International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(10s) 2026 pp. 1155-1170
https://doi.org/10.70917/ijcisim-2026-3705
Article

Machine Learning-Based Performance Assessment of Fragmented and Non-Fragmented Encryption in Cloud Data Security

M.Pravallika[1] , P. Bhargavi[2]

[1,2]Dept. of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati

Abstract: – With the rapid advancement of cloud-based and network-centric applications, securing confidential data against cyber-attacks over the communication channel has become essential. Protecting against cyber-attacks through the traditional approach will be difficult because the traditional approach encrypts the entire dataset as a single unit, which consumes more computational resources and leaves a single point of failure, leading to the compromise of the entire dataset. So Fragmented-based encryption strengthens security due to limited data exposure. This paper presents a performance evaluation of fragment-based encryption versus conventional encryption based on the performance metrics and security resistance. Here, the proposed method employs splitting the data by column, then the fragments are set by fixed-size, and those fragments are labelled and validated as sensitive using a rule-based algorithm and a Random Forest algorithm. The fragmented data is encrypted using the AES–256 algorithm in GCM Mode with PKI authentication. This analysis shows that fragmented encryption achieves optimal computational resources and improves security by limiting data exposure.

Keywords: Fragmentation, Rule-based algorithm, Random Forest algorithm, AES – 256 encryptions

1. INTRODUCTION

In today's digital landscape, sharing data over the network and storing data in a cloud environment has become necessary. So, the data contains a mixture of sensitive (Account numbers, credit card numbers, PAN numbers, personal identifiers) and non–sensitive attributes. Securing the data during data transmission is essential in order to prevent security breaches and cyber intrusions.

The traditional approach encrypts the whole dataset as a single unit before sending it into the network or storing the data. While this approach ensures confidentiality and integrity, it addresses several limitations. Those limitations are that CPU and memory consumption will be high, different security policies are applied in a limited form and if the encryption keys are compromised or a single point of failure leads to exposure of the entire data.

To address these challenges, this paper examines the fragmentation-based encryption, which divides the entire data into small units and encrypts each unit individually. Here, fragments are labelled and validated as sensitive using a rule-based mechanism and a random forest mechanism. This sensitivity detection, performed using a hybrid Random Forest and rule-based classifier, is not used to alter the encryption strength applied to individual fragments; rather, it serves as an evaluation mechanism to identify fragments containing sensitive attributes and non-sensitive attributes so that the security and performance of the proposed scheme can be validated for sensitive and non-sensitive data and also to evaluate whether this uniform protection holds equally well for sensitive and non-sensitive data, not to vary encryption strength. To validate these findings, evaluate the performance and security of fragmentation-based encryption and non–fragmented encryption on different financial datasets.

2. RELATED WORKS

This section describes the previous work on the performance of the AES algorithm, Rule based algorithm, the Random forest algorithm and the Data fragmentation techniques.

Zhou, S., Zhao, S., & Ren, Z. (2025, April) [1] have done work on "Loosely Synchronized Rule-Based Planning for Multi-Agent Path Finding with Asynchronous Actions". Here, the author discusses the method of Multi Agent Path Finding (MAPF), which balances solution quality and

scalability, which handles 1000 agents efficiently. This loosely synchronized Rule – based planning is designed to handle asynchronous actions MAPF, which improves traditional synchronous action assumption.

Lovrenčić, R., & Škvorc, D. (2023) [3] have done work on "Multi-cloud applications: data and code fragmentation for improved security". Here, the author discusses the core idea that involves leveraging data and code fragmentation, distributing these fragments across multiple independent computer clouds to enhance data privacy both in storage and during computation. This approach moves away from traditional encryption methods by fragmenting data, allowing it to remain available in plaintext while still preserving higher security levels through distribution.

Hafizah, S. S., Noraziah, A., & Odili, J. B. (2020) [7] have done work on "Fragmentation Techniques with Ideal Performance in Distributed Database-A Review". Here, the author discusses performance of various fragmentation techniques in distributed database systems which improves the performance, reducing transfer costs, and minimizing access times and also suggests further research to explore additional techniques in this area.

Alsirhani, A., Bodorik, P., & Sampalli, S. (2018) [16] has done work on Data Fragmentation Scheme: Improving Database Security in Cloud Computing. Here, the author discusses the methods to enhance the data confidentiality in cloud computing by integrating encryption algorithms with data fragmentation techniques.

Sarica, A., Cerasa, A., & Quattrone, A. (2017) [18] have done work on "Random Forest algorithm for the classification of neuroimaging data in Alzheimer's disease". Here, the author presents a paper on the Random Forest algorithm for predicting Alzheimer's disease using neuroimaging data. The paper highlights the algorithm's effectiveness in handling multi-modal data and makes its helpful in early diagnosis.

Naser, S. S. A., Alamawi, W. W., & Alfarra, M. F. (2016) [21] have done work on "Rule-based system for diagnosing wireless connection problems using SL5 object". Here, the author presents the paper to diagnose wireless connection problems with Rule Based System, leveraging the SL5 object expert system language. The proposed system showed a 87% success rate, and it indicates effectiveness in accurately diagnosing wireless connection problems.

Panda, M. (2016) [19] have done work on "Performance analysis of encryption algorithms for security". Here, the author evaluates the performance of various symmetric and asymmetric algorithms, especially focusing on their efficiency in terms of encryption time, decryption time and throughput. It concludes that AES exhibits balanced performance.

Patil, P., & Narayankar, P. (2016) [20] have done work on "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish". Here, the author analyses the various encryption algorithms focusing on their strengths, weaknesses, cost and performance. In today's digital landscape, this analysis helps to choose the right algorithm based on security and efficiency for effective information and conclude that AES occupies this place.

Qin, B., Xia, Y., Prabhakar, S., & Tu, Y. (2009) [29] have done work on "A rule-based classification algorithm for uncertain data". Here, the author presents a rule – based classification algorithm, 'uRule', proposed design to handle uncertainties in real – world data and also discussed innovative measures for generating, pruning and optimizing classification rules, demonstrating strong performance even with highly uncertain datasets.

