing (e) and (5) characters.
UD(e5)	It denotes the interval among pressing (e) and releasing (5) characters.
H(5)	It denotes the hold time of (5) character.
DD(5Ctrl+R)	It denotes the interval among pressing (5) and (Ctrl+R) characters.

Attribute	Definition
UD(5Ctrl+R)	It denotes the interval among pressing (5) and releasing (Ctrl+R).
H(Ctrl+R)	It denotes the hold time of the combination character (Ctrl+R).
DD(Ctrl+Ro)	It denotes the interval among pressing (Ctrl+R) and (o) characters.
UD(Ctrl+Ro)	It denotes the interval among pressing (Ctrl+R) and releasing (o) characters.
H(o)	It denotes the hold time of (o) character.
DD(oa)	It denotes the interval among pressing (o) and (a) characters.
UD(oa)	It denotes the interval among pressing (o) and releasing (a) characters.
H(a)	It denotes the hold time of (a) character.
DD(an)	It denotes the interval among pressing (a) and (n) characters.
UD(an)	It denotes the interval among pressing (a) and releasing (n) characters.
H(n)	It denotes the hold time of (n) character.
DD(nl)	It denotes the interval among pressing (n) and (l) characters.
UD(nl)	It denotes the interval among pressing (n) and releasing (l) characters.
H(l)	It denotes the hold time of (l) character.

Insider dataset contains only user's log information and mitigation requires keystroke details of unintentional insiders detected in phase I. So, the keystroke behavior of random 'Name' is associated for each unintentional insider and used for mitigation. These timing of digraph features are used to capture unique temporal patterns in typing behavior for password '.tie5Roanl'. It is used to evaluate the UIM phase for authenticating the unintentional insiders and classify them into genuine and intentional insiders.

5.2.2 UNINTENTIONAL INSIDER MITIGATION (UIM)

Detected unintentional insiders from the P&ID phase undergo authentication in the UIM phase. The keystroke behavior of unintentional insiders are first processed using the CKPCA technique in Feature Engineering and the outcome is the intricate keystroke behavior. Then, the intricate keystroke behaviors are trained using DBN to identify core behavior for unintentional insider authentication. It classifies unintentional insider into genuine and intentional insider.

The proposed methodology for phase II contains two layers. In the first layer, a Feature Engineering technique namely CKPCA is proposed comprising of population subset selection, feature mapping, and dimensionality reduction to effectively obtain intricate keystroke behavior. In the second layer, the DBN algorithm is trained using intricate

keystroke behavior to identify the core behavior of unintentional insiders and authenticate them to further classify them into genuine and intentional insiders. The working procedure involved in the UIM phase is tabulated in Table 5.5.

Table 5.5: Working Procedure of the UIM Phase

Layer 1- Feature Engineering (CKPCA)
Step 1: The keystroke behavior is processed to identify the best population among them (Population Subset Selection) Step 2: Identify the in-depth variations in the selected keystroke population and map them into high-dimensional structure (Feature Mapping). Step 3: High-dimensional features are reduced to obtain the abstract of engineered features (Dimensionality Reduction).
Layer 2- Core Behavior Identification
Step 4: The feature engineered data is trained using a DBN technique to define the behavioral core of each unintentional insider and authenticate them. It classifies the unintentional insider based on anomalous behavior into a genuine and intentional insider. (Core Behavior Identification)

Layer 1: Feature Engineering

The keystroke behavior of each user requires crucial Feature Engineering to obtain the intricate keystroke behavior containing an abstract of high-dimensional features from the best keystroke population. Figure 5.2 illustrates the overview of the Feature Engineering layer.

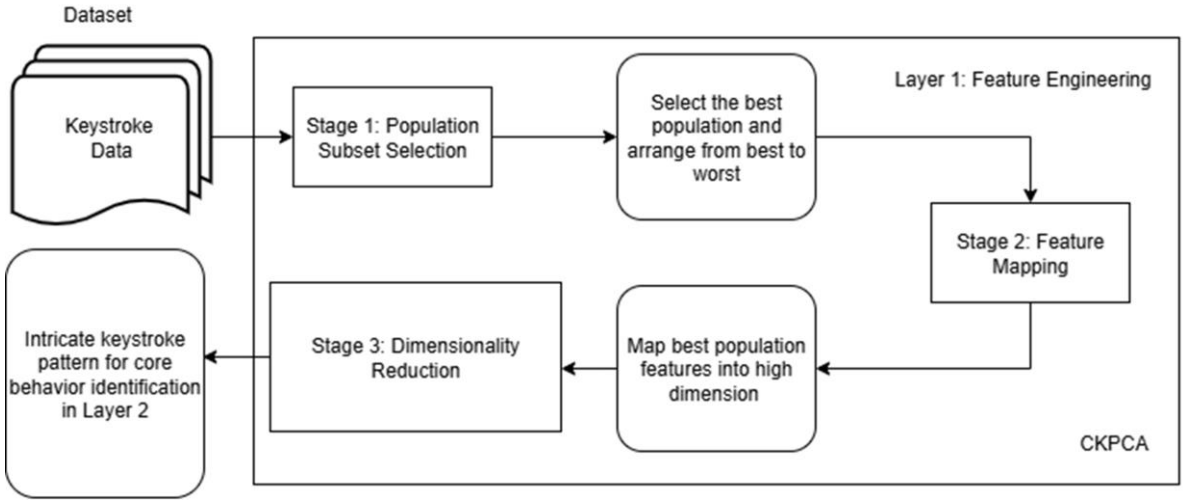


Figure 5.2. Overview of Feature Engineering Layer

The keystroke behavior of each user is processed in population subset selection. The outcome of population subset selection is the arranged keystroke population from best to worst. Arranged keystroke population is mapped using feature mapping to obtain the high dimensional features. These features undergo dimensionality reduction to obtain the abstract of high dimensional intricate keystroke behavior. These intricate keystroke behaviors are trained using a Deep Belief Network to identify the core behavior of each user in layer 2.

The following subsection discusses the proposed CKPCA technique and the three stages involved in CKPCA for Feature Engineering.

A) Clonal Kernel Principal Component Analysis (CKPCA)

The CMU and collected keystroke datasets require effective Feature Engineering to obtain intricate keystroke features while preserving critical information about keystroke behavior. It is addressed by proposing a novel Feature Engineering technique namely Clonal Kernel Principal Component Analysis (CKPCA). CKPCA is a novel hybrid Feature Engineering technique designed to identify the best population, map features to high dimensions and reduce the features while retaining the critical insights. It is achieved by incorporating three stages of the CKPCA technique and illustrated in Figure 5.3: Population Subset Selection, Feature Mapping, and Dimensionality Reduction.

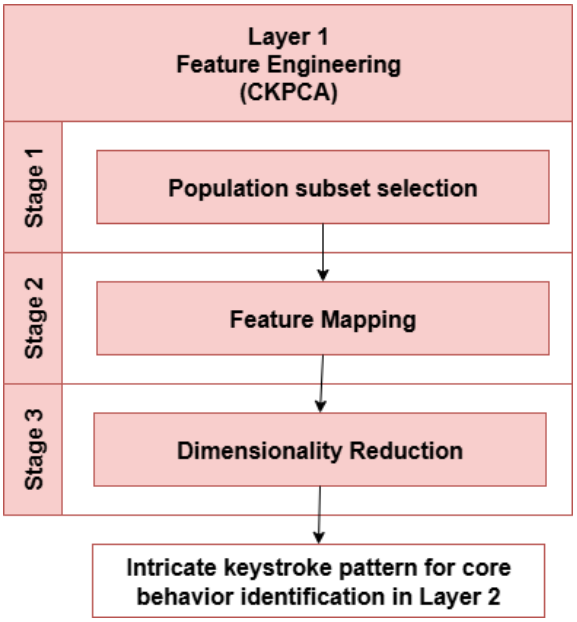


Figure 5.3. Three Stages of CKPCA Technique for Feature Engineering

Population subset selection which arranges keystroke population from best to worst, feature mapping that maps features into high dimensions, and dimensionality reduction which handles overfitting in high dimensional features and obtains the intricate keystroke behavior. The following subsection discusses the three stages of the CKPCA technique for Feature Engineering.

