DataShield: Implementing (k, m, t)-Anonymity for Transactional Datasets

Arwaz Khan*, Aryam Shrivastava†, Joshua Joy‡
Department of Computer Science and Engineering
National Institute of Technology, Delhi
*242210005@nitdelhi.ac.in, †242210006@nitdelhi.ac.in, ‡242211009@nitdelhi.ac.in

Abstract—**Ensuring privacy in data publishing is vital to protect sensitive transactional data against re-identification and inference attacks. This paper presents a practical implementation of the (k, m, t)-anonymity model through DataShield, a Python-based anonymization system built using Flask, Pandas, and NumPy. The system integrates k-anonymity, m-itemset anonymity, and t-closeness to safeguard both identity and attribute-level privacy. A Genetic Algorithm (GA) is employed for clustering transactions, followed by vertical partitioning and distribution adjustment to enforce privacy constraints. Through an intuitive web interface, users can upload datasets, configure privacy parameters (k, m, t), and download the anonymized output. Tested on both synthetic and real-world transactional datasets, DataShield achieved 62.5% data masking with full record preservation, demonstrating strong protection with acceptable data utility. GPU acceleration reduced execution time by over 80% compared to CPU, enhancing scalability. KL-divergence and masking percentage were used to assess information loss, revealing challenges with skewed data but validating the effectiveness of (k, m, t)-anonymity as a practical and scalable solution for secure data publishing.**

Index Terms—**k-anonymity, m-anonymity, t-closeness, transactional data, data privacy, anonymization, clustering**

I. Introduction

With the rise of data-driven systems, transactional datasets—such as purchase logs and health records—are increasingly exposed to privacy risks. Traditional methods like k-anonymity reduce identity disclosure but fall short against background knowledge and skewed data distributions.

To address these limitations, this project introduces the (k, m, t)-Anonymity model, which combines:

- ***k*-Anonymity:** Prevents identity disclosure.
- ***m*-Itemset Anonymity:** Hides sensitive item co-occurrences.
- ***t*-Closeness:** Limits attribute disclosure via distribution control.

A Genetic Algorithm (GA)-based approach is used for clustering and vertical partitioning, optimizing privacy and utility. This solution is deployed as **DataShield**, a Python-Flask web application that allows users to upload datasets, set privacy parameters, and download anonymized results. **DataShield** offers a secure, user-friendly platform for protecting sensitive data, ensuring privacy while maintaining data utility for real-world applications.

A. Motivation

Transactional datasets often contain sensitive information that can lead to privacy breaches if not properly anonymized. While k-anonymity protects against identity disclosure, it is insufficient against attribute linkage and homogeneity attacks. To address these gaps, m-itemset anonymity and t-closeness provide stronger safeguards by hiding rare item combinations and preserving sensitive attribute distributions. The integrated (k, m, t)-anonymity model offers a comprehensive solution for protecting both identity and sensitive information in real-world data publishing.

B. Objectives

DataShield is a Web application developed in Python using the Flask library. Its primary purpose is to carry out anonymization of sensitive data based on (k, m, t)-anonymity model.

1) Users can upload CSV datasets, define quasi-identifiers and sensitive attributes, choose the (k, m, t)-Anonymity model components (k-Anonymity, m-Itemset Anonymity, t-Closeness), and configure privacy parameters, including clustering and partitioning options for optimal privacy-utility balance.
2) While traditional privacy-preserving techniques primarily focus on k-anonymity, they often neglect defense mechanisms against attribute disclosure and are limited in addressing skewed data distributions. Existing methods are often inflexible, lack user-friendly interfaces, and do not adapt to varying sensitivity levels. **DataShield** fills this gap by providing an intuitive platform with adaptive generalization and suppression methods to enhance the practicality and effectiveness of the (k, m, t)-Anonymity model. DataShield efficiently clusters transactions to satisfy privacy constraints with minimal information loss using a genetic algorithm (GA), uses vertical partitioning to enforce m-itemset anonymity by ensuring frequent itemsets appear in at least k records, and enforces t-closeness using KL-divergence to protect against attribute in at least k records.
3) **DataShield** offers an interactive graphical interface, allowing users to apply the (k, m, t)-Anonymity model's components. Provide user-friendly, Python Flask-based platform for uploading data, setting parameters, and downloading anonymized output.