Knauf, R., Gonzalez, A. J., & Abel, T. (2002) [31] have done work on "A framework for validation of rule-based systems". Here, the author presents a paper on a rule-based system for validating

expert systems to compute dependency sets and refining rules based on expert evaluation. It helps for selecting minimal yet effective test cases to ensure system reliability and accuracy.

3. METHODOLOGY

Here, the proposed methodology aims to enhance cloud data security and optimal computational resource consumption. The workflow of the proposed approach is

Step 1: Financial datasets are taken as input, and basic data preprocessing is performed. Step 2: Column Fragmentation is performed on those datasets.

Step 2: In this column, a fragmented data Rule–based algorithm is applied to label the fragments as sensitive or non–sensitive.

Step 3: These labelled fragments are validated by the Random Forest algorithm. If a fragment is labelled as sensitive by the rule-based method, the final result is also sensitive. Otherwise, if the fragment is labelled as non-sensitive, the random forest algorithm validates the fragment's patterns of sensitive attributes, structure, footprint, and size based on its training. If the non-sensitive fragments consist only of non-sensitive data, the fragment is finally labelled as non-sensitive. Otherwise, if the non-sensitive fragments contain sensitive data, it will reclassify that fragment's label as sensitive.

Step 4: Again, on these labelled fragments, a fixed-size fragmentation technique is applied, splitting them into fixed-size blocks (8KB).

Step 5: All the fragmented data is encrypted using the AES–256 algorithm.

A. Proposed Fragmentation – Based Secure Encryption Framework

Input

- Financial Dataset D
- Set of sensitive attributes SA
- Fragment size k
- Validating and Generalization fragments with the Random Forest classifier RF

Output

- Securely encrypted all the fragments
- Performance metrics (Encryption time, CPU usage, Memory, Latency, Throughput)

Begin

Data Preprocessing

- Eliminating all the missing and noise values
- Handle duplicates
- Standardize the attribute formats Column – Based Fragmentation
- Dataset D is divided into column fragments C = {c1, c2, --- cm} Rule – Based Sensitive Labelling
- For each fragment of column Ci
- If Ci∩SA ≠ ∅, label as Sensitive
- Otherwise, label as Non – Sensitive Random Forest Validation
- RF extracts the fragment features (column count, size, memory footprint)
- Trained an RF model to predict the sensitivity label
- Confirm and validate rule – based label. Fixed – Size Fragmentation
- Validated fragments are converted into byte streams
- Each stream is divided into fixed – size blocks (8 KB) Fragment – Level Encryption
- Every fragment is encrypted with the AES–256 algorithm Performance Measurement
- Measured encryption time, CPU Utilization, Memory Consumption, Latency and Throughput.

Result

- Save sensitivity labels and performance metrics in Excel files.

End

B. Datasets

Real-world Financial datasets are used to evaluate the performance and security analysis of fragmentation-based encryption in comparison with non–fragmented encryption. The selected datasets represent structured data, and it contains mixture of sensitive and non–sensitive attributes. Dataset sizes are Financial Transactions (2MB), Metraverse_transactions_dataset (14.1 MB), credit_card_transactions (340MB), transactions_data(4) (1.17GB) and trained dataset sizes are Combined Fictional F.T (89 KB), Transaction data (129 KB), bank_transactions_data(2) (337 KB), Cards_data (498 KB), users_data (161 KB).

C. DATA PREPROCESSING

Here, the structured financial datasets are taken as an input. On these datasets, basic data preprocessing is performed. Those are:

- Eliminating all the missing and noise values
- Handle duplicates
- Standardize the attribute formats

D. FRAGMENTATION

Fragmentation is a data fragmentation technique that splits the data into multiple chunks, and those small chunks are called fragments. These small chunks are divided before storage or transmission, which enhances cloud data security. While splitting data into fragments, an attacker cannot access the single fragment without reconstructing the remaining fragments and their contextual relationships.

The objective of the proposed work is:

- It minimizes data exposure in case of partial compromise of a fragment.
- Reduces the computational overhead by applying the selective fragment encryption (i.e., sensitive fragments)

This paper focuses on two fragmentation strategies

a. Column - Wise Fragmentation

b. Fixed – size Fragmentation

a. . Column – Wise Fragmentation

Column fragmentation is a data fragmentation technique that splits the data vertically based on the column attribute. This is also known as Vertical Fragmentation. Every fragment contains a subset of the columns from the original table, and it also consists tuple ID (i.e., Primary Key). This technique is suitable for structured data.

Example:

A Table with the column names

ID	Name	Email	Account No	PAN	Balance	Transaction Date

With the help of column-wise fragmentation, the attributes are grouped as fragments

Fragment F1

ID	Name	Email

Fragment F2

Account No	PAN

Fragment F3

Balance	Transaction Date

The above example shows that fragments will contain sensitive, non – sensitive attributes and a mix of both.

Fragment F1 consists of only non–sensitive attributes

Fragment F2 consists of only sensitive attributes

Fragment F3 consists of both sensitive and non–sensitive attributes

The above explains that in the proposed work, fragments consist of only sensitive or only non – sensitive or a mix of both attributes. So that it improves scalability, clarity and performance of the overall system.

Algorithm 1: Column-based Fragmentation 1: Initialize empty fragment set F

2: Extract column list C from dataset D

3: ***for*** *i = 1 to |C| step k* ***do***

4: Create fragment Fi containing columns {Ci, Ci+1, ..., Ci+k-1} 5: ***Add*** *Fi to F*

6: ***end for***

7: ***return*** *F*

Column–based fragmentation approach is shown in Algorithm 1. It extracts the column list (C) from the dataset (D). Now it divides the columns (C) sequentially into groups of size k. Fragment (Fi) contains the columns (Ci) of size (k). Finally, it added to the fragment set. Here, the group size of k =3 is selected because to achieve an optimal balance between information isolation and processing overhead. Choosing a very small number leads to excessive fragment counts, like higher metadata management, encryption invocation overhead, and memory access costs and having a large number leads to the risk of sensitive attribute correlation within a single fragment and weakens the security.

b. Fixed Size Fragmentation.