- **Population Subset Selection**

The keystroke dataset consists of inter-session and intra-session keystroke behavior without population arrangement prone to insufficient model training. Population subset selection is required to arrange the population from best to worst where the best population is used for training the DL algorithm. In this phase, the best population of keystroke behavior is identified using a Clonal Selection Algorithm (CSA).

a) Clonal Selection Algorithm (CSA)

CSA applies a nature inspired algorithm containing the principles of the immune system, to recognize diverse antigens effectively. It identifies the most informative subset of the population using the artificial immune system with respect to foreign antigens as an antibody and detects them by stimulating, cloning, and separating them into memory cells and plasma (Nabil et al., 2020). The working of the CSA algorithm for choosing the best population is discussed below.

- Temporal information of the keystroke behavior is utilized to calculate the initial population (P) by segregating into memory (M) and remaining solutions (P_r), $P = P_r + M$ (Population Initialization)
- Selecting n solutions based on affinity measures for identifying population (Pn). Producing clone population (C) using selected n solutions (Cloning and Selection).
- Affinity mutation applying clone population (C) results in generating a matured population C* (Mutation)
- Reselecting the maximal affinity solution from C* is considered as memory (M) and the solutions are swapped between P and C* (Evaluation).
- Replacing minimal affinity solutions (d) with a new set of initial solutions for enhancing population diversity (Iteration).

Population subset selection using CSA involves seven parameters: number of random cells, problem size, selection size, population size, mutation rate, cloning rate, and stop condition. The hyperparameters are considered to increase the performance of CSA through turning. They are population size, cloning rate, and mutation rate. Table 5.6 shows the hyperparameter used for population subset selection using CSA.

Table 5.6: Hyperparameter Used for Population Subset Selection using CSA

Parameter	**Value**	**Definition**
Population Size	50,75,100	It denotes the number of candidate solutions (feature subsets) in each generation.
Cloning Rate	10,20,30,40,50	It denotes the number of clones generated for the best-performing individuals in the population.
Mutation Rate	0.1,0.2,0.3,0.4,0.5	It denotes the probability of introducing random alterations in the cloned solutions to traverse in search space and find the new regions.

It is required to select the best value for hyper-tuning parameters for population subset selection using CSA. The effectiveness of CSA is evaluated using a Random Forest (RF) classifier for population subset generation based on accuracy. Since, it combines multiple decision trees based on ensemble learning technique, it provides robustness to overfitting and results in higher accuracy. The combination of CSA and RF ensures the best parameters are opted for selecting the best population subset. After an extensive grid search and evaluation process, RF obtained 85.53% accuracy based on the best hyperparameter value listed in Table 5.7 for population subset selection using CSA.

Table 5.7: Best Hyperparameter Values for Population Subset Selection using CSA

Parameter	**Value**	**Reason**
Population Size	50	With 50 individuals, CSA balances computational efficiency and maintains diversity in the population
Cloning Rate	10	A cloning rate of 10 is the best solution without over-representing them.
Mutation Rate	0.1	A low mutation rate of 0.1 minimizes the risk of destabilizing high-performing solutions.

From Table 5.7, it is evident that CSA and RF underscore the importance of parameter tuning for optimal population subset selection with 85.53% accuracy. It aims to select the best population of keystrokes for each user based on less cloning and less mutation.

- **Feature Mapping**

After population subset selection, feature mapping is incorporated in this stage. The temporal information of the best population containing keystroke information seems to be less in size. It results in a deep learning model to encounter maximal variability due to inadequate data distribution. Feature mapping is required to produce insightful information relevant to keystroke archetype from the selected population. In this section, the features are represented in high-dimension features using Kernel Mean Embedding (KME).

a) Kernel Mean Embedding (KME)

KME is a technique that applies probability distribution over an implicitly infinite-dimensional population consisting feature vector, equivalent to feature map $\phi(x)$ for procreating kernel Hilbert space (Chatalic et al., 2022). It integrates an estimator to compute the upper bound for a random subset of selected population based on principles of Nystrom approximation. The working of the KME technique for transforming feature vectors in high dimensions is discussed below.

- The ranked population of keystroke behavior is utilized to identify the estimators ($\hat{\mu}_m$) using KME and the equation is expressed in Eqn 9.

$$\hat{\mu}_m = \sum_{1 \le j \le m} \alpha_j \phi\left(\tilde{X}_j\right) \; with \; \alpha = \frac{1}{n} K_m^+ K_{mn} 1_n \quad (9)$$

  Where 1_n denotes the n-dimensional vector of ones,
  K_m^+ denotes the pseudo-inverse of K_m (Moore-Penrose),
  j denotes entries of population,
  $\phi(\tilde{X}_j)$ denotes partial kernel matrices,
  α denotes the Nystrom approximation.

- Kernel functions such as gaussian or polynomial kernel are utilized to create a mapping that transforms probability distribution into a complex, non-linear structure in the data.

- The KME calculates the mean distribution of the features in the higher-dimensional space.

Feature mapping using KME requires two parameters: Gamma, and number of components are listed in Table 5.8.

Table 5.8: Parameter Used for Feature Mapping using KME

Parameter	**Definition**
Gamma	It denotes the gamma parameter for the Radial Basis Function kernel.
Number of components	It denotes the number of features that need to be constructed for feature mapping.

Number of components (k) is a hyperparameter that influences the performance of KME for feature mapping. Table 5.9 shows the hyperparameter used for feature mapping using KME.

Table 5.9: Hyperparameter Used for Feature Mapping using KME

Parameter	**Value**
Number of components (k)	100, 120, 150, 160, 170, 200

It is required to select the best value for hypertuning parameter for feature mapping using KME. The performance of KME is evaluated using an RF classifier for feature mapping based on accuracy. The feature mapping is done using KME to obtain the high-dimensional features containing keystroke behavior. Table 5.10 represents the performance of RF with respect to a number of components for feature mapping.

Table 5.10: Performance Evaluation of KME and RF

k (number of components)	**Accuracy (%)**
100.0	99.87
120.0	99.87
150.0	**99.94**
160.0	99.92
170.0	99.89
200.0	99.92

From the table 5.10, the following observations are made:

- The highest score of 99.94% is achieved at k = 150.
- At k=100 and k=120, the score stabilizes at 99.87% due to an insufficient number of feature components.
- Higher values of k beyond 150 generate redundancy and decline score values to 99.92% and 99.89%.

From the above analysis, the peak performance is observed at k=150. It balances the trade-offs between underfitting (low k) and overfitting (high k). Thus, k=150 is a better parameter that maximizes the performance and ensures computational efficiency. It aims to map the feature from the best population to a high dimension containing 150 features for better performance.

- **Dimensionality Reduction**

After feature mapping, dimensionality reduction is performed in this stage. The ranked population containing high-dimension features of keystroke behavior is prone to the curse of dimensionality that results in overfitting. Dimensionality reduction is required to transmute the high dimensional population of keystroke behavior to the reduced feature space. In this section, the high dimensional features are reduced using Principal Component Analysis (PCA).

a) Principal Component Analysis (PCA)

PCA is a preprocessing technique used to convert high-dimension features into less number of features that simplifies data distribution and interpretation. It maintains high variation within data distribution while converging multi-dimensional feature populations into lower dimensions (Sahu et al., 2022). The working of the PCA technique for reducing high-dimensional feature vectors into low dimensions is discussed below.

