Internet of Things 23 (2023) 100831

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

An edge-aided parallel evolutionary privacy-preserving algorithm for Internet of Things

Akbar Telikani [a], Asadollah Shahbahrami [b,c,*], Jun Shen [a], Georgi Gaydadjiev [b], Jerry Chun-Wei Lin [d]

[a] School of Computing and Information Technology, University of Wollongong, New South Wales, Australia
[b] Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, Faculty of Science and Engineering, University of Groningen, The Netherlands
[c] Department of Computer Engineering, Faculty of Engineering, University of Guilan, Iran
[d] Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Data privacy
Edge computing
Evolutionary algorithms
Big data
GPU platform

ABSTRACT

Data sanitization in the context of Internet of Things (IoT) privacy refers to the process of permanently and irreversibly hiding all sensitive information from vast amounts of streaming data. Taking into account the dynamic and real-time characteristics of streaming IoT data, we propose a parallel evolutionary Privacy-Preserving Data Mining (PPDM), called High-performance Evolutionary Data Sanitization for IoT (HEDS4IoT), and implement two mechanisms on a Graphics Processing Units (GPU)-aided parallelized platform to achieve real-time streaming protected data transmission. The first mechanism, the Parallel Indexing Engine (PIE), generates retrieval index lists from the dataset using GPU blocks. These lists are used in place of the dataset during the PPDM process. The second mechanism, called Parallel Fitness Function Engine (PF2E), parallelizes the index lists on the GPU threads to speed up the computation of the quality of solutions generated by the evolutionary algorithm, in which deferential evolution is adopted as the evolutionary algorithm. In addition to the ability for Big data, the HEDS4IoT can be adaptively adjusted for dynamic nature of IoT where new streaming data is considered for data sanitization. Our experimental results with extensive benchmarks show that, at the kernel level, the PIE and PF2E mechanisms are averagely 33.5x and 53.7x faster than their CPU-implemented version, respectively. At the application level, our findings demonstrate that the HEDS4IoT can perform the PPDM process 47.7x faster than some of the state-of-art methods.

1. Introduction

A huge amount of data is generated at a breakneck speed in the era of the Internet of Things (IoT) and is outsourced to external entities, such as cloud centers, for storage and processing. This means that data generated by IoT devices is transmitted and stored to a remote location, where it can be processed, analyzed, and used to derive insights and inform decision-making. This collected data is analyzed using data mining and machine learning techniques, particularly frequent pattern mining, to improve service quality and help industrial sectors for different objectives, such as stock prediction, process automation, and resource planning. For instance,

* Corresponding author at: Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, Faculty of Science and Engineering, University of Groningen, The Netherlands.
E-mail address: a.shahbahrami@rug.nl (A. Shahbahrami).

https://doi.org/10.1016/j.iot.2023.100831
Received 7 December 2022; Received in revised form 6 May 2023; Accepted 22 May 2023
Available online 1 June 2023

the IoT network foundation and localization of IoT devices for improving retailers' customer service can be established through the extraction of relationships between spatial patterns [1]. In spite of the benefits of analytics in IoT-related data, privacy divulgence may be a major risk when information discovery is performed by a third party with the objective of getting touchy data [2]. For example, in a cloud-based collaboration between healthcare practitioners and education systems (e.g., universities) in a smart city, privacy leakage can threaten patients' treatment, such as medication habits and appointment behavior patterns. Moreover, organizations' stored data on the cloud or streamed over the network may be revealed because of cyber security incidents. In addition to conventional countermeasures (e.g., access control and encryption) for security threats, certain mechanisms are accordingly developed to avouch securing communication, storage, and knowledge discovery. Furthermore, data transmission in IoT networks may incur some security issues and malicious attackers may obtain the sensitive sensory data for benefit [3].

Privacy-Preserving Data Mining (PPDM) techniques can be taken into consideration for IoT environments to protect confidential and sensitive knowledge against security and privacy threats [4]. Data sanitization is the process of reducing the sensitivity level of sensitive information below an insignificant level from the point of view of a data owner and to whom the data is about. This reduction is performed through the modification of transactions generating sensitive knowledge. The objective of this process is to select the best sensitive transactions for modification in a way that all sensitive knowledge is hidden and all non-sensitive information is maintained in the sanitized dataset [5]. Evolutionary algorithms have brought benefits for sensitive transactions selection phase of PPDM algorithms. By selecting the transactions that need to be sanitized based on their impact on privacy, rather than a random or heuristic-based approach, it is possible to minimize the amount of data that needs to be sanitized, while maintaining the privacy of the data.

Despite providing promising opportunities for this objective in IoT, evolutionary PPDM algorithms are not adjustable for the streaming nature of IoT networks and scalable data sanitization techniques are needed [6,7]. This is because the fitness value commutation phases can be overwhelmingly expensive as a database scan is required for each fitness value calculation. In an evolutionary PPDM with hundreds iterations and various fitness value evaluation in each iteration, a lot of computational resources is needed. The use of GPU for privacy preserving has taken into consideration in these years [8–10]. Despite their efficiency, the existing privacy protection mechanisms only integrated data encryption and deep learning models and used GPU for the acceleration of this process. However, they do not consider PPDM and data sanitization mechanisms for IoT environments.

Based on the shortcoming of the evolutionary approaches that cannot provide an efficient PPDM process for IoT environments with streaming data, we develop a edge-computing-aided PPDM framework, called High-performance Evolutionary Data Sanitization for IoT (HEDS4IoT), to speed up evolutionary algorithms for IoT platform. This method exploits the GPU platform to remove sensitive knowledge at high speed before sharing the data. In the IoT, the edge computing can move computing resources from remote cloud centers to servers near IoT devices [11]. Application of GPU on the edge computing is already being utilized to take the advantage of fast computation of data as reported in the literature [12,13]. However, this advanced strategy has not been introduced for privacy protection in IoT environments. The main contributions of this study are as follows:

- A new PPDM algorithm, called HEDS4IoT, with the use of the GPU devices at the edge layer to accelerate the sanitization process for large-scale IoT-related data, is developed. An evolutionary algorithm is also employed in the HEDS4IoT to select sensitive transactions for sanitization. By exploiting the GPU devices and the evolutionary algorithm, sensitive knowledge can be hidden efficiently and effectively before disseminating the IoT data to the fog and cloud layers.
- The HEDS4IoT includes two GPU-aided high-performance computing mechanisms. The first mechanism is the Parallel Indexing Engine (PIE), which is responsible for generating specific index lists in a distributed manner using GPU devices. With this indexing strategy, the lists are scanned when computing the fitness value rather than scanning the dataset. The second mechanism is the Parallel Fitness Function Engine (PF2E), which performs fitness function through the distribution of the lists between GPU threads.
- A set of experiments have been conducted to evaluate the efficiency of the HEDS4IoT algorithm in terms of kernel-level and application-level on different real and synthetic datasets. Experimental results with extensive benchmarks show average kernel-level speedups of up to 33.5x and 53.7x for the PIE and PF2E, respectively. Also, the HEDS4IoT can achieve an average speed up of 44.6x at the application-level over its sequential version. In addition, comparisons with state-of-the-art models, such as Artificial Bee Colony for Association Rule Hiding (ABC4ARH) [14], Pareto Ant Colony Optimization to Delete Transactions (PACO2DT) [6], and Ant Colony System to Delete Transactions (ACS2DT) [15], show the superiority of the HEDS4IoT algorithm in terms of both reducing side effects and improving efficiency. The HEDS4IoT achieves averaged performance improvements of up to 47.41x, 44.11x, and 51.55x compared to the ABC4ARH, PACO2DT, and ACS2DT,respectively.

We organize the structure of this study as follows: In Section 2, the importance of data sanitization in IoT applications is discussed. Section 3 develops a GPU-aided evolutionary PPDM algorithm for IoT on the edge layer. Evaluation results are discussed in Section 4. Section 5 briefly summarizes related work on GPU-assisted privacy preservation and data sanitization process. Section 6 concludes the paper and suggests future work.

2. Fundamentals for PPDM in IoT

IoT network can be established in different domains, such as industry, medicine, retail, and smart buildings, and leads to generating a huge amount of data that is produced streamingly. The data is then sent to external entities, e.g., organizations and cloud, for storing and processing. However, data breaches in cloud providers or accessing the storage and communication resources

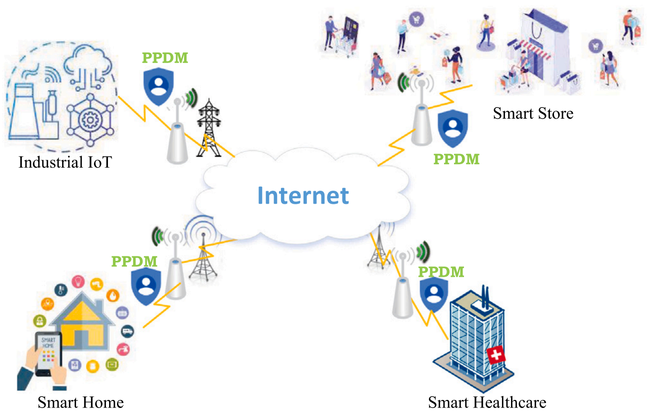


Fig. 1. Privacy-Preserving Data Mining (PPDM) in the IoTs, where the PPDM is employed at the edge of smart IoT-enabled applications, such as healthcare, industry, smart home, and smart store.