Fixed-size fragmentation is a technique that divides the data into equal-sized blocks (Eg, 4KB, 8KB, 16 KB or 1KB). This technique is suitable for both structured and unstructured data, such as text files, images, logs and encrypted datasets.

- It enables parallel processing and efficient resource utilization
- Obscures data patterns and boundaries
- Enhances resistance against traffic analysis attacks. Example:

Let's assume an encrypted dataset having a size of 24 KB and splitting the fragment into 8 KB. The dataset is divided as follows:

Fragment F1: 0 – 8KB Fragment F2: 8 -16KB Fragment F3: 16 – 24KB

In the proposed work, fixed-size fragmentation is applied after encrypting the dataset with the AES-256 algorithm.

Algorithm 2: Fixed – size Fragmentation

1: Convert fragment F into byte stream Bstream 2: Initialize empty block set B

3: ***for*** *i = 0 to length(Bstream) step S* ***do*** *4: Extract block Bi = Bstream[i : i+S] 5:* ***Add*** *Bi to B*

6: ***end for***

7: ***return*** *B*

Fixed – size fragmentation approach is shown in Algorithm 2. It converts the column fragment (F) into a byte stream, which is then divided into blocks of size B. If the last block size is less than B, padding is added to bring the size to B. Here, the block size of B = 8KB is selected to conclude system–level efficiency and cryptographic performance considerations. Choosing the smaller block size leads to an increase in encryption operations, higher CPU Context switching, function–call overhead and degraded throughput. Having a large block size increases memory pressure and encryption latency, which leads to the risk of data exposure in case of partial compromise. Therefore, it achieves balanced performance of encryption granularity.

E. Rule-Based Algorithm

A rule-based algorithm is a well-known technique for data mining. Here, it makes or concludes decisions by using different "If…else" rules. In IF…else rules, the IF part specifies one or more attribute conditions, and the THEN part assigns a corresponding statement. The decision will be made as soon as the condition is satisfied. It produces consistent and predictable results.

A rule–based sensitivity labelling approach is shown in Algorithm 3. It works on column-based fragmented data to identify the sensitive and non-sensitive fragments before encryption. The algorithm follows a consistent, sequential decision process; it evaluates each fragment (f) based on the predefined sensitive attributes. Here, each fragment(fi) consists of a group of attributes. If the fragment contains at least one attribute (a) belonging to the sensitive attribute set (s), then labelled as a sensitive fragment; otherwise, it is labelled as a non–sensitive fragment. Any fragment containing sensitive information is always identified correctly, so this ensures zero false negatives.

Algorithm 3: Rule-Based Sensitivity Labelling (RB-SL)

Input:

Dataset D

Sensitive Attribute Set S Column Fragment Size k

Output:

Labelled Fragment Set F

Begin

1: F ← ∅

2: Perform column fragmentation on D using k columns per fragment 3: **for** each fragment f ∈ D **do**

4: label ← Non-Sensitive

5: **for** each attribute a ∈ f **do**

6: **if** a ∈ S **then**

7: label ← Sensitive

8: break

9: **end if**

10: **end for**

11: Assign label to fragment f 12: Add (f, label) to F

13: **end for**

14: **return F** *End*

F. Random Forest Algorithm

Random Forest Algorithm is a machine learning Algorithm. This approach consists of several independent decision trees that are evaluated to improve prediction accuracy. The decision trees are constructed using bootstrap sampling of the training data, and each tree consists of different random parts of the data. Every tree learns decision boundaries independently based on different subsets of data and attributes. Each tree produces its own answer or prediction based on what it's learning in the training phase, and every individual tree produces a prediction result.

Voting will be performed on every predicted result, and the most voted prediction result will be taken as the final output.

A Random Forest Algorithm approach is shown in Algorithm 4. Here, this algorithm works on validating and generalizing the fragment sensitivity label. Training datasets are generated into fragments with the help of fragmentation techniques and labelled as sensitive or non–sensitive using a rule–based algorithm and converted into feature vectors. So, that algorithm learns a sensitivity pattern by random sampling and feature selection through multiple decision trees. Now, for the new datasets, the trained model predicts the sensitivity of the fragments through majority voting and then combined to rule-based decisions to generate the final label.

Algorithm 4: Random Forest – Based Fragment Validation

Input:

Dataset D

Rule-based fragment labels R Number of trees T

Output:

Validated fragment labels RF_Label

Begin

1: FragmentSet ← Perform column fragmentation on D 2: FeatureSet ← ∅

3: LabelSet ← ∅

4: **for** each Fragment Fi ∈ FragmentSet **do**

5: Extract feature vector Xi from Fi 6: FeatureSet ← FeatureSet ∪ Xi 7: LabelSet ← LabelSet ∪ R(Fi)

8: **end for**

9: Train Random Forest RF using FeatureSet and LabelSet 10: **for** each Fragment Fi ∈ FragmentSet **do**

11: Predict label Li = RF(Xi) 12: **end for**

13: **return** RF_Label *End*

G. *AES (Advanced Encryption Standard)*

Overview

Advanced Encryption Standard (AES) is a symmetric Encryption Algorithm. It was designed by Joan Daemen and Vincent Rijmen and published by the National Institute of Standards of Technology (NIST) in 2001. This algorithm is also called the Rijndael algorithm.

Architecture

1) It follows the Feistel Cipher Structure.
2) It is a Symmetric block cipher algorithm with a block size of 128 bits.
3) It takes 128 bits of plaintext as input and produces 128 bits of ciphertext as output.
4) Its key size is 256 bits and performs 14 rounds.
5) For every round, it uses a key of 128 bits and 128 bits of plain text.