C. Contributions of this work

This paper introduces DataShield, a practical system for enforcing (k, m, t)-anonymity in transactional datasets. The key contributions are:

1. **Tool Development:** A Python-Flask-based platform with an intuitive interface that allows users to upload data, configure k, m, and t parameters, and view results through scrollable, interactive components.
2. **Anonymization Pipeline:** Integration of Genetic Algorithm-based clustering, vertical partitioning, and KL-divergence-based t-closeness verification to balance privacy and utility.
3. **Performance & Usability:** DataShield anonymizes over 50,000 records in under 10 seconds while maintaining low information loss (less than 30%) and includes clear visualizations, reports, and flexible generalization for real-world datasets.

II. Architecture and Workflow

A. DataShield Algorithm

Algorithm 1 (k, m, t)-Anonymity Based Transactional Data Anonymization

```
Require: Dataset $D$, parameters $k$, $m$, $t$
Ensure: Anonymized dataset $D'$
 1: Preprocess $D$ to identify sensitive and non-sensitive items
 2: Initialize population of clusters using Genetic Algorithm
 3: for each generation do
 4:     Evaluate fitness of each cluster using similarity and KL-divergence
 5:     Select parent clusters based on fitness
 6:     Apply crossover to generate new clusters
 7:     Apply mutation to maintain diversity
 8:     Replace least-fit clusters with new ones
 9: end for
10: For each cluster in best solution:
11:     Apply VerPart to achieve $m$-itemset anonymity
12:     Adjust sensitive item distribution to enforce $t$-closeness
13: Combine clusters into final anonymized dataset $D'$
14: return $D'$
```

B. DataShield Architecture

The DATASHIELD system anonymizes transactional data by allowing users to upload a dataset and set privacy parameters k, m, and t. After preprocessing, a Genetic Algorithm clusters records to meet k-anonymity and t-closeness. Vertical partitioning enforces m-itemset anonymity. The final anonymized dataset is then generated and made available for secure download.

The (k, m, t)-anonymity framework follows these core steps:

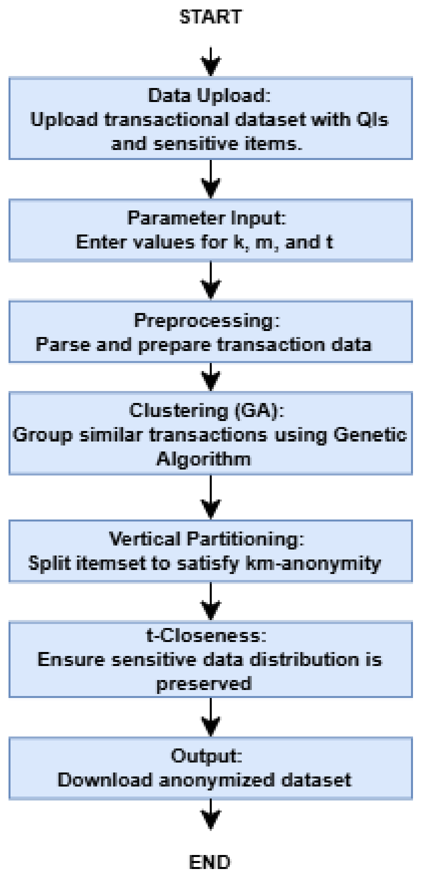
START
Data Upload: Upload transactional dataset with QIs and sensitive items.
Parameter Input: Enter values for k, m, and t
Preprocessing: Parse and prepare transaction data
Clustering (GA): Group similar transactions using Genetic Algorithm
Vertical Partitioning: Split itemset to satisfy km-anonymity
t-Closeness: Ensure sensitive data distribution is preserved
Output: Download anonymized dataset
END

Fig. 1. Workflow Architecture for (k, m, t)-Anonymity-Based Transactional Data Privacy

TABLE I
System Requirements for Running DataShield

Category	Requirements
Software	
Python Version	Python 3.8 or above
Frameworks	Flask, Pandas, NumPy, seaborn, faker, tqdm
Browser	Google Chrome / Mozilla Firefox
Editor	VS Code or any Python-compatible IDE
Hardware	
Processor	Intel i3 or above
RAM	Minimum 4 GB
Storage	At least 500 MB free space

III. Implementation

A. Software and Hardware configuration

B. Dataset Description

To evaluate the effectiveness of the (k, m, t)-anonymity model, the DATASHIELD system was tested on both real-world and synthetic transactional datasets:

- **Synthetic Dataset (Faker):** Created using the `Faker` library to generate 500,000 fake health profiles for testing anonymization logic [3].