- PCA identifies the top principal components that explain the maximum variance in the data (Variance Maximization).
- It ensures the dimensionality reduction without compromising the most critical information.

Dimensionality reduction using PCA requires one parameter: number of principal components. The parameters used for dimensionality reduction using PCA is listed in table 5.11.

Table 5.11: Parameter Used for Dimensionality Reduction using PCA

Parameter	**Definition**
Number of principal components (n_components)	It denotes the number of features to be reduced while retaining the variance.

Number of principal components (n_components) is a hyperparameter that influences the performance of PCA for dimensionality reduction. Table 5.12 shows the hyperparameter used for dimensionality reduction using PCA.

Table 5.12: Hyperparameter Used for Dimensionality Reduction using PCA

Parameter	**Value**
Number of principal components (n_components)	5.0, 8.0, 10.0, 12.0, 14.0, 16.0, 18.0, 20.0

It is required to select the best value for number of principal components for dimensionality reduction using PCA. The performance of PCA is evaluated using the RF classifier for dimensionality reduction based on accuracy. The dimensionality reduction is done using PCA to process the mapped data from KME and project the data into a lower-dimensional subspace while retaining variance. Table 5.13 represents the performance of RF with respect to number of principal components for dimensionality reduction using PCA.

Table 5.13: Performance of RF for Dimensionality Reduction using PCA

n_components (Number of principal components)	**Accuracy (%)**
5.0	95.99
8.0	96.86
10.0	96.65
12.0	97.04
14.0	97.47
16.0	97.70
18.0	**98.72**
20.0	98.49

From Table 5.13, it is observed that:

- When the number of principal components is 18, PCA achieves the highest accuracy of 98.72%.
- A slight decline is encountered in n_components = 20 due to noise and redundant features.
- At n_components = 5, the accuracy is 95.99% is relatively low compared.
- As the number of components increases to 12, the accuracy improves to 97.04%.
- Variance is retained when using 16 components and it obtained 97.70% accuracy.

From the above observations, it is analyzed that tuning the number of principal components in PCA for dimensionality reduction is significant. It is observed that the best performance (98.72% accuracy) is achieved with 18 components in PCA.

All the selected hyperparameters obtained from population subset selection using CSA, feature mapping using KME, and dimensionality reduction using PCA are used in CKPCA for Feature Engineering. Table 5.14 shows the parameter specification of CKPCA.

Table 5.14: Parameter Specification of Proposed CKPCA

Technique	Parameter	Description	Value
CSA	Population size	Number of candidate solutions	50
	Cloning rate	Number of clones generated	10
	Mutation rate	Random alternation in clones	0.1
	Selection size	Selected candidates in solution	10
	Problem size	Parameter for input solution	28
	Number of Random cells	Total number of potential solutions	20
	Stop condition	Number of iteration	20
KME	Gamma	Value for RBF kernel	0.321
	Number of components	Number of increasing features	150
PCA	Number of principal components	Number of features to be reduced	18

The pseudo code of the CKPCA technique for Feature Engineering is discussed below.

a) Pseudo code of CKPCA:

Input: D denotes the keystroke behavior dataset with n samples and m features, K denotes the Kernel function (e.g., RBF, Polynomial), N denotes the population size for CSA, G denotes the number of generations in CSA, C denotes the cloning factor, T denotes the number of top candidates to keep (affinity threshold), d denotes the desired number of principal components

Step 1: Dataset D is used.

Step 2: Initialize the population P of the N candidate subsets.

A subset of samples (or features) from D are contained in each subset $S_i \in P$.

Step 3: For generation = 1 to G do

(a) Affinity of each subset S_i in P is evaluated

Use a kernel-based criterion (e.g., within-class variance, separation margin)

Affinity_i = EvaluateFitness(S_i, K)

(b) Select top T candidates with the highest affinity

P_selected ← top T subsets from P

(c) Clone selected candidates

For each S_i in P_selected:

Generate C clones of S_i with mutation

Apply mutation to introduce diversity (e.g., perturb feature/sample indices)

(d) Evaluate the affinity of all clones and mutated versions

Combine original and cloned candidates

Re-select top N candidates to form a new population P

Step 4: From the final population P, select the best subset S_best with the highest affinity

Step 5: Apply Kernel Mean Embedding (KME) on S_best

Step 6: Compute Kernel Matrix K_matrix for S_best

Step 7: Center Kernel Matrix

Step 8: Apply PCA on K_centered

(a) Compute eigenvalues and eigenvectors of K_centered

(b) Select the top d eigenvectors corresponding to the largest eigenvalues

Step 9: Project data onto principal components

D_reduced ← K_centered × top_d_eigenvectors

Output: D_reduced ← Dimensionality-reduced feature set

Table 5.15 summarizes the advantage of CKPCA over traditional PCA for Feature Engineering and is elaborated in the below section:

Table 5.15: Advantage of CKPCA over Traditional PCA Approach

Specialized Advantage	CKPCA	Traditional PCA Approach
Targeted Population Selection	CKPCA clones and mutates the best population subsets to ensure a more focused exploration	It may focus less on the best population subset selection
Non-Linear Feature Mapping	It transform features into a high number of features based on non-linear relationships among features.	It requires additional modification for non-linear interactions.
Effective Dimensionality Reduction	CKPCA simplifies the model and reduces dimensionality while keeping the most significant variance in the data.	It addresses dimensionality reduction without any additional techniques
Integrated and Specialized Approach	CKPCA handles the specific challenges of biometric data, such as variability and noise, more effectively	It is often more broadly applicable but less specialized
Reduced Risk of Overfitting	It manages overfitting and concentrates on maximizing variance.	A built-in mechanism is included for dimensionality reduction.
Domain-Specific Optimization	CKPCA is tailored specifically for keystroke data, such as variability and temporal aspects.	It requires additional tuning for domain-specific optimization.

In the next layer, the abstract of high dimensional keystroke behavior obtained in this layer from CKPCA is analyzed to identify the core behavior of unintentional insiders. Deep Belief Network is used to authenticate the unintentional insiders and classify them into genuine, and intentional insiders.

Layer-2: Core Behavior Identification

After Feature Engineering, the Core Behavior Identification is done using an abstract of intricate keystroke behavior obtained from PCA for dimensionality reduction. This layer authenticates the unintentional insiders detected in the P&ID phase and mitigates them into genuine and intentional insiders. The following section discusses the Core Behavior Identification for unintentional insiders using the Deep Belief Network.

A) Deep Belief Network (DBN)

The abstract of intricate keystroke behavior obtained using PCA-based dimensionality reduction contains the keystroke population arranged from best to worst for each user. It is utilized for the Core Behavior Identification layer. The unintentional insiders are individuals who carelessly expose the security system to potential risk. These unintentional insiders are detected in the P&ID phase but not mitigated which is crucial. Past research fails to incorporate user authentication mechanism to mitigate the unintentional insiders. Hence, the core behavior of unintentional insiders are detected using the Deep Belief Network which is capable of mitigating unintentional insiders through authentication with minimal error rate.

DBN is a generative model that contains one visible layer (or input layer) along multi-hidden layers where the input layer feeds input samples into Restricted Boltzmann Machine (RBM) based dependent hidden layers (Anakath et al., 2022). However, RBM is a probabilistic model that consists of two sub-layers that includes a visible and hidden layer that shares restricted connectivity to identify hidden data patterns.

The overview of the DBN technique for Core Behavior Identification to mitigate unintentional insiders is illustrated in Figure 5.4.