Algorithm 5 shows that the data is encrypted using the AES–256 algorithm in GCM Mode (i.e., it supports both CTR + authentication) and authenticated using the PKI framework. In the PKI framework RSA algorithm is suitable for secure key exchange because it supports all the parameters (encrypting key, attaching a digital signature and key agreement.

Algorithm 5: AES – 256 Encryption

Input:

- *AES key K*
- *RSA public key PK*

Output:

- *AES key K′ encrypted*

Steps:

1. *AES-256 key K is generated*
2. *Every fragment of data is encrypted using AES-256 with a key K.*
3. *Using the RSA public key PK*, *The key K is Encrypt*
4. *Encrypted key K'is stored separately from encrypted fragments.*
5. *Use the RSA private key only during decryption*.

H. Performance Metrics

i. Encryption Time

The encryption time can be calculated as the time taken to convert from plain text into ciphertext. It is measured in seconds.

Encryption time is measured as:

$T_i = t_{end} - t_{start}$

Here

t_{start} = time is measured when encryption begins

t_{end} = time is measured when encryption completes

Multiple fragments are encrypted, and the average encryption time is calculated

$$T_{enc} = \frac{1}{N} \sum_{i=1}^{N} T_i$$

Here

N = total number of encrypted fragments

ii. CPU Usage

CPU Utilization is calculated to know how much CPU power is consumed while encrypting the whole data or fragmented data.

$CPU_i = CPU_{after} - CPU_{before}$

Here
CPU_{before} = CPU usage before encryption

CPU_{after} = CPU usage after encryption

$$\bar{C}\bar{P}\bar{U} = \frac{1}{N} \sum_{i=1}^{N} CPU_i$$

iii. Memory Usage

Memory usage is calculated based on the amount of memory consumed during encryption and decryption work. At that time, it requires temporary memory to store temporary data, keys and buffers to process the data present in the blocks.

$$M_i = Memory_{after} - Memory_{before}$$

Here

$Memory_{before}$ = CPU usage before encryption

$Memory_{after}$ = CPU usage after encryption

$$\bar{M} = \frac{1}{N} \sum_{i=1}^{N} M_i$$

iv. Latency

Latency is used to calculate the time delay while sending the plaintext to the encryption system and receiving the encrypted result.

Case A: Whole Dataset Encryption Latency = End_time − Start_time

Case B: Fragmentation-Based Encryption Latency = Average encryption time per fragment

" The average time taken to encrypt every individual fragment after fragmentation"

Mathematically:

$$Latency = \frac{\sum_{i=1}^{n} T_{enc}(fragment_i)}{n}$$

Here:

n= number of fragments

T_{enc}= encryption time per fragment

v. Throughput

The amount of data that is encrypted and decrypted per second

Without Fragmentation

$$\text{Throughput} = \frac{D}{T_{enc}}$$

Here:

D= total data size (MB or bytes)

T_{enc}= total encryption time (seconds)

With Fragmentation

$$\text{Throughput} = \frac{\sum_{i=1}^{n} D_i}{\sum_{i=1}^{n} T_{enc}(fragment_i)}$$

Since:

$$\sum_{i=1}^{n} D_i = D$$

This simplifies to:

$$\text{Throughput} = \frac{D}{T_{total}}$$

4. EXPERIMENTAL RESULTS

Evaluating the performance of various dataset sizes of Financial Transactions (2MB), Metraverse_transactions_dataset (14.1 MB), credit_card_transactions (340MB), transactions_data (1.17GB), to analyse its scalability and computational efficiency. The performance of these datasets is evaluated based on the two encryption strategies. They are:

- ***Non-fragmented Encryption***

  Encrypting the entire dataset as a single unit using AES – 256 in GCM mode with PKI (RSA) authentication.

- ***Fragmented – Based Encryption***

Here, the dataset is divided into chunks called as fragments. These chunks are divided based on column - fragmentation followed by fixed-size fragmentation. Each fragment is encrypted individually using AES – 256 in GCM mode with PKI (RSA) authentication. These fragments are labelled as sensitive and non – sensitive using a rule – based mechanism, and the labels are validated using a Random Forest (RF) classifier.

The performance of these two encryption strategies is evaluated based on these performance metrics:

- encryption time,
- CPU utilization,
- memory consumption,
- latency
- throughput

Environment Setup

The Jupyter Notebook was used to analyse fragment-based encryption with conventional encryption. The Python programming language is used to implement these experiments using a 12th Gen Intel(R) Core (TM) i5-1240p 1.70 GHz CPU with 16 GB RAM, a 64-bit operating system, and an x64-based processor.

Datasets

Real-world Financial datasets are used to evaluate the performance and security analysis of fragmentation-based encryption in comparison with non–fragmented encryption. The selected datasets represent structured data, and it contains mixture of sensitive and non–sensitive attributes. Dataset sizes are

The following are the sensitive attributes among all the data sets:

"cc_num","first","last","street","zip","lat","long","dob","amt",
"trans_num","merchant","category","job","gender","merch_lat",
"merch_long","merch_zipcode","accountid","transactionamount",
"accountbalance","transactionid","address","latitude","longitude",
"credit_score","yearly_income","per_capita_income","total_debt",
"birth_year","birth_month","id","client_id","card_number","cvv",
"expires","credit_limit","year_pin_last_changed","card_on_dark_web",
"location","deviceid","ipaddress","merchantid","customerage",
"customeroccupation","sender","receiver","amount","fee","senderaccount
id","receiveraccountid","geolocation","device used","pin ode","sending_address","receiving_address",
"ip_prefix","location_region","purchase_pattern","risk_score", "cardid","merchant id","mcc".