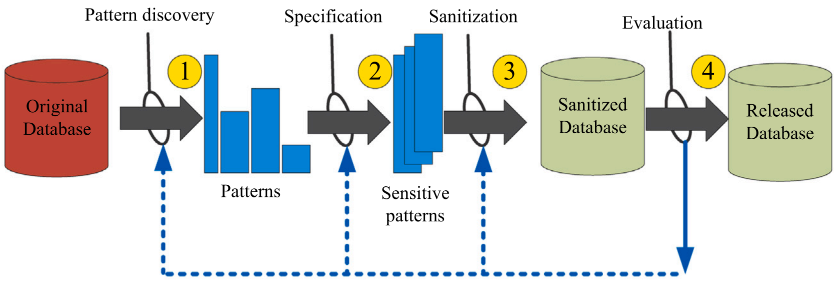


Fig. 2. Generic framework of PPDM.

by hackers may threaten data privacy. Hence, providing secure and protected data sharing and analysis are significant challenges in IoT.

Data encryption [16,17] and PPDM [18] can be considered two strategies for protecting the privacy and securing data analysis, transfer, and storage. Data encryption involves encrypting the data produced by IoT devices and storing it securely in a third-party repository. Security and privacy in the IoT are first protected through encryption. Otherwise, by breaking this protection level by attackers or extracting sensitive information by data recipients who participate under a multilateral agreement to analyze the data on the cloud, significant harms can be resulted to individuals or organizations. This makes PPDM a significant alternative for preventing sensitive information from being lost.

Fig. 1 illustrates how the PPDM can play a role for data privacy and security in IoT, where the data gathered from IoT devices and stored on the edge servers is modified to remove sensitive knowledge data distribution and sharing. Any application may have concerns about the private and/or sensitive information of the users and businesses. It is expected that the sensitive information will be properly hidden or masked to prevent attackers and third parties from revealing data owners' knowledge. Indeed, PPDM techniques are employed at the edge servers for data sanitization reasons.

In the PPDM, the sensitivity level (i.e., support or confidence) of the confidential and sensitive information is reduced below minimum thresholds. The PPDM process for association rules includes four steps: (1) pattern discovery, (2) specification, (3) sanitization, and (4) evaluation, as depicted in Fig. 2. In this process, a set of patterns are mined from the original data through a machine learning algorithm in the pattern discovery step. In the Specification step, patterns using personal or sensitive data are chosen as sensitive. The sensitive patterns are hidden via database modification strategy in the sanitization step. The evaluation step includes measuring the side effects of the PPDM process on the sanitized data.

Fig. 3 shows an illustration example of the PPDM process. In this example, we assume there are n transactions in the dataset (i.e., left hand side dataset) and some frequent patterns are extracted from it (i.e., step #1). Considering two patterns of "$I_2, I_8 \rightarrow I_{10}$" and "$I_3, I_6 \rightarrow I_9, I_{13}$" are sensitive (i.e., step #2), their corresponding transactions are modified in the sanitized dataset (i.e., step #3). In our example, we remove some items of the sensitive patterns from the transactions, which are shown in red color in the sanitized dataset. This means that removing the items belonging to the sensitive patterns "$I_2, I_8 \rightarrow I_{10}$" and "$I_3, I_6 \rightarrow I_9, I_{13}$" results in hiding these two patterns from the sanitized dataset.

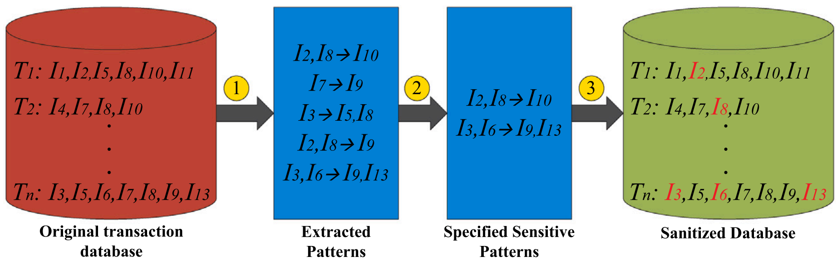


Fig. 3. An example of data sanitization for PPDM.

3. Proposed data sanitization approach for IoT

As the IoT infrastructure grows, the demand for big data processing capabilities for the data sanitization process increases. The reason for this is that huge amounts of IoT-related data are being collected at an astounding rate, which requires more sophisticated and innovative PPDM solutions. Cloud centers have the ability to quickly scale up large amounts of storage and processing. However, a cloud center may pose a risk to private and sensitive knowledge when outsourcing and processing IoT-related data remotely [19]. By bringing computing resources closer to the IoT devices where the data is generated, the edge computing can reduce transmission latency and protect privacy against hackers both in the communication network and on the cloud side regarding data storage and data analysis.

In this section, we propose a powerful evolutionary data sanitization algorithm, called HEDS4IoT (Fig. 4), for Big streaming IoT data over an edge computing platform. As the IoT environments generate streaming data, static application of the HEDS4IoT cannot guarantee privacy because the sensitivity level of patterns is changed by adding new data. Thus, in our approach, we have designed a strategy to employ the PPDM process periodically and the HEDS4IoT is re-executed after generating a new dataset by combining the last version of the dataset and new data traffic. In this framework, the data traffic is sent to a master node that dispatches the different requests to worker nodes. The reason why the master node collects the traffic data in our framework is that the workload can be distributed between all workers. In traditional architecture, a worker may gather a large amount of data from IoT devices while other workers do not receive much traffic data. In our framework, the master node transfers the traffic to specified workers, leading to the workload distribution between worker nodes.

In this strategy, data streams are appended to the previously sanitized dataset and after the data collection period, the master node sends an updated list of sensitive patterns with their support/confidence to each worker. The data sanitization process is performed on the worker nodes, where the HEDS4IoT algorithm is employed by each worker to modify the dedicated dataset. The sanitized dataset is sent back to the master node for outsourcing to third parties. We define a condition for the re-execution of the HEDS4IoT in a period of 1000 observations.

This algorithm converts a dataset to index lists to be used in fitness function. The sequential implementation of this conversion as well as the fitness function increase the time complexity of the PPDM process. The sequential time complexity of the indexing phase for each index list (i.e., this strategy generates two lists) is $O(N\times M\times P) = O(N^3)$, where N and M are the number of transactions and items in the dataset, and P is the number of patterns. On the other hand, the sequential time complexity of fitness function is $O(D\times L_1\times L_2) = O(N^3)$, where D is the length of the individual, L_1 and L_2 are the lengths of the first list and the second list, respectively.

To alleviate these time complexities, two mechanisms are developed, namely Parallel Indexing Engine (PIE) and Parallel Fitness Function Engine (PF2E). The PIE generates two index lists by distributing the dataset among GPU blocks. Meanwhile, the PF2E calculates the quality of individuals in a distributed manner by dividing the lists between GPU threads.

3.1. Parameter determination

This step aims to automatically specify the parameters of the number of sensitive records, i.e., S_t, and the superlative sanitization ratio, i.e., Sr_{Max}. These values are utilized to generate the initial chromosomes. The first value is defined to reduce the size of individuals in the population by selecting only the transactions that are common between sensitive patterns. To calculate S_t, initially, the dataset is scanned to obtain all the transactions associated with the sensitive patterns. Afterwards, the union of the transactions is considered as S_t (Eq. (1)).

$$S_t = \left|T_{r_1} \cup T_{r_2} \cup, \dots, \cup T_{r_{|P_S|}}\right| \tag{1}$$

The term T_r is the transactions for the pattern r and $|P_S|$ stands for the size of sensitive information set.