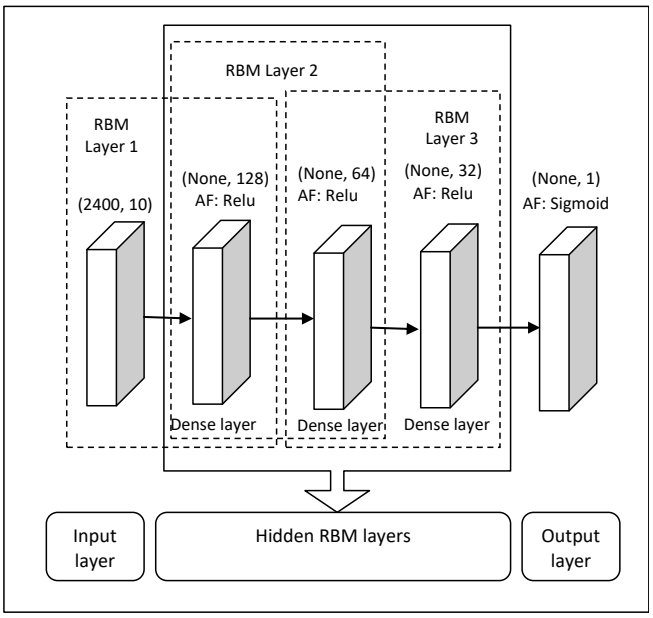


Figure 5.4: Overview of the DBN Technique

Figure 5.4 shows that the DBN technique contains one input and output layer, a hidden layer with tuning parameters. The input layer in DBN analyzes the abstract of intricate keystroke behavior obtained from Feature Engineering as keystroke images. The hidden layer in DBN consists of three stacked RBM layers. It generates a unique pattern containing the abstract and higher-level representation of genuine users' core keystroke behavior. Tuning parameters such as optimizer learning rate, and activation function including 'relu', and 'sigmoid' help optimize multiple RBM layers. In the output layer of DBN, the unintentional insider is authenticated based on their unique pattern of keystroke behavior where it identifies the unintentional insider as genuine, if the authentication is successful. It considers the unintentional insider as an intentional insider, if the authentication fails.

The DBN technique operates in a multi-layer approach for mitigating the unintentional insiders which are discussed below.

1. *Data Splitting:* The initial step involves data splitting in which the abstract of intricate keystroke data are obtained from Feature Engineering. The top 70% of keystroke data contains the best population which is sufficient to identify the core behavior. The intricate keystroke data is split in the ratio of 70:20:10 for training, testing, and validation.

2. *Input Layer:* After Feature Engineering, the input layer of the DBN is employed to receive the raw keystroke behavior and convert it into keystroke images. These images are considered as input for hidden layers.
3. *Hidden Layers Using RBMs*: The keystroke images are processed using hidden layers of the DBN consists of a stack of RBMs. Where each RBM refines the input features into more abstract representations by performing unsupervised learning to uncover intricate patterns and relationships in the data. Each RBM layer involves pre-training and fine tuning.

    - Pre-training based on synaptic weights is done to adjust based on the possibility of data reconstruction specified in eqn.10 to minimize reconstruction errors.

      $$P(h_{j=1}|v) = \sigma\left(b_j + \sum_{i=1}^{n} v_i w_{ij}\right) \quad (10)$$

      Where,

      P denotes activation probability,

      n is number of iterations,

      h_j denotes the hidden unit,

      v denotes every visible unit,

      σ denotes sigmoid function,

      b_j denotes the bias of hidden unit,

      w_{ij} denotes the weight between hidden and visible units.

    - In fine tuning, the weights of all units on the network are optimized by involving back propagation that enhances the performance of the network in prediction.

    In DBN, the hidden layers consist of three RBMs, where each RBM refines the input features into more abstract representations by performing unsupervised learning to uncover intricate patterns and relationships in the data. The working of each RBM is defined below.

- First RBM Layer: processes the raw keystroke images to identify low-level features from the input data.
- Intermediate RBM Layer: Subsequent layers further abstract the data into more compressed and informative feature sets. Non-linear activation functions such as ReLU (Rectified Linear Unit) are used to enhance the learning process by introducing non-linear transformations.
- Final RBM Layer: Encodes a compact representation of the user's behavior, optimizing for core biometric features.

4. *Output Layer:* It classifies unintentional insiders as genuine or intentional insiders based on the learned representation of their keystroke behavior. The output neuron is activated using a sigmoid function, which outputs a probability score indicating user authenticity and detects anomalies in the keystroke patterns. The analysis of decision using the DBN technique is depicted in Table 5.16.

Table 5.16: Analysis of Decision using DBN

User	**DBN Authentication**	**Result**
Unintentional insider	Success	Genuine
Unintentional insider	Fails	Intentional insider

From Table 5.16, it is observed that

- If the authentication of an unintentional insider is successful, then the unintentional insider is considered as Genuine.
- If the authentication fails for an unintentional insider, then the unintentional insider is considered as Intentional insider

5.3 EXPERIMENTAL RESULTS

5.3.1 PERFORMANCE METRICS FOR DBN

The intricate keystroke behavior obtained from Feature Engineering is analyzed using the DBN technique for Core Behavior Identification. The performance of DBN for authenticating unintentional insiders and classifying them into genuine and intentional insiders is evaluated using four performance metrics. The metrics include False Acceptance Rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER), and Accuracy.

FAR is a percentage of incorrectly accepted unintentional insiders. Meanwhile, FRR is a percentage of incorrectly rejected genuine users. The value between FAR and FRR should be maintained for a better authentication system. Table 5.17 illustrates performance metrics for evaluating the performance of DBN for Core Behavior Identification.

Table 5.17. Performance Metrics to Evaluate DBN for Core Behavior Identification.

Performance Metrics	Description	Formula
False Acceptance Rate	FAR is defined as a rate of imprecise detection of unauthorized users.	$False\ Acceptance\ Rate\ (FAR) = \frac{\text{Number of False Positives}}{\text{Total number of verifications}}$
False Rejection Rate	FRR denotes the probability of inaccurately recognizing authorized users.	$False$ Rejection Rate (FRR) $= \frac{\text{Number of False Rejections}}{\text{Total number of verifications}}$
Equal Error Rate	EER defines the intersection of FAR and FRR.	Equal Error Rate (EER) $= \frac{(\text{FAR} + \text{FRR})}{2}$
Accuracy	For every recognition, the total number of precise recognitions of authorized and unauthorized users is considered as accuracy.	Accuracy= $1 - EER$

The present study applies all metrics mentioned above for a significant evaluation.

5.3.2 ELABORATIVE RESULT ANALYSIS OF UIM PHASE

The performance of the UIM phase is evaluated using the CMU keystroke dataset and the collected keystroke dataset. The following subsection discusses the experimental results of the UIM phase containing Feature Engineering in layer 1 and Core Behavior Identification in layer 2.

Layer 1: Feature Engineering

The performance of Feature Engineering is evaluated using the CMU keystroke dataset and the collected keystroke dataset. The following subsection explains the evolution of CKPCA and the results obtained from three stages of CKPCA for Feature Engineering

namely Population subset selection, Feature mapping, and Dimensionality reduction using both datasets.

A. Evolution of CKPCA

This section demonstrates the performance of the proposed CKPCA (Clonal Kernel Principal Component Analysis), to analyze and extract intricate keystroke behavior. It refines and optimizes the extracted features for efficient user authentication.

Population subset selection identifies and retains the best keystroke population and ranks them based on a population score to assess their relevance that distinguishes the pattern of common keystroke behavior. Figure 5.5 shows the results of population subset selection.