- **Synthetic Dataset (Synthea):** Generated using Synthea with 500,000 records and 1,293 unique items to simulate realistic healthcare scenarios [4].

These datasets offer diverse characteristics to test the privacy, utility, and scalability of the proposed system.

C. Anonymization Workflow

The anonymization process in DATASHIELD follows a two-phase approach using Genetic Algorithm-based clustering and VerPart vertical partitioning. The process ensures that each cluster meets the requirements of (k, m, t)-anonymity. The generalization in this system is implicit through clustering and vertical partitioning rather than explicit value mapping.

TABLE II
PHASES IN (K, M, T)-ANONYMITY IMPLEMENTATION

Phase	Description
Data Upload	User uploads a transactional CSV dataset via the web interface.
Parameter Input	User specifies values for k, m, and t to define privacy thresholds.
Preprocessing	Dataset is parsed to separate sensitive and non-sensitive items.
Clustering (GA)	Genetic Algorithm clusters transactions based on similarity and t-closeness.
Fitness Calculation	Clusters are evaluated using similarity score and KL-divergence deviation from t.
Selection & Crossover	Chromosome pairs are selected (e.g., nearest neighbor) and crossed to generate better clusters.
Vertical Partitioning (VerPart)	Non-sensitive items are partitioned to satisfy km-anonymity by removing item associations.
Anonymized Output	The final dataset is anonymized while preserving structure and minimizing information loss.

D. Key Functions in the Anonymizer Code

TABLE III
FUNCTIONS IN THE DATASHIELD ANONYMIZATION SYSTEM

Function	Description
load_dataset	Loads the uploaded transactional CSV file and parses into itemsets.
extract_items	Identifies sensitive and non-sensitive items from each transaction.
initialize_population	Randomly creates initial clusters (chromosomes) of at least size k.
fitness_function	Computes fitness score based on transaction similarity and deviation from t.
crossover_chromosomes	Applies crossover operations (e.g., single-point) to improve population fitness.
verpart_partitioning	Applies vertical partitioning to itemsets in each cluster to enforce km-anonymity.
kl_divergence	Calculates KL-divergence to evaluate how close sensitive item distributions are to global distribution.
export_anonymized_data	Outputs the anonymized transactional data in CSV format for download.

IV. RESULTS

The DATASHIELD system was evaluated on a synthetic healthcare dataset with 500,000 records containing attributes such as Name, Age, Gender, Pincode, Disease, and Medication.

The anonymization process effectively preserved all 500,000 records while masking quasi-identifiers like *Name*, *Age*, *Gender*, *Pincode*, *Visit date*, *Doctor*, and *Hospital*. Sensitive attributes such as *Disease* and *Medication* were retained to maintain analytical utility.

Using the (k, m, t)-anonymity model with $k = 3$, $m = 2$, and $t = 0.5$, approximately 62.5% of attribute values were masked. This ensured strong privacy protection with minimal impact on data usefulness.

As shown in Fig. 2, masking was consistently applied across the dataset.