a. Results

Analysis of Encryption Time

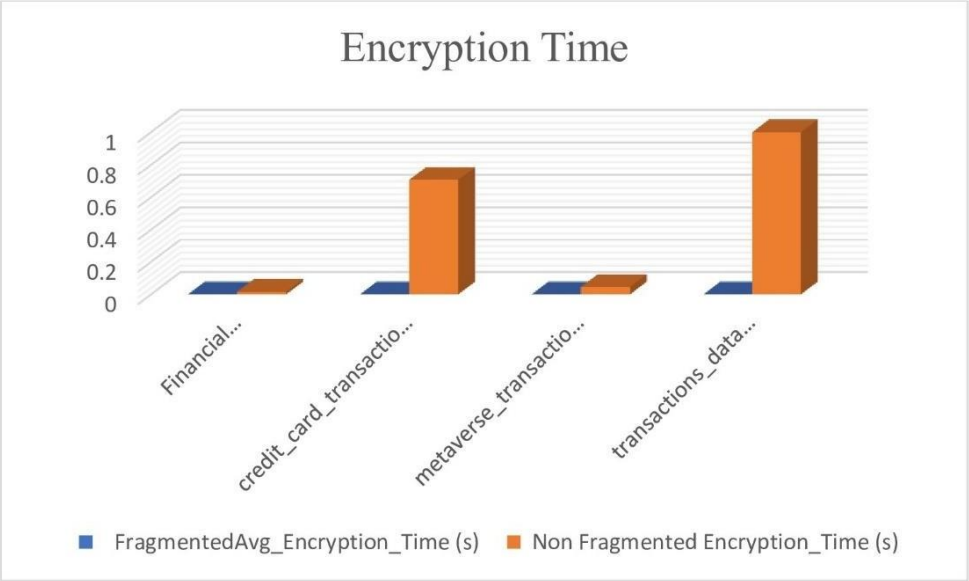

Figure 1. Encryption time on fragmented and Non – Fragmented data

Figure 1 visualizes the comparison of the average encryption time for fragmented data with the encryption time for non–fragmented data on four real-time datasets. In the figure, the X–axis represents Encryption time in seconds and the Y – axis represents the fragmented and non–fragmented datasets of various sizes. This visualization shows that in the non–fragmented scenario, encryption time increases substantially with the dataset size, because of encrypting the entire dataset as a single unit, which leads to sustained CPU utilization and increased memory access. In a fragmentation–based encryption approach achieves a lower average encryption time compared to a non–fragmented approach over all the datasets.

Analysis of CPU Utilization

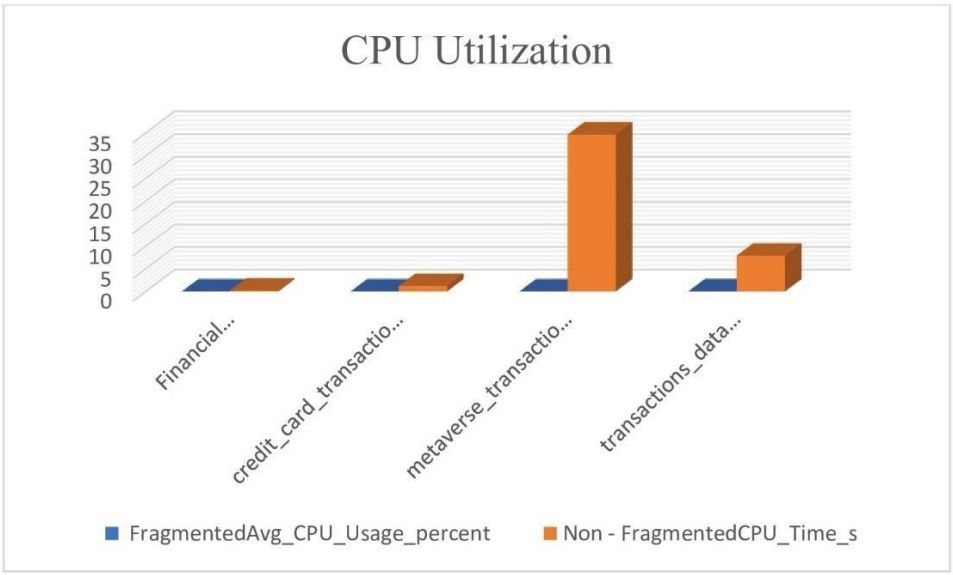

Figure 2. CPU Utilization on fragmented and non–fragmented data

Figure 2. visualizes the comparison of average CPU usage (%) of fragmented data and CPU usage (%) of non–fragmented data on four real-time financial datasets. In the figure, the X–axis represents CPU usage in percentage and the Y – axis represents the fragmented and non–fragmented datasets of various sizes. This visualization shows that the fragmentation–based dataset reliably achieves lower CPU utilization compared to the non–fragmented dataset,

because the non–fragmented dataset encrypts the entire dataset as a single unit, which results in high CPU consumption. As a result, a fragmented dataset enables more efficient CPU consumption and reduced peak utilization.