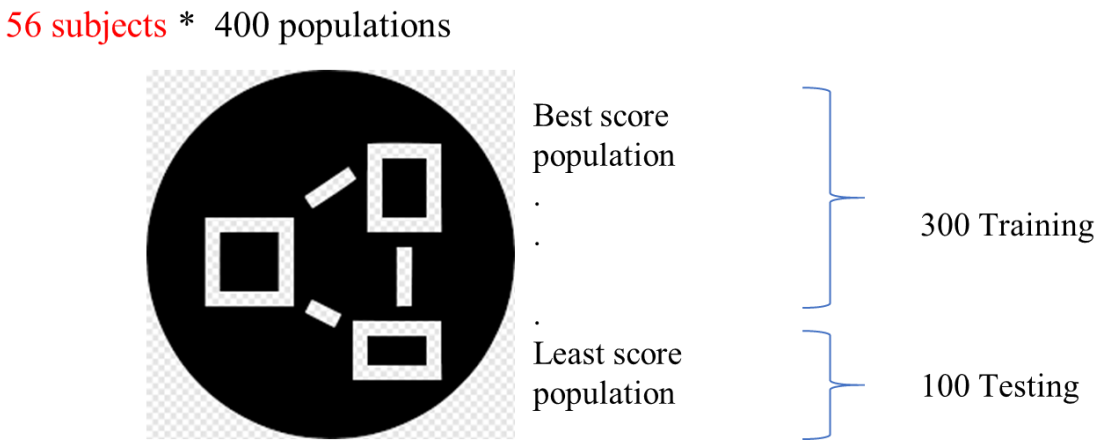
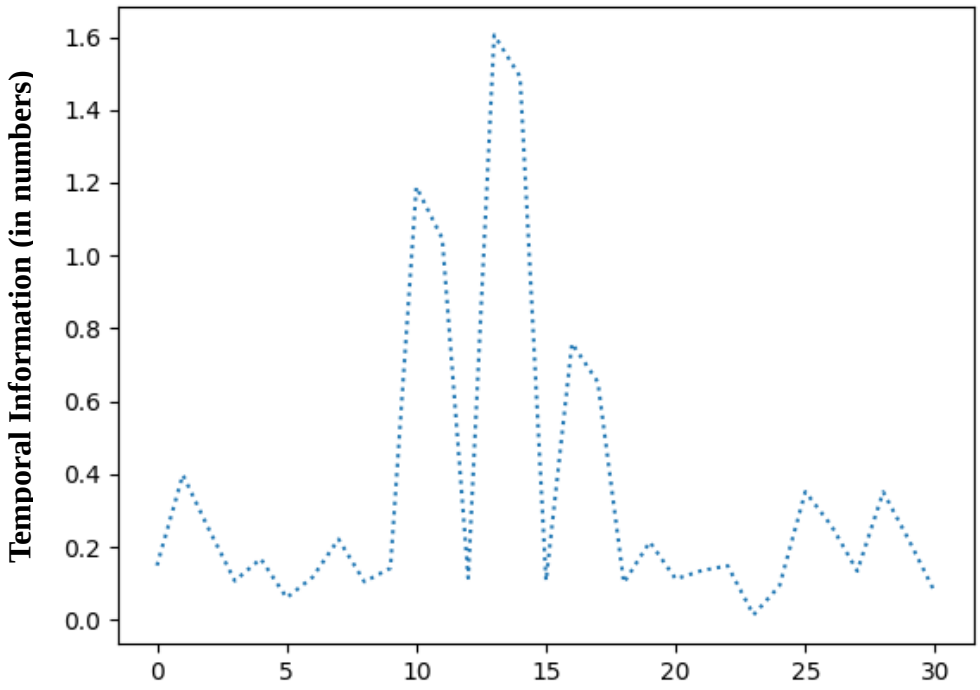


Figure 5.5: Results of Population Subset Selection

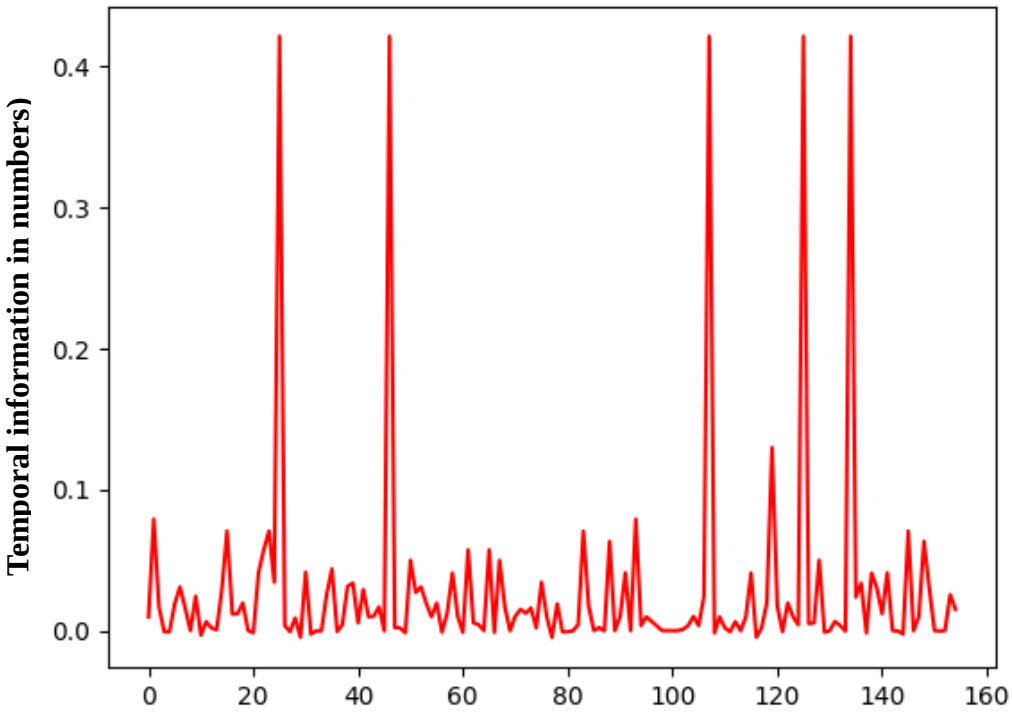
From the figure 5.5, the following observations are made:

- Population is arranged from best to worst, ensuring that only contributing populations are retained for model training.
- The resulting plot shows the train-test split of 400 records that are subjected into 300:100. Where 300 records for training and 100 records for testing. Because, 300 best populations are selected and used for training the user behavior pattern. The outlying 100 records with the lowest population score are used for testing.

After population subset selection, feature mapping is performed. It transforms the keystroke features of the arranged population into a higher-dimensional space for improved representation. The results of feature mapping are illustrated in Figure 5.6.



(a) Selected First Population



(b) After Feature Mapping

Figure 5.6: Results of Feature Mapping

From Figure 5.6, the following observations are made:

- Peaks in the Fig 5.6 (a) represents that the data distribution in selected population subset is normal without huge variance.
- Peaks in Fig 5.6 (b) represents mapped feature values, which highlight higher data distribution with more features of keystroke behavior.
- The distinctive behaviors emerge in the high-dimensional space, improving the model's ability to differentiate between genuine users and imposters.

After feature mapping, the dimensionality reduction is performed. The performance of dimensionality reduction represents reduced dimension, where each principal component (x-axis) is associated with a corresponding abstracted variance (y-axis). Figure 5.7 illustrates the result of dimensionality reduction.