The parameter Sr_{Max} refers to the maximum number of transactions that are needed to be altered for the hiding of the rule with the largest support level. This parameter aims to improve diversification in the evolutionary algorithm by starting the evolution

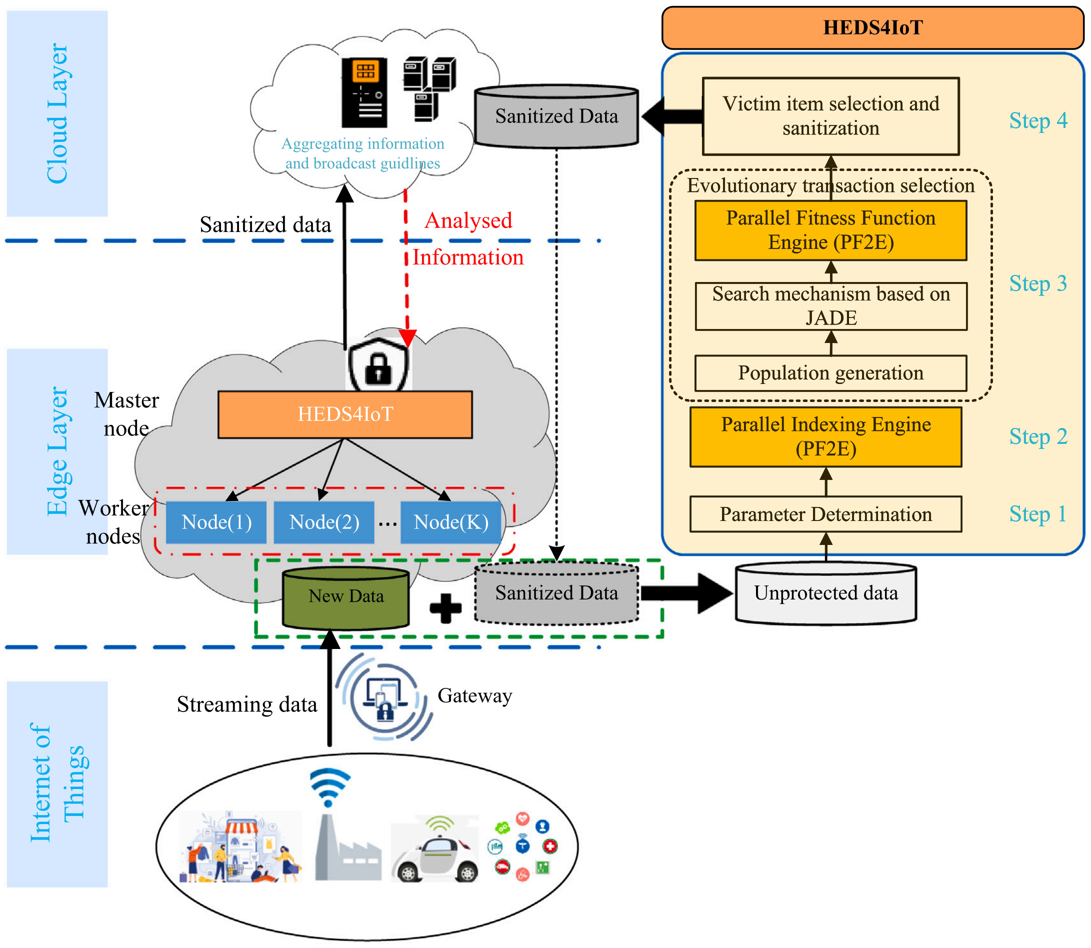


Fig. 4. Flowchart of the designed HEDS4IoT that is employed at the edge layer of the IoT platform.

process with the transactions with higher impacts on the knowledge hiding. To determine Sr_{Max}, We first calculate the sanitization ratios of all rules in the sensitive knowledge set. Then, the highest ratio is considered as Sr_{Max} (Eq. (2)):

$$Sr_{Max} = \max_{r=1,2,\dots,|P_S|}(Sr_r), \tag{2}$$

where Sr_r reflects what rate of transactions should be sanitized for the hiding of the pattern r (Eq. (3)):

$$Sr_r = \lceil n - (n \times \alpha_r) - (n \times \alpha_{min}) \rceil , \tag{3}$$

In this equation, n, α_r, and α_{min} are respectively the size of dataset, the support level of the rule r, and the support threshold, respectively.

3.2. *Parallel indexing engine*

Since we use an evolutionary algorithm, i.e., the JADE [20], for transaction selection, which is a repetitive process, different scans of a dataset must be conducted at each iteration. This leads to a complex PPDM process and reduces the efficiency in big IoT environments. To mitigate this problem, we introduce two index query lists, namely Transaction Index for Sensitive Itemsets (TISI) and Transaction Index for Non-Sensitive Itemsets (TINSI). The TISI and TINSI respectively store indexes of sensitive and non-sensitive patterns by the sensitive transactions. In this way, only the lists are used in the calculation of the fitness value, instead of the dataset, which leads to decreasing in computational and space resources when performing the fitness function. To generate these files, we design the PIE based on the CUDA computational model [21]. Fig. 5 shows the PIE mechanism, where the dataset is divided horizontally into k non-overlapping partitions of the same width, donated by $|TS|_{1:k}$. Each block of threads is assigned to a partition and each thread is tasked with processing a transaction. Each thread checks the generativity of a transaction for sensitive and non-sensitive patterns.

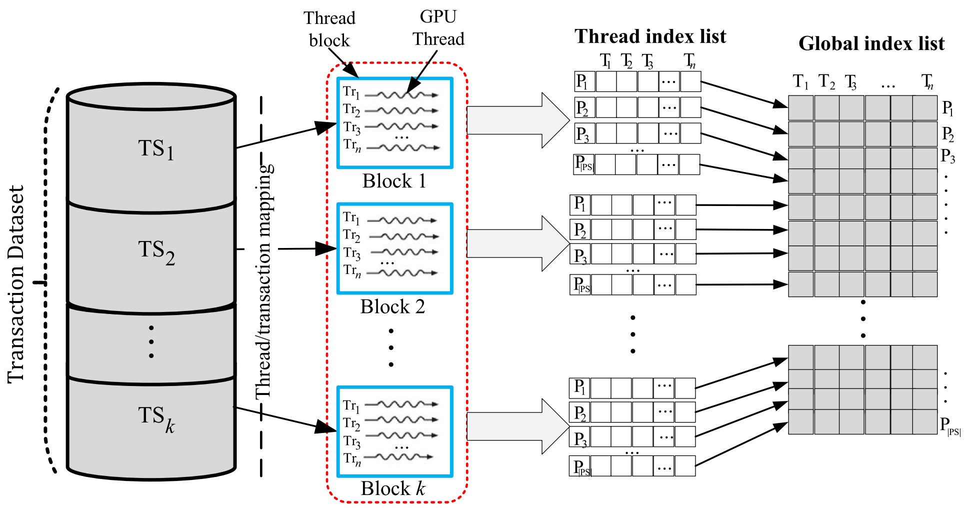


Fig. 5. Parallel indexing engine.

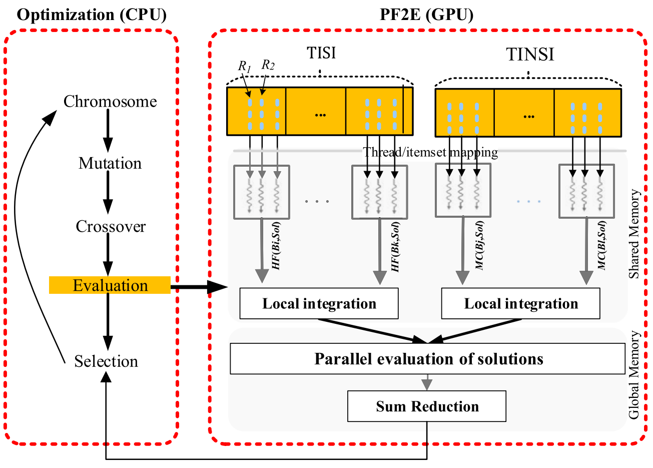


Fig. 6. Our master/slave paradigm for transaction selection.

Since all threads need to access all patterns for local index lists, the patterns need to be stored in global memory. A thread can only access one pattern at a time, leading to generating two 1D arrays with the size of $|P_S|$ and $|\neg P_S|$ where $|\neg P_S|$ is the size of non-sensitive information set. One list stores the index of sensitive patterns and the other stores the index of non-sensitive patterns. Following that, a local index table is constructed by integrating the index lists generated by each thread in a block through a local integration technique in each block. Finally, synchronization is performed between blocks to combine the index lists generated in each block. In the HEDS4IoT, only the data collected during a time window is considered for the indexing phase. In this way, the entire dataset does not needs to be scanned and the indexing process is performed in a real-time manner.

3.3. Evolutionary transaction selection

To select transactions for sanitization, we use the master–slave paradigm, where the CPU executes the master, which generates a population and initializes individuals, and the GPU executes the slave, which assess the fitness value of a chromosome in a parallel manner. Fig. 6 depicts our developed master/slave mechanism for transaction selection. In this mechanism, the JADE algorithm [20] is performed on CPU to generate a new chromosome by using mutation and crossover operators. The chromosome is then sent to GPU for fitness value computation using the PF2E mechanism. The resulting fitness value computed on the GPU devices is sent back to the CPU for individual selection step.