Analysis of Memory Consumption

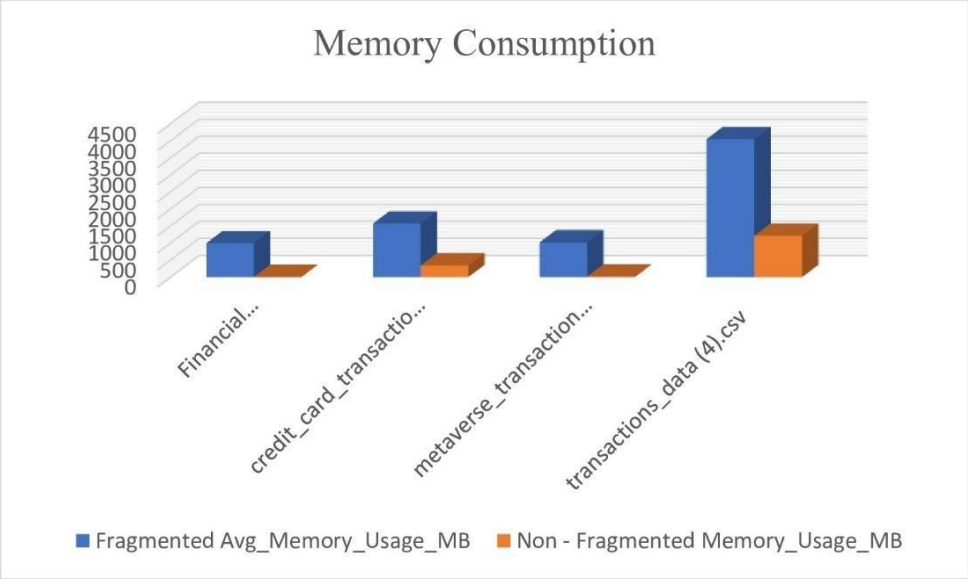

Figure 3. Memory Consumption of fragmented and non–fragmented dataset

Figure 3. visualizes the comparison of average memory consumption (MB) of fragmented data and memory consumption (MB) of non–fragmented data on four real-time financial datasets. In the figure, the X–axis represents Memory Usage in Megabytes and the Y – axis represents the fragmented and non–fragmented datasets of various sizes. This visualization shows that the fragmentation–based dataset consistently achieves higher memory consumption compared to a non–fragmented dataset, because the fragmented dataset adds additional intermediate operations like column–wise separation, fixed – size block, sensitivity labelling and metadata handling of each fragment. So, each fragment requires temporary buffering before encryption, which cumulatively increases memory consumption. But the fragmentation-based approach improves security, and it minimizes the exposure of data from memory-based attacks and enhances fault tolerance. A single point of failure will not lead to the loss of the entire data.

Analysis of Throughput

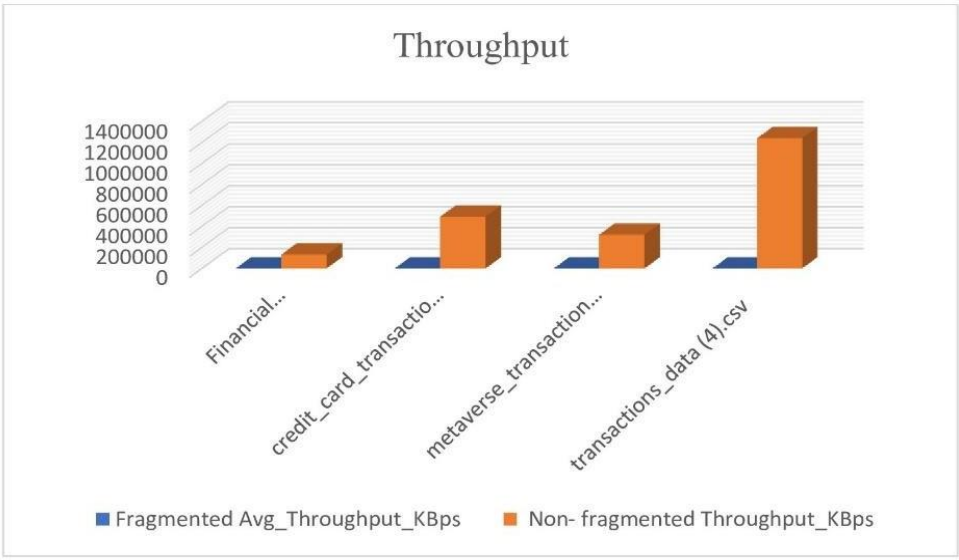

Figure 4. Throughput on fragmented and non – fragmented dataset

Figure 4. visualizes the comparison of average Throughput (KBps) of fragmented data and Throughput (KBps) of non – fragmented data on four real-time financial datasets. In the figure, the X–axis represents Throughput in KBps and the Y – axis represents the fragmented and non–fragmented datasets of various sizes. This visualization shows that the fragmentation–based dataset consistently achieves higher throughput compared to non – fragmented dataset, because the fragmented dataset.

Analysis of Latency

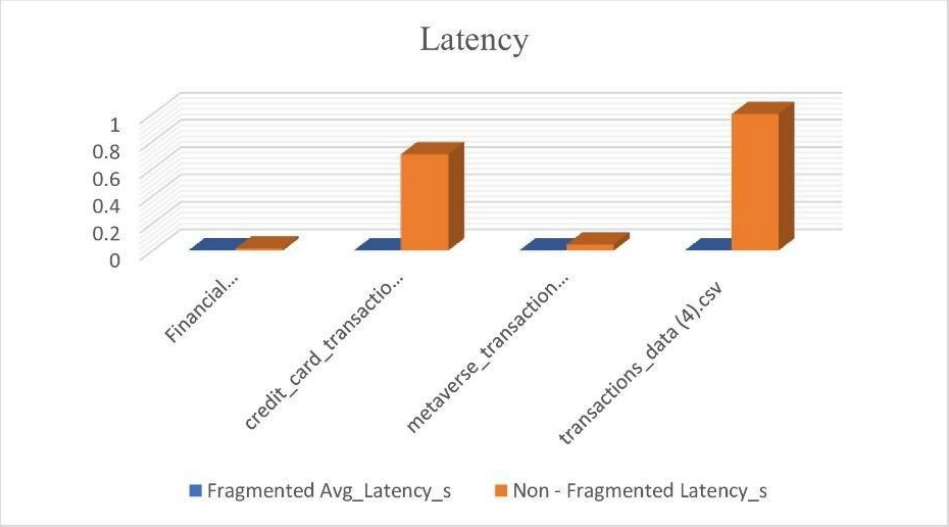

Figure 5. Latency on Fragmented and Non–Fragmented Data

Figure 5. visualizes the comparison of average Latency (s) of fragmented data and latency (s) of non–fragmented data on four real-time financial datasets. In the figure, the X–axis represents latency in seconds and the Y – axis represents the fragmented and non–fragmented datasets of various sizes. This visualization shows that the fragmentation–based dataset consistently achieves

lower latency across all the datasets compared to non – fragmented dataset. Fragmentation-based encryption minimizing the processing delay per encryption operation, which makes it most suitable for secure data transmission in large–scale, low-latency environments such as financial system-based applications.