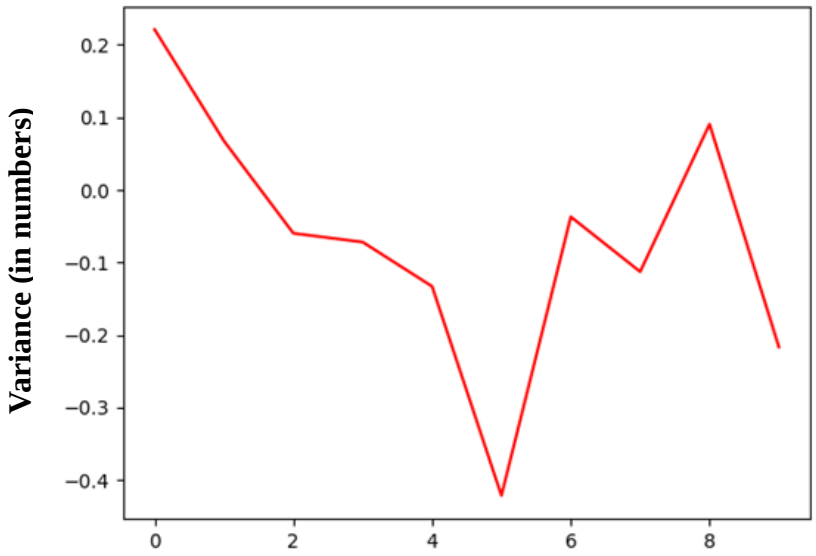


Figure 5.7: Results of Dimensionality Reduction

From Figure 5.7, it is observed that dimensionality reduction captures the essence of high-dimensional features in a compact form, enabling efficient processing for further Core Behavior Identification.

B. Experimental results of CKPCA

The performance of CKPCA is evaluated using two datasets namely the CMU benchmark dataset, and collected keystroke dataset. The following subsection explains the

result obtained from three stages of Feature Engineering namely Population subset selection, Feature mapping, and Dimensionality reduction using both datasets.

i) Population Subset Selection

The result of population subset selection in the CMU keystroke dataset and collected keystroke dataset using Clonal Selection Algorithm is explained below.

The Table 5.18 shows the result of the population subset selection using the Clonal Selection Algorithm in the CMU keystroke dataset. It arranges the keystroke population for 51 subjects from best to worst based on the population affinity value.

Table 5.18. Result of Population Subset Selection using Clonal Selection Algorithm in CMU Keystroke Dataset.

Subject	Worst Population value	Best population value
s002	0.0948	0.21605
s003	0.0784	0.17943
s004	0.1225	0.18507
s005	0.0734	0.23809
s007	0.0744	0.15553
s008	0.0905	0.15852
s010	0.0969	0.14407
s011	0.0776	0.15493
s012	0.1478	0.17642
s013	0.0763	0.14268
s015	0.0895	0.16944
s016	0.1225	0.36054
s017	0.0729	0.15071
s018	0.0863	0.18705
s019	0.0512	0.22177
s020	0.0818	0.23688
s021	0.0808	0.18604
s022	0.056	0.41631
s024	0.0744	0.22815
s025	0.0942	0.22046
s026	0.0787	0.17545
s027	0.0903	0.22414
s028	0.0679	0.18323
s029	0.0834	0.15766
s030	0.0876	0.25483

s031	0.0937	0.24926
s032	0.1029	0.19016
s033	0.0757	0.34322
s034	0.096	0.18212
s035	0.0493	0.20513
s036	0.0414	0.46472
s037	0.1031	0.18317
s038	0.1124	0.25863
s039	0.0731	0.21034
s040	0.1414	0.30947
s041	0.1543	0.24974
s042	0.101	0.17951
s043	0.0657	0.2973
s044	0.0715	0.26954
s046	0.1253	0.27366
s047	0.0797	0.31642
s048	0.1047	0.16091
s023	0.1132	0.57862
s050	0.0966	0.20026
s051	0.1166	0.14445
s009	0.0425	0.20756
s053	0.0939	0.14071
s054	0.1121	0.1651
s055	0.0842	0.11835
s056	0.0894	0.16346
s057	0.1105	0.14281
Average	0.0902	0.22212
Minimum	0.0414	0.11835
Maximum	0.1543	0.57862

From the above table 5.18, the following observations are made:

- The best population value ranges from 0.11835 to 0.57862 contains the best sequence of keystroke population for each user, where the top ranked sequence contains fundamental keystroke pattern of users such as frequent typing signatures.
- Meanwhile, the lowest population value ranging from 0.0414 to 0.1543 contains the low ranked sequence with an irregular keystroke signature which is considered less significant.

The following Table 5.19 shows the result of the population subset selection using the Clonal Selection Algorithm in the collected keystroke dataset. It arranges the keystroke population for 34 subjects from best to worst based on the population affinity value.

Table 5.19. Result of Population Subset Selection using Clonal Selection Algorithm in Collected Keystroke Dataset.

Subject ID	Worst population value	Best population value
7	0.14926	0.30035
8	0.16346	0.30259
9	0.20143	0.34352
10	0.21606	0.36095
11	0.09519	0.22779
12	0.15538	0.2324
13	0.10741	0.24661
14	0.11385	0.24198
15	0.18012	0.38164
16	0.27422	0.39538
17	0.12488	0.26733
18	0.09774	0.25348
19	0.11939	0.30912
20	0.1021	0.29932
21	0.13054	0.29865
22	0.168	0.31498
23	0.18275	0.35934
24	0.19534	0.33848
25	0.08801	0.22357
26	0.13468	0.22722
27	0.12182	0.24357
28	0.1069	0.2389
29	0.19838	0.38617
30	0.25825	0.41347
31	0.16163	0.26958
32	0.1285	0.26048
33	0.12365	0.30555
34	0.097	0.30874
35	0.22103	0.35749
36	0.12399	0.29194
37	0.11712	0.22444
38	0.25722	0.39494
39	0.15302	0.26343
40	0.12083	0.30744
Average	0.152622029	0.299730294
Min	0.088013	0.223569
Max	0.274219	0.413469

From the above table, the following observations are made:

- The best population value ranging from 0.223569 to 0.413469 contains the best sequence of keystroke population. Where the top ranked sequence contains a critical keystroke typing signature.
- The lowest population value ranging from 0.088013 to 0.274219 contains the low ranked sequence with irregular keystroke signature.

ii) Feature mapping

After performing population subset selection, feature mapping is incorporated using the best population to map the features into higher dimensional data. The result of feature mapping using the CMU keystroke dataset and the Collected keystroke dataset is explained below.

The results of feature mapping using Kernel Mean Embedding in the CMU keystroke dataset are shown in Table 5.20.

Table 5.20. Performance of Feature Mapping using CMU Keystroke Dataset.

Before Feature Mapping	After Feature Mapping
Number of rows: 20400	Number of rows: 20400
Number of columns: 31	Number of columns: 150
Features generated: nil	Features generated: 119
Type of value: integer	Type of value: integer

The table 5.20 describes that 31 features from CSA are mapped into 150 features using Kernel Mean Embedding. The samples containing 150 features are obtained that contains 119 instantly generated features based on less frequent temporal information.

The results of feature mapping using Kernel Mean Embedding in the collected keystroke dataset are shown in Table 5.21.

Table 5.21. Performance of Feature Mapping using Collected Keystroke Dataset.

Before Feature Mapping	After Feature Mapping
Number of rows: 13600	Number of rows: 13600
Number of columns: 28	Number of columns: 150
Features generated: nil	Features generated: 132
Type of value: integer	Type of value: integer

From Table 5.21, it is observed that the selected keystroke population contains 28 features which are mapped using Kernel Mean Embedding to obtain 150 features. It generates 127 new features containing the information of the less frequent keystroke population.

iii) Dimensionality Reduction

After performing feature mapping, dimensionality reduction is incorporate to handle the dimensionality curse using Principal Component Analysis. The obtained sequence of temporal information is fed into PCA to diminish the dimension. The result obtained from dimensionality reduction using both the CMU keystroke dataset and the Collected keystroke dataset is explained below.

The results of dimensionality reduction using Principal Component Analysis in the CMU keystroke dataset are mentioned in Table 5.22.