3.3.1. Population generation

The HEDS4IoT initializes a population of chromosomes in the first step through a developed heuristic based on S_t and Sr_{Max} calculated in the parameter determination step (Section 3.1). In our approach, the length of a chromosome, D, is equal to S_t. As we encode the transaction selection problem into binary chromosomes indicating whether a sensitive transaction is selected ("1") or not ("0"), an adaptive version of the Bernoulli technique is designed. In this technique, first, a random value between 0 and D is generated. Then, for each chosen bit of the chromosome, a value within [0,1] is randomly generated. Finally, the corresponding bit sets to "1" if the value is greater than 0.5; otherwise, it receives "0".

3.3.2. JADE based search mechanism

The HEDS4IoT algorithm evolves the population using the JADE [20], which is an adaptive version of Differential Evolution (DE) [22]. Three operators of mutation (Eq. (4)), crossover (Eq. (5)), and selection (Eq. (6)) are used in the DE to generate a new generation from the population [23].

$$v_i^G = x_{S1}^G + F.(x_{S2}^G - x_{S3}^G) \tag{4}$$

$$u_{i,j}^G = \begin{cases} x_{i,j}^G & if \quad rand_{i,j}[0,1] \le CR \\ v_{i,j}^G & \qquad otherwise \end{cases} \tag{5}$$

$$u_i^{G+1} = \begin{cases} u_i^G & if \quad f(u_i^G) < f(x_i^G) \\ x_i^G & \qquad otherwise \end{cases} \tag{6}$$

The notation F ∈[0,1] refers to the exploration length $(x_{S2} - x_{S3})$. The term j is a random value between [1,D]. The CR is the crossover probability that is in the range of [0,1].

In the DE, the mutation operation is first performed to generate a vector (v_i^G) using three randomly chosen individuals (x_{S1}^G, x_{S2}^G, and x_{S3}^G), as explained in Eq. (4). Then, the trial individual (u_i^G) is generated by the crossover operation (Eq. (5)) by crossing the individual (x_i^G) with its mutant version (v_i^G). In the final step, the individual u_i^G is replaced with x_i^G if its quality is higher than x_i^G.

A difference between the JADE and the DE is the mutation operation, where the JADE employs "DE/current-to-pbest/1" (Eq. (7)) and the DE uses "DE/rand/1" (Eq. (4)). Moreover, an automatic adjustment is made to the JADE's control parameters at every generation without the parameter settings being known in advance.

$$v_i^G = x_i^G + F.(x_{best}^G - x_i^G) + F.(x_{S1}^G - x_{S2}^G) \tag{7}$$

Non-consecutive binomial crossover is used in HEDS4IoT to produce offspring from two chromosomes. According to the fitness value, the best individual is chosen between the parents and offspring. In the HEDS4IoT, the fitness value of a chromosome is computed based on the side effects produced by the data sanitization, as described in Section 3.3.3. Each chromosome's crossover probability is generated independently during the parameter fitting phase of the JADE using normal distribution with mean μ_{CR} and standard deviation $\sigma = 0.1$. For each chromosome, a specific mutation factor is generated by using the Cauchy distribution with location parameter μ_F and scale parameter 0.1. It is necessary to adjust both μ_{CR} and μ_F at the end of each generation

3.3.3. Parallel fitness function engine

We design a fitness function considering hiding failures and lost rules to assess the quality of an individual produced by the JADE's operators, as expressed in Eq. (8):

$$f_i = w_1 \times \delta + w_2 \times \varphi, \tag{8}$$

where w_1 and w_2 are the weight of the hiding failure and the lost rules, respectively, which are set by the user of the IoT application. The notation δ refers to the number of patterns that could not be hidden, calculated by Eq. (9). While, φ stands for the number of patterns that are wrongly removed from the dataset, computed using Eq. (10).

$$\delta = \sum_{1}^{|P_S|} r_i state \tag{9}$$

$$r_i state = \begin{cases} 1, & \text{if } \alpha_{new}(r_i) \ \ge \alpha \\ 0, & \text{otherwise} \end{cases}$$

$$\alpha_{new}(r_i) = \alpha(r_i) + \left(\frac{Freq_{r_i}}{n} \right),$$

In these equations, r_i is the pattern i, $\alpha(r_i)$ and $\alpha_{new}(r_i)$ refer to the current support level and its new support level if we sanitize the chosen transactions, respectively. The term $freq_{r_i}$ is the support level of r_i in the transactions chosen by the individual.

$$\varphi = \sum_{1}^{|\neg P_S|} r_i state \tag{10}$$

$$r_i state = \begin{cases} 1, & \text{if } \alpha_{new}(r_i) < \alpha_{min} \\ 0, & \text{otherwise,} \end{cases}$$

Calculations of δ and φ need a high execution time because two scans of the dataset are required for each sensitive transactions to find the hiding failure (i.e., δ) and the lost rules (i.e., φ). As a huge amount of data in IoT environments is generated at breakneck speed, this computation is overwhelmingly time-consuming. We have developed the PF2E mechanism to accelerate the fitness value computation using a CUDA implementation on the edge layer of IoT. In the PF2E mechanism, the TISI and TINSI lists are distributed across the GPU blocks. The TINSI list is often larger than the TISI list because the non-sensitive patterns often outnumber the sensitive patterns. Therefore, the majority of the GPU blocks are allocated to the TINSI list. In the PF2E, we assign 10% of the GPU blocks to the TISI list and the remaining blocks to the TINSI list.

Fig. 6 (in the right box) explains how the PF2E works. The blocks assigned to the TISI list compute the hidden failures, while the blocks assigned to the TINSI list compute the lost rules. A 2D table is generated by each thread to store the transaction indices and their conflict degree. All generated tables in the shared memory are combined using a local integration mechanism in each block. The tables in the blocks are sent to the global memory to produce two tables for the hiding failure and the lost rules. Based on these two tables in the global memory, the fitness value is computed via a global sum reduction technique and is sent to the CPU.

3.4. Victim item selection and sanitization

In order to select items to be removed from the selected transactions, we a conflict degree measure. The conflict degree for an item means the percentage of patterns containing the item. In our algorithm, the item with the minimum impact on the non-sensitive patterns and the maximum influence to the sensitive information set is selected as the victim item (Eq. (11)).

$$Conf_i = \frac{Conf_{i,S}}{Conf_{i,NS}} \tag{11}$$

The notations $Conf_{i,S}$ and $Conf_{i,NS}$ stand for the number of times that the item i appears in the sensitive and non-sensitive patterns, respectively. The conflict degree of all items in a sensitive pattern is computed and the item with the maximum degree is chosen to be removed from the selected transactions. This process continues until the pattern's support level is reduced below the minimum support level.

3.5. Pseudocode of HEDS4IoT

Algorithm 1 shows the pseudocode for the HEDS4IoT works. The HEDS4IoT receives three sets of datasets, sensitive patterns set, and non-sensitive patterns set as well as three parameters of Minimum Threshold, Population size, and Maximum number generations of as inputs. After performing four steps of our algorithm, a sanitized dataset is generated for sharing in the IoT platform. In Lines 1–7, two parameters of S_t and Sr_{max} are calculated. Lines 8–18 implement the PIE strategy using CUDA programming with the aim of generating TISI and TINSI lists. A population is created using S_t and Sr_{max} in Line 20. Then, the JADE technique is performed to produce a new individual in Line 22. The fitness value of the individual is assessed using the PF2E in Lines 23–31. Finally, sensitive patterns are hidden by selecting victim items and removing the items from the selected transactions (Lines 34 to 44).

3.6. Complexity analysis of the algorithm

In this section, we analyze the time complexity of the HEDS4IoT algorithm using the main steps in Algorithm 1. **Step 1:** The time complexity is $O(n{\times}P_S)$, where n and P_S are the number of transactions and sensitive patterns, respectively. **Step 2:** The PIE requires $O(n/Tr{\times}P$ operations, where Tr is the number of threads in each block. **Step 3:** The time complexity of the evolutionary transaction selection step is $O(n)$, i.e., the aggregate complexity of the population generation ($O(S_t)$), search mechanism ($O(pop)$), and PF2E ($O(n/Tr)$), where pop is the population size. **Step 4:** The time complexity of this step is $O(n/Tr)$.