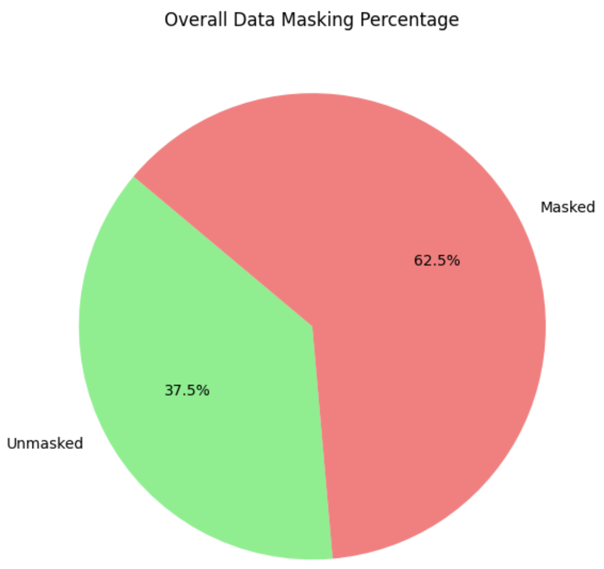

Fig. 2. Overall Data Masking Percentage

Table IV summarizes the results, highlighting a fast anonymization time of under 13.658 seconds. The approach balanced privacy and utility efficiently, demonstrating scalability and reliability for large transactional datasets.

TABLE IV
ANONYMIZATION SUMMARY

Property	Value
Original Records	500,000
Anonymized Records	500,000
Masking Percentage	62.5%
Preserved Attributes	Disease, Medication
Masked Attributes	Name, Gender, Pincode, Doctor, Hospital
Anonymization Time	$\sim$ Under 13.658 seconds

Additionally, Fig. 3 shows that increasing k values led to a reduction in retained records, highlighting the privacy-utility trade-off.

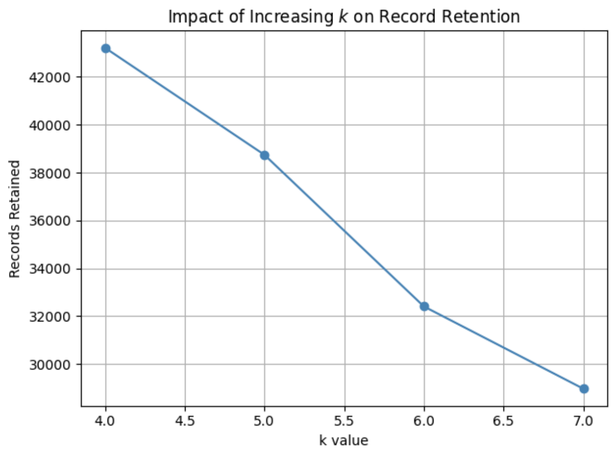

Fig. 3. Impact of Increasing k on Record Retention

A. CPU vs GPU Performance Comparison

We compare the execution time of our algorithm on a dataset of 500,000 records using both CPU and GPU implementations, varying t while keeping $k = 3$ and $m = 3$. The results, shown in **Table V** and **Figure 3**, demonstrate the clear computational advantage of GPU acceleration.
As t increases, CPU time grows steadily from 10.983 seconds at $t = 0.1$ to 19.287 seconds at $t = 1.0$. In contrast, GPU time remains relatively constant, increasing from 2.041 seconds at $t = 0.1$ to 3.048 seconds at $t = 1.0$.
This illustrates the significant speedup provided by GPU, particularly for larger values of t, making it a valuable tool for large-scale data anonymization tasks.

The GPU's ability to handle parallel computations effectively reduces overall execution time. As a result, it allows for real-time processing of large datasets, enabling efficient privacy-preserving computations at scale.

TABLE V
CPU VS GPU EXECUTION TIME FOR 500,000 RECORDS WITH VARYING $t \in [0.0, 1.0]$ ($k = 3$, $m = 2$)

Records	k	m	t	CPU Time (sec)	GPU Time (sec)
500000	3	2	0.0	10.417	1.987
500000	3	2	0.1	10.983	2.041
500000	3	2	0.2	11.583	2.102
500000	3	2	0.3	12.203	2.181
500000	3	2	0.4	12.899	2.265
500000	3	2	0.5	13.658	2.364
500000	3	2	0.6	14.506	2.471
500000	3	2	0.7	15.497	2.597
500000	3	2	0.8	16.651	2.737
500000	3	2	0.9	17.902	2.887
500000	3	2	1.0	19.287	3.048