Table 5.22. Performance of Dimensionality Reduction using CMU Keystroke Dataset.

Before dimensionality reduction	**After dimensionality reduction**
Number of rows: 20400	Number of rows: 20400
Number of columns (features): 150	Number of columns (features): 18 - 132 features are reduced

From Table 5.22, it is notable that only 18 features are considered for further processing.

The results of dimensionality reduction using Principal Component Analysis in the collected keystroke dataset are given in Table 5.23.

Table 5.23. Performance of dimensionality reduction using collected keystroke dataset.

Before dimensionality reduction	**After dimensionality reduction**
Number of rows: 13600	Number of rows: 13600
Number of columns (features): 150	Number of columns (features): 18 - 132 features are reduced

From the above Table 5.23, it is notable that only 18 features are considered for further processing.

From the above results, it is evident that Feature Engineering is accomplished in layer 1. The keystroke population is ranked in population subset selection using Clonal Selection Algorithm. Then, keystroke features are mapped into high dimensional features in feature mapping using Kernel Mean Embedding. The intricate keystroke behaviour is obtained by dimensionality reduction using Principal Component Analysis.

Layer 2: Core Behavior Identification

After dimensionality reduction, the Core Behavior Identification is performed with highly informative keystroke behavior obtained from Layer 1. DBN technique is used to identify core behavior for unintentional insider mitigation using user authentication. The following subsection discusses the results obtained from DBN for Core Behavior Identification using both CMU keystroke dataset and collected keystroke dataset.

i) DBN for user authentication

The result of DBN for user authentication using the CMU keystroke dataset and collected keystroke dataset is explained below.

Table 5.24 shows the result of the DBN and Convolutional Neural Network (CNN) in the UIM phase with CKPCA, Clonal Selection Algorithm (CSA), Clonal Selection Algorithm and Principal Component Algorithm (CSA_PCA), Principal Component Analysis (PCA) using the CMU keystroke dataset. It is evaluated using False Acceptance Rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER), and Accuracy.

Table 5.24: Results of DBN in the UIM Phase using CMU Keystroke Dataset.

Feature engineering Techniques	DL Algorithms	FAR (%)	FRR (%)	EER (%)	Accuracy (%)
CKPCA	**DBN**	**0.002**	**8.49**	**0.152**	**99.85**
CSA	DBN	0.156	16.74	0.449	99.55
CSA_PCA	DBN	0.17	16.20	0.45	99.54
PCA	DBN	0.2109	29.41	0.73	99.27
-	DBN	0.15	14.81	0.41	99.58
CKPCA	**CNN**	**0.52**	**9.53**	**0.66**	**99.34**
CSA	CNN	0.421	12.25	6.33	93.66
CSA_PCA	CNN	0.97	19.50	1.201	98.79
PCA	CNN	1.148	20.44	1.37	98.62
-	CNN	0.64	11.61	0.789	99.21

From the above table, the following observations are made:

- PCA with DBN obtained higher FRR (29.41%) and EER (0.73%).
- CSA_PCA with DBN outperforms PCA with DBN by lesser FRR (16.20%)
- CSA with DBN achieves accuracy (99.55%) and suppresses CSA_PCA with DBN
- The DBN obtained lesser EER (0.41%) and outperforms CSA with DBN.
- CKPCA with DBN achieves the best performance by obtaining FAR (0.02%), FRR (0.84%), EER (0.15%), and accuracy (99.84%).
- CSA with CNN achieves less accuracy (93.66%) and highest EER (6.33%).
- PCA with CNN outperforms CSA with CNN by achieving lesser ERR (1.37%).
- CSA_PCA with CNN obtained lesser EER (1.201%), FAR (0.97%) and outperforms PCA with CNN.
- CSA_PCA with CNN falls short of CNN in terms of EER (0.789%), FAR (0.64%), and FRR (11.61%).
- CKPCA with CNN achieves the best performance than CSA_PCA with CNN by obtaining FAR (0.52%), FRR (9.53%), EER (0.66%), and accuracy (99.34%).
- CKPCA with CNN demonstrates robust performance by achieving 99.34% accuracy but falls short of CKPCA with DBN due to slightly higher FAR (0.52%), and EER (0.65%).

Table 5.25 shows the result of the DBN in the UIM phase with a traditional CNN using the collected keystroke dataset. It is evaluated using False Acceptance Rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER), and Accuracy

Table 5.25: Results of DBN in the UIM Phase using Collected Keystroke Dataset.

Feature engineering Techniques	DL Algorithms	FAR (%)	FRR (%)	EER (%)	Accuracy (%)
CKPCA	**DBN**	**0.0011**	**2.47**	**0.081**	**99.91**
CSA	DBN	0.0078	9.996	3.24	96.75
CSA_PCA	DBN	0.0279	1.838	0.93	99.06
PCA	DBN	0.0849	8.92	0.37	99.62
-	DBN	2.62	66.35	11.27	88.72
CKPCA	**CNN**	**0.27**	**2.34**	**0.34**	**99.65**
CSA	CNN	3.23	0	3.23	96.76
CSA_PCA	CNN	0.57	5.42	2.99	97
PCA	CNN	1.49	8.35	1.65	98.34
-	CNN	3.235	0	3.23	96.76

From Table 5.25, it is observed that

- DBN achieved FAR (2.62%), FRR (66.35%), EER (11.27%).
- CSA with DBN outperforms DBN by lesser ERR (3.24%).
- CSA_PCA with DBN achieves accuracy (99.06%) and suppress CSA with DBN
- PCA with DBN obtained lesser EER (0.37%), accuracy (99.62%) and outperforms CSA_PCA with DBN.
- CKPCA with DBN achieves the best performance by obtaining FAR (0.0011%), FRR (2.47%), EER (0.081%), and accuracy (99.91%).
- CSA with CNN and standalone CNN obtained a highest FAR of 3.235%, EER of 3.23%, and accuracy of 96.76%.
- CSA_PCA with CNN achieves higher accuracy (97%) and outperforms standalone CNN and CSA with CNN.
- PCA with CNN outperforms CSA_PCA with CNN by achieving lesser ERR (1.65%), and accuracy (98.34%).
- CKPCA with CNN achieves the best performance than PCA with CNN by obtaining FAR (0.27%), FRR (2.34%), EER (0.34%), and accuracy (99.65%).

From the above findings, it is observed that the results emphasize the critical role of CKPCA in optimizing the Deep learning model for keystroke dynamics-based biometric authentication for unintentional insiders.

The following Table 5.26 gives the mean performance of CKPCA-DBN in the UIM phase for authenticating the unintentional insiders. It is evaluated based on FAR, FRR, EER, and accuracy.

Table 5.26: Mean Performance of CKPCA-DBN in the UIM Phase

Dataset	DL algorithms	FAR (%)	FRR (%)	EER (%)	Accuracy (%)
CMU keystroke dataset	CKPCA-DBN	0.002	8.49	0.152	99.85
Collected keystroke dataset		0.0011	2.47	0.081	99.91
Mean Performance		**0.84**	**0.15**	**0.5**	**99.84**

From the above table 5.26, the following observations were noted:

- The mean performance of CKPCA-DBN achieved 0.84% FAR, 0.15% FRR, and 0.5% EER, with an overall accuracy of 99.84%.

5.3.3 COMPARISON OF PROPOSED CKPCA-DBN WITH EXISTING METHODS

The performance of the UIM phase (Unintentional Insider Mitigation) is compared with various state-of-the-art methods for biometric authentication. It is evaluated using FAR, FRR, EER, and Accuracy. Table 5.27 compares the average performance of the UIM phase with existing techniques after applying CKPCA-DBN.