Table I

Performance Analysis of Fragmented and Non – Fragmented Encryption Approach

Dataset	Data Size	Encryption Time		CPU Utilization		Memory consumption		Throughput		latency	
		Fragmented	Non - Fragmented	Fragmented	Non - Fragmented	Fragmented	Non - Fragmented	Fragmented	Non - Fragmented	Fragmented	Non - Fragmented
Financial Transactions.csv	2	0.000170882	0.015880585	0	0.2	994.8337935	2.41796875	444.9268666	131549.6248	0.000170882	0.015880585
credit_card_transactions.csv	340	0.000129633	0.706002235	0	1.2	1571.733987	340.3789063	805.0775455	493687.6645	0.000129633	0.706002235
metaverse_transactions_dataset.csv	14.1	0.000124362	0.044999838	0	34.7	1018.219296	14.109375	81.03224959	321007.515	0.000124362	0.044999838
transactions_data (4).csv	1213	0.000104038	0.999516964	0.00809574	7.9	4050.629047	1212.921875	159.2453796	1242628.512	0.000104038	0.999516964

Table I shows the results of the performance of both fragmentated and non–fragmented data, and its performance is evaluated on the parameters of encryption time, CPU utilization, Memory Consumption, throughput and latency.

After analyzing the results, it shows that the traditional encryption method achieves higher computational consumption because it encrypts the whole dataset under a single unit, so it takes a longer time to encrypt and decrypt. While encrypting the data for a longer time takes high CPU and Memory Usage, it is not an optimal choice.

On the other hand, the fragment-based encryption achieves optimal computational consumption, which is acceptable because crypto starts working after splitting the data into small pieces called fragments so it takes less

encryption and decryption time and also consumes less CPU energy due to computational overhead distributed more evenly, which prevents the CPU saturation.

But it produces high throughput and memory due to handling multiple fragments and temporary buffers. This behaviour is expected, that the throughput math spreads evenly across fragments. This holds steady and stable for every session, especially in time-critical environments. So, Fragmented-based encryption is an optimal choice.

Security Resistance

The proposed frameworks aim to achieve optimal computational consumption and strong security. When it comes to the security of cloud data in case of non–fragmented–based encryption, it cannot achieve strong security because a single point of failure leads to loss of the whole data. On the other hand, fragmentation-based encryption achieves strong security because it limits information exposure; any single fragment of failure does not reveal meaningful or complete data. In case of unauthorized access to encrypted fragments by any third party, the fragment data cannot be reconstructed due to the absence of contextual linkage across fragments.

This work achieves strong security resistance by applying fragmentation and sensitive-aware labelling, thereby providing cryptographic protection. Sensitivity – aware labelling ensures that the fragments contain critical attributes that require focused protection. Rule–based algorithm identifies the critical fields, while the random forest model validates and generalises this classification across heterogeneous datasets.

Overall, the security design framework – integrating fragmentation, intelligent sensitivity labelling, and strong encryption provides strong resistance against cryptographic, inference attacks, attribute linkage and data–centric attacks.

5. Conclusion

This paper presents a performance evaluation of both fragmentation-based encryption and non-fragmentation-based encryption. This evaluation was based on the performance metrics like encryption time, CPU usage, memory consumption, throughput, latency and security strength.

Here, the evaluation validates that fragmentation not only balances performance by consuming less encryption time, CPU usage and low latency, but also strengthens security by limiting data exposure, even in the case of partial compromise, which doesn't lead to full data disclosure. Compared to the traditional encryption method, the proposed approach is better suited for time–sensitive, resource–constrained, and high – volume data processing scenarios. This sensitivity detection, performed using a hybrid Random Forest and rule-based classifier, is not used to alter the encryption strength applied to individual fragments; rather, it serves as an evaluation mechanism to identify fragments containing sensitive attributes and non-sensitive attributes so that the security and performance of the proposed scheme can be validated for sensitive and non-sensitive data and also to evaluate whether this uniform protection holds equally well for sensitive and non-sensitive data, not to vary encryption strength.

This comparative analysis confirms that a fragmentation-based approach is the optimal choice because it provides a scalable, secure and efficient performance for modern data protection systems.

References

1. Zhou, S., Zhao, S., & Ren, Z. (2025, April). Loosely Synchronized Rule-Based Planning for Multi-Agent Path Finding with Asynchronous Actions. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 14, pp. 14763-14770).
2. Mushagalusa, C. A., Fandohan, A. B., & Glèlè Kakaï, R. (2024). Random forest and spatial cross-validation performance in predicting species abundance distributions. Environmental Systems Research, 13(1), 23.
3. Lovrenčić, R., & Škvorc, D. (2023). Multi-cloud applications: data and code fragmentation for improved security. International Journal of Information Security, 22(3), 713-721.
4. Hegde, S. K., Hegde, R., Hombalimath, V., Palanikkumar, D., Patwari, N., & Gowda, V. D. (2023, January). Symmetrized feature selection with stacked generalization based machine learning algorithm for the early diagnosis of chronic diseases. In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 838-844). IEEE.
5. Mohammed, M., & Alsunosi, R. (2022). Effect of selecting validation dataset on building random forest and decision tree models. AlQalam Journal of Medical and Applied Sciences, 470-478.
6. Nurgaliyev, A., & Wang, H. (2021, October). Comparative study of symmetric cryptographic algorithms. In 2021 International Conference on Networking and Network Applications (NaNA) (pp. 107-112). IEEE.
7. Hafizah, S. S., Noraziah, A., & Odili, J. B. (2020). Fragmentation Techniques with Ideal Performance in Distributed Database-A Review. vol, 6, 18-24.