4. Experimental results

In this section, we review the evaluation of the proposed approach in three groups of experiments: (1) side effects in comparison with some state-of-the-art work, including ABC4ARH [14], PACO2DT [6], ACS2DT [15] (Section 4.2), (2) speedup at both kernel-level and application-level (Section 4.3). At the kernel level, speedups of the PIE and PF2E are compared with their sequential versions. At the application level, the speedup of the HEDS4IoT is evaluated over its sequential version (named HEDS4IoTSeq). (3) Performance improvement of the HEDS4IoT is compared with ABC4ARH [14], PACO2DT [6], ACS2DT [15] (Section 4.4).

4.1. Experiment setup

We simulated our edge computing-enabled PPDM using the YAFS [24]. The main features of the implementation environment are shown in Table 1. In all experiments, the number of blocks and threads in each block is set to 256 and 512, respectively. As our implementation system includes the NVIDIA Tesla P100 PCIE GPU model, it provides GPU libraries to make use of CUDA programming.

In our experiments, we used both real and synthetic datasets. We selected four real datasets of ChainStore, Kosarak, PowerC, and Susy, which have been obtained from SPMF repository[1]. The ChainStore dataset includes customer transactions from a major

[1] http://www.philippe-fournier-viger.com/spmf/

Algorithm 1: High-performance Evolutionary Data Sanitization for IoT (HEDS4IoT)

```
Input: $DB$ (Database), $P_S$ (Sensitive Patterns), $\neg P_S$ (Non-sensitive Patterns), $\alpha_{min}$ (Minimum Threshold),
$pop$ (Population size), $G$ (Maximum number of generations)
Output: ($DB'$ (The sanitized dataset)
1: procedure PARAMETER DETERMINATION (DB, $P_S$)                                     ▷ STEP 1 (Sec. 3.1)
2:     $ST \leftarrow$ Index of sensitive transactions;
3:     $S_t \leftarrow$ The number of transactions in ST; (Eq. (1))
4:     Foreach transactions $\in ST$;
5:         $S_{ri} = \lceil n - (n \times \alpha_i) - (n \times \alpha_{min}) \rceil$; (Eq. (3))
6:     return $Sr_{Max} = \max_{i=1,2,\dots,|P_S|}(Sr_i)$; (Eq. (2))
7: end procedure

8: procedure PIE(DB, $P_S$,$\neg P_S$)                                              ▷ STEP 2 (Sec. 3.2)
9:     $idx = blockIdx.x \times blockDim.x + threadIdx.x$;
10:    Foreach $t \in ST[blockIdx.x]$;
11:         TISI[blockIdx.x], TINSI[blockIdx.x] ← Generate initial index lists;
12:        Foreach $I_j$
13:            IF $r_j[blockIdx.x] \in t((i \times blockDim.x) + idx)$
14:                IF $r_j \in P_S$
15:                    $TISI_{i,j}[blockIdx.x] \leftarrow j$;
16:                IF $r_j \in \neg P_S$;
17:                    $TINSI_{i,j}[blockIdx.x] \leftarrow j$;
18: end procedure

19: procedure TRANSACTION SELECTION($S_t$, $S_r$, TISI,TINSI)                          ▷ STEP 3 (Sec. 3.3)
20:    Generate initial population using $S_t$ and $Sr_{max}$;                       ▷ STEP 3-1 (Sec. 3.3.1)
21:    For $g = 1$ to $G$;                                                           ▷ STEP 3-2 (Sec. 3.3.2)
22:        Evolve the population using JADE; (Eqs. (5),(7),(6))
23:        Recuperate Sol from CPU;                                                  ▷ STEP 3-3 (Sec. 3.3.3)
24:        $idt = lockIdx.x \times blockDim.x + threadIdx.x$;
25:          For $i = 1$ to $|ST|$
26:            IF $Sol[blockIdx.x] \in TISI((i \times blockDim.x) + idt)$
27:                $\delta \leftarrow$ Compute hiding failure; (Eq. (9))
28:            IF $Sol[blockIdx.x] \in TINSI((i \times blockDim.x) + idt)$;
29:                $\varphi \leftarrow$ Compute misses cost; (Eq. (10))
30:        Fitness(Buff[blockIdx.x]=$w_1 \times \delta[blockIdx.x] + w_2 \times \varphi[blockIdx.x]$;
31:        Cudamemcpy(fitness(Buff[blockIdx.x]), cudaMemcpyDeviceToHost)
32:    return Fitness value.
33: end procedure

34: procedure ITEM SELECTION AND SANITIZATION($P_S$,$\neg P_S$, Sol)                   ▷ STEP 4 (Sec. 3.4)
35:    Candidates ← all common items in $P_S$;
36:    Foreach item $i \in$ Candidates
37:        $Conf_i, S, Conf_{i,NS} \leftarrow 0$;
38:        Foreach $sensitive\ itemset \in P_S$;
39:              IF item $i \in P_{Si} \rightarrow\rightarrow Conf_{i,S} = Conf_{i,S} + 1$;
40:        Foreach $non-sensitive\ itemset \in \neg P_S$
41:            IF item $i \in \neg P_{Si} \rightarrow\rightarrow Conf_{i,NS} = Conf_{i,NS} + 1$;
42:        Conf.append($Conf_{i,S}/Conf_{i,NS}$);
43:    return $Conf$.
44: end procedure
```

grocecy store chain in California, USA. Th Kosarak dataset consists of transactions from click-stream data from an hungarian news portal. The PowerC dataset is about household electric power consumption. It was prepared using the original dataset[2]. The Susy dataset is related to particles detected using a particle accelerator. The first three datasets have around 1 Million records and a similar average length of transactions (around 7–8) and all three are sparse. On the other hand, the Susy dataset is larger and denser than the others, with around 5 Million records and a density ratio of 0.1.

PPDM algorithms are primarily affected by density. Since the real datasets do not contain the same number of transactions and items and have different density ratios, we generated five datasets with density ratios between 0.001 and 0.1. In these datasets, the number of transactions is constant at 1 million while the length of transactions varies by the maximum length of 10 K items according to the defined ratios. Dataset characteristics are shown in Table 2 in terms of the number of transactions, items, and the average transaction length for each dataset. We selected one dense real dataset (i.e., Susy) and three sparse real datasets.

We investigate the HEDS4IoT's performance in terms of side-effects and efficiency at the both application-level and kernel-level. Regarding the side-effects, hiding failure, misses cost and database similarity at the transaction level and item level are considered. The hiding failures represents how many sensitive patterns are failed to be hidden (Eq. (12)). By calculating the missed costs, we can

[2] https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

Table 1
Characteristics of the implementation environment.

	Characteristic	Type
General	Platform	Linux
	Programming language	Python
	Parallel programming	CUDA Version: 10.1
GPU	GPU Model	Tesla P100-PCIE-16GB
	Maximum Threads Per Block	1024
	Maximum Blocks	1024*64 = 65536
	Maximum Memory Per Block	49152
	Memory	16 GB
CPU	CPU Model	Intel(R) 2.20 GHz
	CPU(s)	2
	Thread(s) per core	2
	L1 cache	32K
	L2 cache	256K
	L3 cache	56320K
	RAM	13 GB
parameters	Population size	100 chromosomes
	Maximum number of generations (G)	2000
	Crossover Ratio (CR)	0.1
	Importance of hiding failure (w_1)	0.9
	Importance of lost rule (w_2)	0.2

Table 2
Characteristics of the datasets (M = Million, K = thousand).

Name	$\|Trans.\|$	$\|Items\|$	Avg. Length	Density
ChainStore	1.1M	46K	7.23	0.00015 (Sparse)
Kosarak	1M	41.2K	8.10	0.0002 (Sparse)
PowerC	1M	140	7.0	0.05 (Sparse)
Susy	5M	190	19.0	0.1 (Dense)
Syn0.001	1M	10K	10.0	0.001 (Dense)
Syn0.005	1M	10K	50.0	0.005 (Dense)
Syn0.01	1M	10K	100.0	0.01 (Dense)
Syn0.05	1M	10K	500.0	0.05 (Dense)
Syn0.1	1M	10K	1000.0	0.1 (Dense)

determine the percentage of non-sensitive information that are wrongly removed from the sanitized dataset Eq. (13). Similarity at the item level (DS_{item}) and transaction-level (DS_{trans}) are the frequency of items and transactions that are changed after sanitization, as shown in Eq. (14) and (15), respectively [14].

$$Hiding\ Failure = \frac{\left|P'_S\right|}{\left|P_S\right|} \tag{12}$$

$$Misses\ Cost = \frac{\left|\neg P_S\right| - \left|\neg P'_S\right|}{\neg P_S} \tag{13}$$

In these equations, $|P'_S|$ and $|P_S|$ are the size of the sensitive inform sets in the modified and the original database, respectively. On the other hand, $|\neg P_S|$ and $|\neg P'_S|$ stand for the number of non-sensitive information in the initial dataset and maintained in the sanitized dataset.

$$DS_{item} = \frac{\sum_{i=1}^{n} f_{DB}^{i} - \sum_{i=1}^{n} f_{DB'}^{i}}{\sum_{i=1}^{n} f_{DB}^{i}} \tag{14}$$

$$DS_{trans} = \frac{|T'|}{|T|} \tag{15}$$

The notations, n is the number of unique elements in the dataset, f_{DB}^{i} and $f_{DB'}^{i}$ are the support of item i in the initial dataset (DB) and the sanitized dataset (DB'), respectively. $|T'|$ refers how many transactions have been modified.