Table 5.27: Performance Evaluation of CKPCA-DBN in the UIM Phase vs Other State-of-art-methods

Study	Method	Dataset	Results			
			EER (%)	FAR (%)	FRR (%)	Accuracy (%)
Proposed UIM	**CKPCA-DBN**	**CMU**	**0.5**	**0.84**	**0.15**	**99.84**
Sharma et al. (2025)	CNN+RF	CMU	7%	-	-	93%
Eltoukhy et al. (2024)	Optimized RF	CMU	0.38	-	-	99.62
Arsh et al. (2024)	Optimized CNN	CMU	0.65	-	-	99.35
Chang et al. (2022)	Enhanced XGBoost	CMU	3.61	-	-	96.39
Singh et al. (2020)	Enhanced XGBoost	CMU	6.41	-	-	93.59
Tewari & verma (2022)	Enhanced CNN	CMU	1.43	-	-	98.57
Chintalapudi et al. (2020)	Enhanced MLP	CMU	9.07	-	-	90.93
Lamiche et al. (2019)	Enhanced MLP	CMU	4.45	8.3	0.6	95.55

From the above Table 5.27, it is evident that the proposed method significantly outperforms other existing state-of-the-art techniques in terms of low EER of 0.151%.

The above analysis highlights the importance of adopting CKPCA with DBN for authenticating the unintentional insider for mitigation. It achieves better accuracy and overall performance compared to other existing methods.

5.3.4 UNINTENTIONAL INSIDER AUTHENTICATION USING CKPCA-DBN

The following figure 5.8 depicts the results of UIM phase after applying CKPCA-DBN to authenticate the unintentional insiders detected in Phase I (P&ID) using CERT insider and CIC darknet datasets.

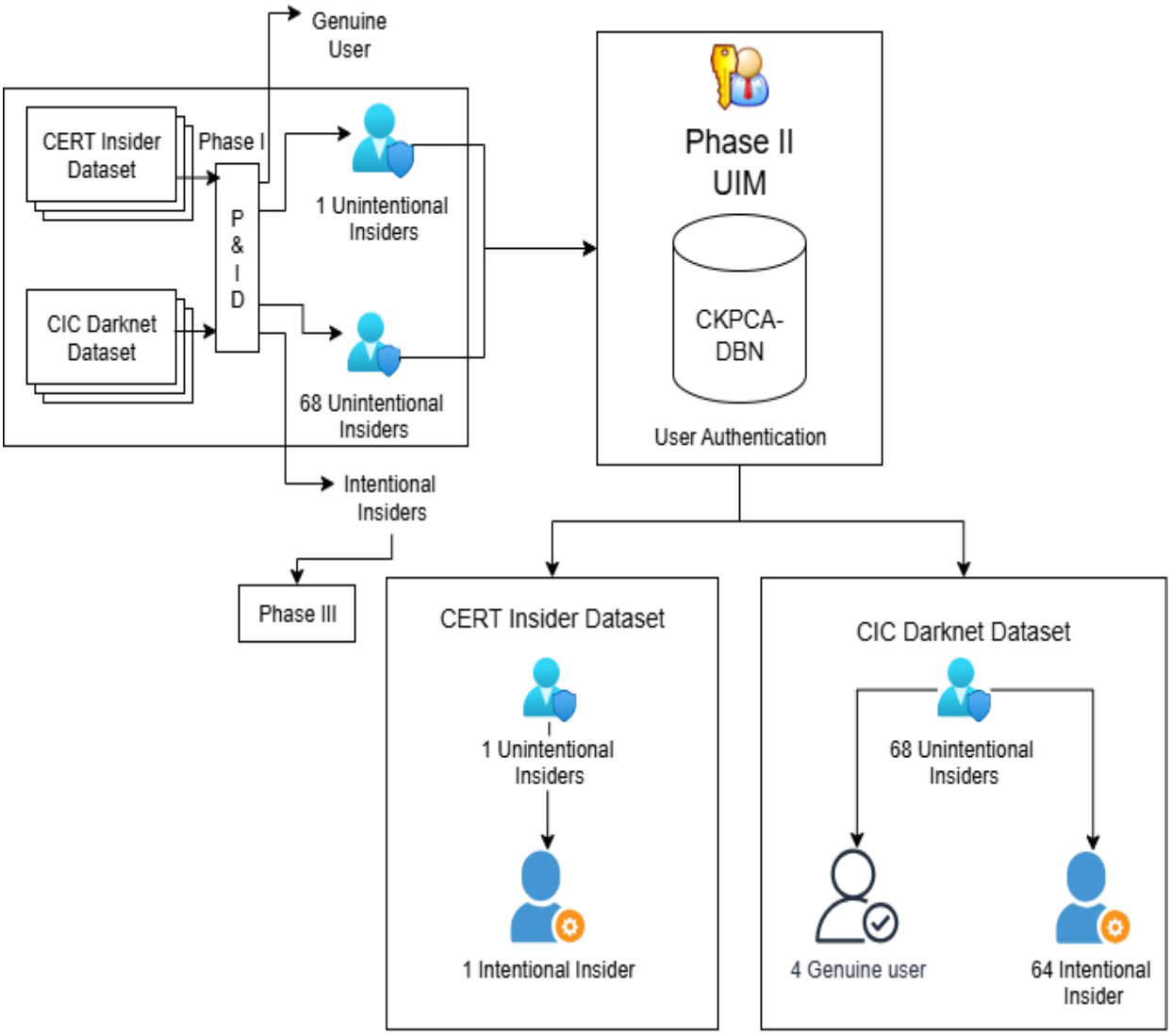


Figure 5.8: Results of UIM Phase After Applying CKPCA-DBN using CERT Insider and CIC Darknet Datasets

The detected unintentional insiders in P&ID phase using CERT insider and CIC darknet datasets is processed in UIM phase. The unintentional insiders are mitigated in UIM phase using proposed CKPCA-DBN. It is observed that UIM phase mitigates one unintentional insider as intentional insider using CERT insider dataset. In CIC darknet dataset, 68 unintentional insiders are mitigated as 4 genuine user and 64 intentional insiders using UIM phase.

5.4 OUTCOME OF PHASE II

The outcome of the UIM phase is listed below.

- The proposed phase enables the extraction of meaningful features, and detects irregularities in keystroke behavior, without losing essential information.
- The incorporation of the CKPCA approach successfully mitigates one unintentional insider and classifies them as an intentional insider using CERT insider dataset.
- The CIC darknet dataset was used to identify 68 unintentional darknet users in the P&ID phase. In the UIM phase, these 68 unintentional darknet users are authenticated using CKPCA-DBN to analyze the keystroke behavior of the user. Upon, 64 were classified as intentional darknet users and 4 as benign users.
- Unintentional Insiders are successfully mitigated using the proposed user authentication mechanism and achieved 99.84% accuracy with minimal EER (0.5%), FAR (0.84%), and FRR (0.15%). Thus, achieving the stated secondary objective 2.

5.5 LIMITATION OF PHASE II

Phase II has the following limitation.

- Phase II is limited to unintentional insiders mitigation. This phase does not offer mechanism for mitigating intentional insiders. So, a mechanism is required for intentional insider mitigation.

5.6 CHAPTER SUMMARY

The Unintentional Insider Mitigation phase addresses the challenge of mitigating unintentional insider threats. A novel Feature Engineering technique CKPCA combines population subset selection, feature mapping, and dimensionality reduction is proposed. CKPCA successfully extracted the keystroke representation. Next to Feature Engineering is Core Behavior Identification. For behavior identification, a DBN model is proposed for authenticating unintentional insiders and classifies them as either genuine user or intentional insider. Next to unintentional insider mitigation is intentional insider mitigation. The next chapter discusses the intentional insider mitigation elaborately.