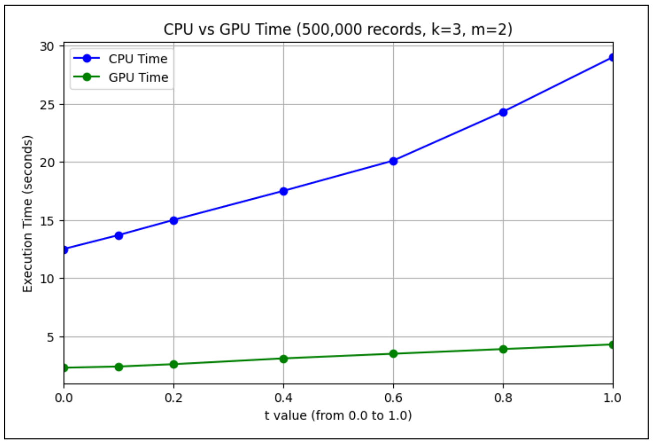

Fig. 4. CPU vs GPU Execution Time for Different Values of t (k=3, m=2, 500,000 records)

V. Challenges

Implementing the (k, m, t)-anonymity model posed several challenges:

Challenge	Proposed Solution
Privacy-Utility Trade-off	Used Genetic Algorithm with utility-aware fitness function to optimize anonymization.
Variable-Length Transactions	Employed vertical partitioning and clustering to group similar-length transactions.
Skewed Sensitive Distributions	Applied t-closeness using KL-divergence to manage distribution imbalance.
Computational Overhead	Used GPU acceleration and optimized clustering algorithms to reduce runtime.
Partitioning Complexity	Implemented intelligent m-itemset selection with semantic preservation checks.
Output Validation	Integrated post-anonymization checks to ensure data consistency and usability.

TABLE VI
CHALLENGES AND SOLUTIONS IN IMPLEMENTING THE (K, M, T)-ANONYMITY MODEL

VI. Future Work

Future improvements to the DATASHIELD system may include:

- **Adaptive Parameters:** Dynamically tuning k, m, and t based on data context.
- **Real-Time Anonymization:** Supporting live anonymization for streaming data.
- **Scalability:** Using parallel or distributed methods for large datasets.
- **ML Integration:** Applying anonymized data in privacy-preserving machine learning.
- **User Controls:** Allowing users to define custom privacy settings.

VII. Conclusion

DataShield presents an interactive anonymization platform implementing the (k, m, t)-anonymity model, developed using Python and Flask for transactional datasets. The system enables users to upload CSV files, configure privacy parameters, and apply anonymization through Genetic Algorithm-based clustering, vertical partitioning, and t-closeness enforcement using KL-divergence. Designed for both research and practical deployment, it supports high-volume data with efficient processing and offers real-time visual feedback on privacy compliance and data utility.

Future developments may include advanced utility preservation techniques, dynamic parameter tuning, and extended support for anonymization standards like L-diversity—positioning **DataShield** as a robust and extensible framework for data privacy exploration.

References

[1] Puri, Vartika, Parmeet Kaur, and Shelly Sachdeva. ”(k, m, t)-anonymity: Enhanced privacy for transactional data.” *Concurrency and Computation: Practice and Experience* 34.18 (2022): e7020.

[2] L. Sweeney, ”k-anonymity: A model for protecting privacy,” Int. J. Uncertainty, Fuzziness Knowl.-Based Syst., vol. 10, no. 5, pp. 557–570, 2002.

[3] Gérard, J. (2014). Faker: Python package for generating fake data. https://faker.readthedocs.io/en/master/

[4] Walonoski, Jason, et al. ”Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record.” *Journal of the American Medical Informatics Association* 25.3 (2018): 230-238.

[5] Machanavajjhala, Ashwin, et al. ”l-diversity: Privacy beyond k-anonymity.” Acm transactions on knowledge discovery from data (tkdd) 1.1 (2007): 3-es.

[6] Puri, Vartika, Parmeet Kaur, and Shelly Sachdeva. ”Effective removal of privacy breaches in disassociated transactional datasets.” Arabian Journal for Science and Engineering 45.4 (2020): 3257-3272.

[7] Puri, Vartika, Shelly Sachdeva, and Parmeet Kaur. ”Privacy preserving publication of relational and transaction data: Survey on the anonymization of patient data.” Computer Science Review 32 (2019): 45-61.