The parameters for the HEDS4IoT are listed in the last row in Table 1. These values were determined according to extensive experiments and the values that resulted in the best performance were considered as the final values. Regarding the ABC4ARH's parameter values, the number of bees is 100 and the limitation condition is $SN * D$. The values of w_1 and w_2 were considered to be 0.9 and 0.2, respectively, as in the HEDS4IoT algorithm.

Table 3
The comparison of the HEDS4IoT and the other three algorithms.

Algorithm	Dataset	HF	MC	DS_T	DS_I
HEDS4IoT	ChainStore	0.086	0.031	0.906	0.969
	Kosarak	0.093	0.032	0.881	0.954
	PowerC	0.098	0.034	0.911	0.986
	Susy	0.109	0.042	0.926	0.974
	Syn0.001	0.081	**0.024**	0.941	0.973
	Syn0.005	0.073	0.037	0.938	0.978
	Syn0.01	0.071	0.041	0.938	0.982
	Syn0.05	0.069	0.044	0.936	0.981
	Syn0.1	0.07	0.048	0.933	0.983
	Average	**0.083**	**0.036**	**0.923**	**0.975**
PACO2DT [6]	ChainStore	0.078	0.041	0.938	0.893
	Kosarak	0.075	0.036	0.942	0.94
	PowerC	**0.063**	0.049	0.946	0.959
	Susy	0.075	0.074	**0.953**	0.929
	Syn0.001	0.086	0.029	0.934	0.934
	Syn0.005	0.065	0.032	0.948	0.941
	Syn0.01	0.059	0.041	0.952	0.944
	Syn0.05	0.056	0.048	0.961	0.952
	Syn0.1	0.051	0.053	0.968	0.956
	Average	**0.067**	**0.045**	**0.949**	**0.938**
ABC4ARH [14]	ChainStore	0.106	0.037	0.868	0.97
	Kosarak	0.103	0.026	0.897	0.967
	PowerC	0.114	0.038	0.891	0.978
	Susy	0.124	0.052	0.901	0.97
	Syn0.001	0.088	0.042	0.876	0.971
	Syn0.005	0.085	0.028	0.884	0.978
	Syn0.01	0.083	0.03	0.893	0.983
	Syn0.05	0.087	0.033	0.905	0.986
	Syn0.1	0.071	0.038	0.921	**0.989**
	Average	**0.095**	**0.038**	**0.892**	**0.976**
ACS2DT [15]	ChainStore	0.112	0.036	0.857	0.874
	Kosarak	0.106	0.031	0.873	0.921
	PowerC	0.124	0.04	0.914	0.935
	Susy	0.119	0.046	0.923	0.948
	Syn0.001	0.037	0.042	0.932	0.938
	Syn0.005	0.096	0.041	0.936	0.942
	Syn0.01	0.095	0.048	0.94	0.94
	Syn0.05	0.091	0.053	0.942	0.946
	Syn0.1	0.081	0.062	0.941	0.953
	Average	**0.102**	**0.042**	**0.917**	**0.933**

4.2. Evaluation results for side effects

Table 3 compares side-effects (i.e., Hiding Failure (HF) and Misses Cost (MC)) and database similarity (database similarity from transaction (DS_T) and item levels (DS_I)) of the PACO2DT [6], ABC4ARH [14], and ACS2DT [15]. Based on our experiments, we found that the parameter values for sensitive pattern = 0.01 and Minimum Support Threshold (MST) = 0.1 can be challenging when evaluating side effects. In this experiment, we do not require a large number of sensitive patterns; therefore, we chose a moderate ratio of sensitive patterns and MST. The PACO2DT algorithm has the lowest error in sensitive knowledge hiding. It averages 6.7%, compared to 8.3% and 9.5% and 10% for the HEDS4IoT, ABC4ARH, and ACS2DT algorithms, respectively. The PACO2DT yielded the lowest hiding failure, averagely 6.7%, compared to 8.3% and 9.5% and 10% for the HEDS4IoT, ABC4ARH, and ACS2DT algorithms, respectively. The averaged misses cost for the HEDS4IoT was 3.6% that was the best misses cost value. Regarding database similarity, the PACO2DT algorithm was the best algorithm in term of transaction-level similarity, which is 94.9%, while, the HEDS4IoT and ABC4ARH provided the most similar item-level datasets, which are about 97.5%. Overall, as a result of using item deletion techniques in the HEDS4IoT and ABC4ARH, lower misses cost and item-level similarity were introduced. While, transaction-deletion-based algorithms, such as PACO2DT and ACS2DT, made them more effective in the hiding failure minimization and transaction-level similarity maximization.

4.3. Evaluation results for speedup

In this subsection, first, the speedup of the two GPU-enabled engines (i.e., PIE and PF2E) is evaluated in comparison with their sequential versions. We then examine the impact of various influencing factors (e.g., support threshold, sensitive patterns ratio, and density ratio) on the computational scalability of the HEDS4IoT framework.

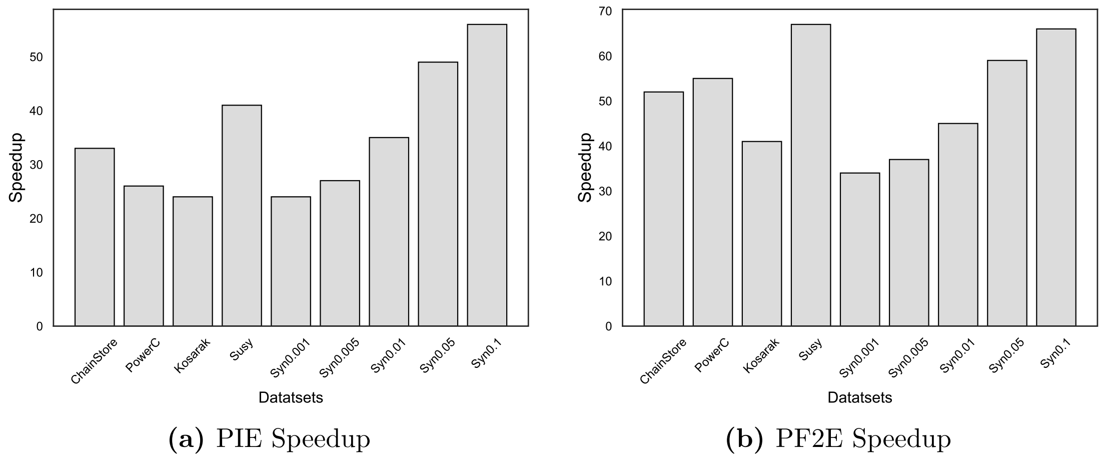


Fig. 7. Kernel-level speedups of the PIE and PF2E over their sequential versions (i.e., CPU implementation) on different datasets ($|Blocks| = 256$, $|Threads| = 512$, MST = 0.1, Sensitive Patterns = 0.01).

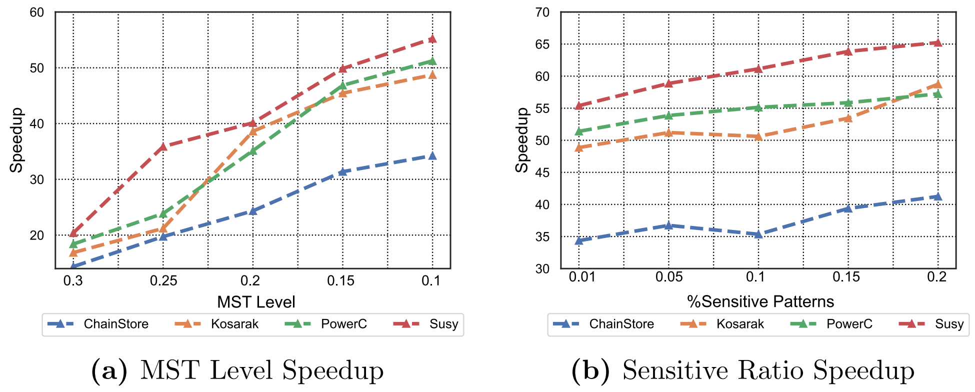


Fig. 8. Application-level speedups of HEDS4IoT over the scalar version (i.e., HEDS4IoTseq) using real datasets in terms of MST and sensitive patterns ratios (|Blocks| = 256, |Threads| = 512, Sensitive Patterns = 0.01).