8. Mar, J., Gorostiza, A., Ibarrondo, O., Cernuda, C., Arrospide, A., Iruin, Á., ... & Alberdi, A. (2020). Validation of random forest machine learning models to predict dementia-related neuropsychiatric symptoms in real-world data. Journal of Alzheimer's Disease, 77(2), 855-864.
9. Castro-Medina, F., Rodríguez-Mazahua, L., Abud-Figueroa, M. A., Romero-Torres, C., Reyes-Hernández, L. Á., & Alor-Hernández, G. (2019, February). Application of data fragmentation and replication methods in the cloud: a review. In 2019 international conference on electronics, communications and computers (CONIELECOMP) (pp. 47-54). IEEE.
10. Castro-Medina, F., Rodríguez-Mazahua, L., Abud-Figueroa, M. A., Romero-Torres, C., Reyes-Hernández, L. Á., & Alor-Hernández, G. (2019, February). Application of data fragmentation and replication methods in the cloud: a review. In 2019 international conference on electronics, communications and computers (CONIELECOMP) (pp. 47-54). IEEE.
11. Javeed, A., Zhou, S., Yongjian, L., Qasim, I., Noor, A., & Nour, R. (2019). An intelligent learning system based on random search algorithm and optimized random forest model for improved heart disease detection. IEEE access, 7, 180235-180243.
12. Kwon, J. M., Kim, K. H., Jeon, K. H., Kim, H. M., Kim, M. J., Lim, S. M., ... & Oh, B. H. (2019).
13. Development and validation of deep-learning algorithm for electrocardiography-based heart failure identification. Korean circulation journal, 49(7), 629-639.
14. Abu Mettleq, A. S., & Abu-Naser, S. S. (2019). A rule based system for the diagnosis of coffee diseases. International Journal of Academic Information Systems Research (IJAISR), 3(3), 1-8.
15. Wang, H. L., Hsu, W. Y., Lee, M. H., Weng, H. H., Chang, S. W., Yang, J. T., & Tsai, Y. H. (2019).
16. Automatic machine-learning-based outcome prediction in patients with primary intracerebral hemorrhage. Frontiers in neurology, 10, 910.
17. Santos, N., Lentini, S., Grosso, E., Ghita, B., & Masala, G. (2019). Performance analysis of data fragmentation techniques on a cloud server. International Journal of Grid and Utility Computing, 10(4), 392-401.
18. Alsirhani, A., Bodorik, P., & Sampalli, S. (2018, November). Data Fragmentation Scheme: Improving Database Security in Cloud Computing. In Recent Trends in Computer Applications: Best Studies from the 2017 International Conference on Computer and Applications, Dubai, UAE (pp. 115-138). Cham: Springer International Publishing.
19. Nashat, D., & Amer, A. A. (2018). A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design. ACM Computing Surveys (CSUR), 51(1), 1-25.
20. Sarica, A., Cerasa, A., & Quattrone, A. (2017). Random forest algorithm for the classification of neuroimaging data in Alzheimer's disease: a systematic review. Frontiers in aging neuroscience, 9, 329.
21. Panda, M. (2016, October). Performance analysis of encryption algorithms for security. In 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES) (pp. 278-284). IEEE.
22. Patil, P., & Narayankar, P. (2016). A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish. Procedia Computer Science, 78, 617-624.
23. Naser, S. S. A., Alamawi, W. W., & Alfarra, M. F. (2016). Rule based system for diagnosing wireless connection problems using SL5 object.
24. Lengyel, L. (2015). Validating rule-based algorithms. Acta Polytech. Hung, 12(4), 17.
25. Sprague, B., Shi, Q., Kim, M. T., Zhang, L., Sedykh, A., Ichiishi, E., ... & Zhu, H. (2014). Design, synthesis and experimental validation of novel potential chemopreventive agents using random forest and support vector machine binary classifiers. Journal of computer-aided molecular design, 28(6), 631-646
26. Ebrahim, M., Khan, S., & Khalid, U. B. (2014). Symmetric algorithm survey: a comparative analysis. arXiv preprint arXiv:1405.0398..
27. Hudic, A., Islam, S., Kieseberg, P., Rennert, S., & Weippl, E. R. (2013). Data confidentiality using fragmentation in cloud computing. International Journal of Pervasive Computing and Communications, 9(1), 37-51
28. Ojugo, A. A., Eboka, A. O., Okonta, O., Yoro, R., & Aghware, F. (2012). Genetic algorithm rule-based intrusion detection system (GAIDS). Journal of Emerging Trends in Computing and Information Sciences, 3(8), 1182-1194.
29. Mandal, A. K., Parakash, C., & Tiwari, A. (2012, March). Performance evaluation of cryptographic algorithms: DES and AES. In 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science (pp. 1-5). Ieee.
30. Prasad, B. D. C. N., & Prasad, P. K. (2010). A Performance Study on AES algorithms. International Journal of Computer Science and Information Security, 8(6), 128-132.
31. Qin, B., Xia, Y., Prabhakar, S., & Tu, Y. (2009, March). A rule-based classification algorithm for uncertain data. In 2009 IEEE 25th international conference on data engineering (pp. 1633-1640). IEEE.
32. Tseng, M. H., Chen, S. J., Hwang, G. H., & Shen, M. Y. (2008). A genetic algorithm rule-based approach for land-cover classification. ISPRS Journal of Photogrammetry and Remote Sensing, 63(2), 202-212.
33. Knauf, R., Gonzalez, A. J., & Abel, T. (2002). A framework for validation of rule-based systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 32(3), 281-295.
34. Cordón, O., Herrera, F., & Villar, P. (2002). Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. IEEE Transactions on fuzzy systems, 9(4), 667-674.
35. Adomavicius, G., & Tuzhilin, A. (2001). Expert-driven validation of rule-based user models in personalization applications. Data Mining and Knowledge Discovery, 5(1), 33-58.
36. Tepandi, J. (1990). Verification, testing and validation of rule-based expert systems. IFAC Proceedings Volumes, 23(8), 175-180.

37. Owoc, M. L., & Galant, V. (1999). Validation of rule-based systems generated by classification algorithms. In Evolution and Challenges in System Development (pp. 459-467). Boston, MA: Springer US.
38. Darwish, A. M., & Jain, A. K. (1988). A rule based approach for visual pattern inspection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(1), 56-68.