Speedup for the Parallel Engines: Fig. 7 compares speedup of the two PIE and PF2E over their sequential versions. The PIE and PF2E mechanisms could complete the steps of index list generation and fitness function with a significantly higher speed using GPU platform than their CPU-based versions, achieving average speedups of 33.5x and 53.7x, respectively.

Speedup for the Influencing Factors: These experiments demonstrate the scalability of the HEDS4IoT in comparison with HEDS4IoTSeq when varying the values of influencing parameters, including MST, the ratio of sensitive patterns, and density ratio. Fig. 8 shows how the developed algorithm can be scalable for the real datasets by changing the ratios of MST level 8(a) and sensitive patterns 8(b). It is obvious that varying the MST level has a greater impact on the speedup of the HEDS4IoT and significantly increases the speedup compared to its sequential version. On the other hand, increasing the ratio of sensitive patterns has less impact on the speedup of the HEDS4IoT algorithm. This is because the number of patterns is constant with varying the sensitive pattern ratio and only the number of sensitive and non-sensitive patterns are modified. This only influences the length of chromosomes.

The averaged HEDS4IoT's speedups are 33.5 and 35.8x for MST level and sensitive patterns ratio comparisons, respectively. Fig. 9 shows the speedup for the HEDS4IoT executing on the GPU device relative to the baseline model executing on the CPU host for the synthetic datasets. Significant speedup can be achieved by offloading sequential operations to GPUs, so that we can observe an average speedup of 55x and 51.3x without accuracy degradation over the baseline fully implemented inside the CPU.

Fig. 10 demonstrates how our high performance GPU-aided mechanisms could obtain high speedup when increasing the density ratio. The HEDS4IoT achieved higher speedup on denser datasets compared to sparser datasets because more patterns with longer length are produced for a denser dataset than a sparse version. This demonstrates that the density has a direct effect on the efficiency of the PPDM algorithms.

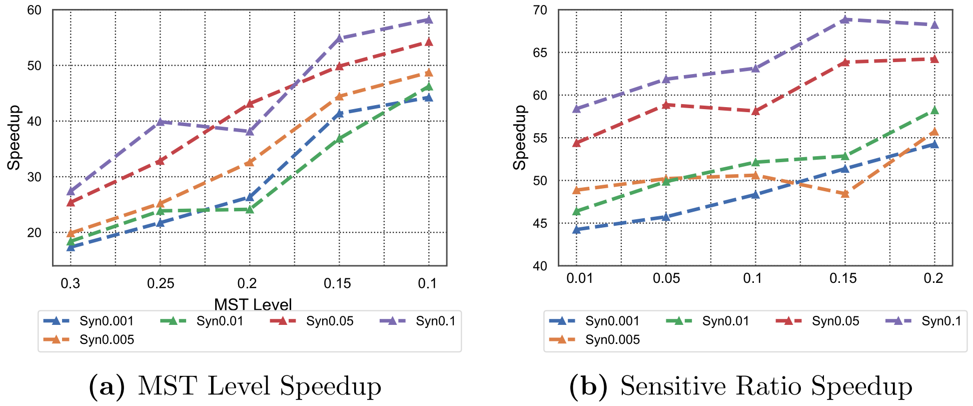


Fig. 9. Application-level speedups of HEDS4IoT over the scalar version (i.e., HEDS4IoTseq) using synthetic datasets in terms of MST and sensitive patterns ratios (|Blocks| = 256, |Threads| = 512, Sensitive Patterns = 0.01).

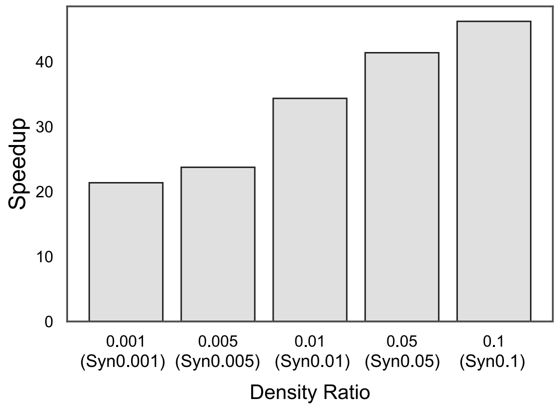


Fig. 10. Application-level speedups of HEDS4IoT over the scalar version (i.e., HEDS4IoTseq) in terms of density ratios ($|Blocks|$ = 256, $|Threads|$ = 512, MST = 0.1, Sensitive Patterns = 0.01).

4.4. Performance improvement comparisons with state-of-the-art algorithms

Similar to Section 4.3 that evaluated the HEDS4IoT's speedup over HEDS4IoTseq from prospective of effective parameters (e.g., density ratio, MST, and sensitive patterns ratio), experiments of this section provides such evaluations but in comparison with some state-of-the-art PPDM algorithms, including ABC4ARH [14], PACO2DT [6], and ACS2DT [15], using four real datasets.

Speedup for MST level: Fig. 11 shows the speedup with different MST levels. The HEDS4IoT demonstrates improved performance over the other algorithms with a change in MST ratio from 0.3 to 0.1, as shown in this figure. This is because as the MST level is decreased, the number of patterns to be extracted increases, leading to an increase in sensitive and non-sensitive patterns. Therefore, sequential operations employed in the other algorithms requires a large number of computational resources to calculate the fitness value from a huge number of patterns while the HEDS4IoT evaluates all calculation in a parallel way. For a case, the HEDS4IoT could provide a speedup of 17x for MST = 0.3, while this speedup reached 50x for MST = 0.1.

Speedup for sensitive patterns ratio: Fig. 12 provides an analysis of the effects of a constant value of MST = 0.1 and different ratios of the sensitive patterns. The HEDS4IoT algorithm achieves an averaged speedup of 50x over the others. This value is 65x for the Susy dataset, which is a larger and denser dataset. On average, the speedup of the HEDS4IoT approach was 56.1x, 60x, and 63.4x compared to the PACO2DT, ABC4ARH, and ACS2DT, respectively. We can see that the speedup of the HEDS4IoT increases when we vary the ratio of sensitive patterns. However, this speedup is relatively small. This is because with a constant number of patterns, only the balance between sensitive and non-sensitive information sets is altered and the main information set is not changed, resulting in a constant execution time for the PIE mechanism for different sizes of sensitive patterns set. As the ratio of sensitive patterns increases, the length of individuals becomes the only factor that influences the ratio, leading to more computations for the hiding failure and the misses cost.

Speedup for density ratio: Fig. 13 investigates how the density ratio factor affects the speedup of our developed algorithm than the other algorithms. We generated five synthetic datasets based on five density ratios. According the results, the HEDS4IoT could provide significant speedup for all density ratios. In this experiment, the HEDS4IoT achieved an average speedup of 28x, 33.9x, and 42.5x over the PACO2DT, ABC4ARH, and ACS2DT, respectively.

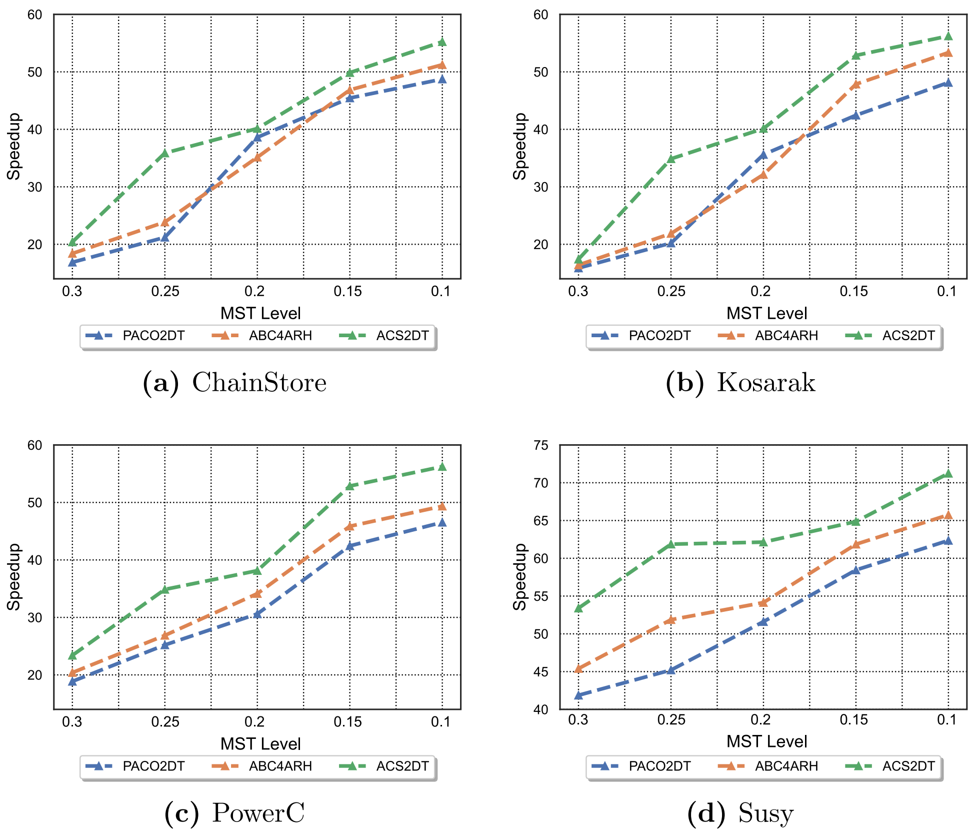


Fig. 11. Performance improvement of GPU implementation of HEDS4IoT over sequential versions of other privacy preserving data mining algorithms using four real datasets (MST = 0.1 to 0.3 and Sensitive Patterns = 0.01).

Despite the efficiency of HEDS4IoT, one of its limitations is that in the PF2E phase, the CPU platform and the GPU platform cannot be executed simultaneously. They are executed serially, i.e., a new individual is generated on CPU and then its quality is evaluated on the GPU. This can lead to some waste of resources and to processing components waiting for each other.

5. Related work

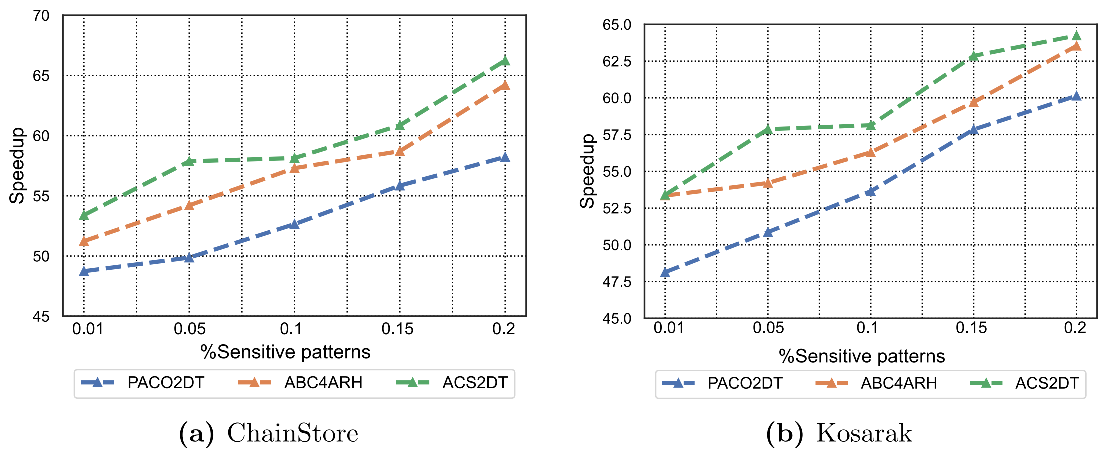
This section first describes related work in GPU-assisted PPDM methods 5.1 and data sanitization approaches 5.2 are then analyzed.

5.1. GPU platform for privacy preserving

The acceleration of protected training for machine learning using GPU has considered been recently. A GPU-aided convolutional neural network on encrypted data was developed in [8]. Goten [25] is a GPU implementation of a query protection mechanism in a Trusted Execution Environment (TEE). Tan et al. [9] executed all operations in deep learning models on GPU devices and developed an encrypted multi-party computation. Hashemi et al. [10] introduced the DarKnight framework by encoding the input data in a matrix format in the TEE and offloading it to the GPU.

5.2. Data sanitization algorithms

Data sanitization, as a subset of PPDM, is used for hiding frequent patterns. Data sanitization techniques can be divided into two categories: heuristic and evolutionary. Both groups focused on how to select sensitive transactions and items for sanitization in such a way that side effects are minimized. The heuristic approaches use predefined criteria (e.g., transaction length and item frequency) [2,26–28]. Srivastava et al. [29] proposed a privacy-preserving utility mining mechanism for IoT applications and selected items that have the most conflict in the sensitive patterns for sanitization. A database extension method was proposed by Li et al. [30] to reconstruct a database by changing the sensitivity level of the non-sensitive patterns. As a shortcoming, the heuristic methods are ineffective in providing minimum side effects in the resulting sanitized data. An approach for PPDM in the



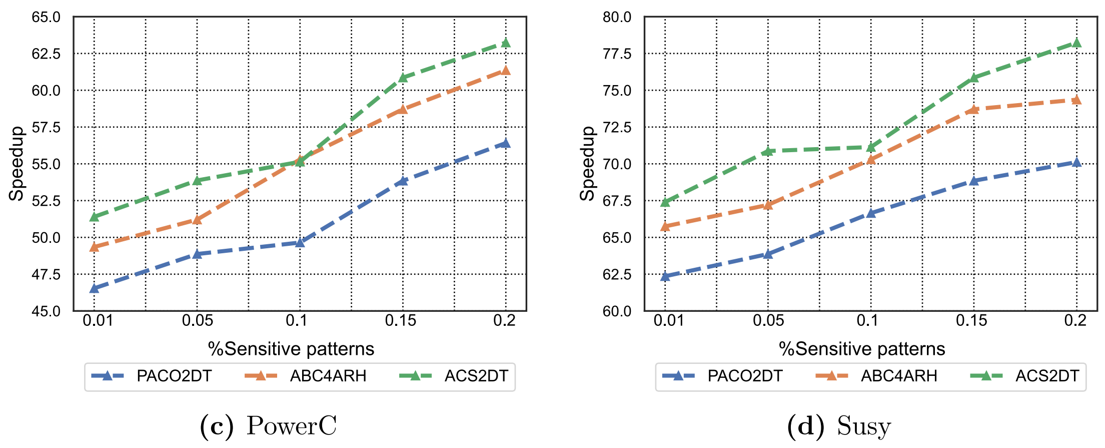


Fig. 12. Performance improvement of the proposed approach over sequential versions of other privacy preserving data mining algorithms using four real datasets (MST = 0.1 and Sensitive Patterns = 0.01 to 0.2).

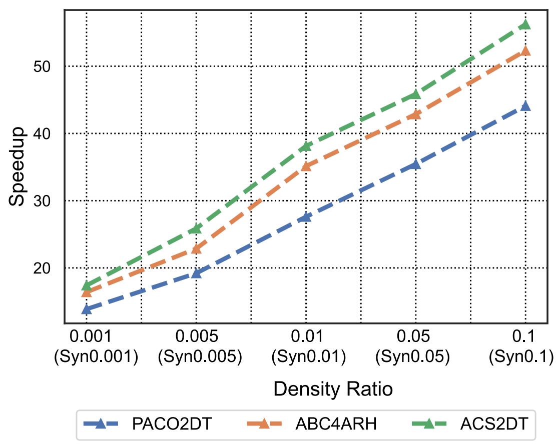


Fig. 13. Speedup of the HEDS4IoT over other methods in terms of different density ratios ($|Blocks| = 256$, $|Threads| = 512$, MST = 0.1, Sensitive Patterns = 0.01).

vertically partitioned healthcare data was developed in [31] for discovering the correlation related to disease and preserving the privacy of the patients.

On the other hand, evolutionary data sanitization approaches use the potentially more effective strategy than the heuristic approaches and can find near-optimal solutions by formulating the hiding problem in terms of individuals. Different evolutionary algorithms have been utilized or optimized for data sanitization, such as Genetic Algorithm (GA) in [32,33], Particle Swarm Optimization (PSO) in [34], Ant Colony Optimization (ACO) in [15], cuckoo optimization in [35], Tabu-genetic algorithm [36], and firefly optimization algorithm in [37]. Suma and Shobha [38] employed the Fractional-Salp swarm algorithm for data sanitization. The random key generation of the sanitization process was performed using the proposed Fractional-SSA algorithm by arbitrarily initializing various keys. The fitness was obtained using success and failure scenarios. A two-phase framework was presented in [39]. the cuckoo search algorithm was deployed for optimal key generation in data sanitization phase to preserve the sensitive data. In the restoration phase, the sanitized data can be restored by exploiting the same key.

Some studies improved base evolutionary algorithms for data sanitization purposes through hybridization [40] or enhancement [14]. For example, Navale and Mali combined the GA and the crow search algorithm to hide frequent patterns [40]. Telikani et al. [14] enhanced a binary artificial bee colony approach for sensitive transaction selection while using a heuristic for victim items. The authors extended this work for big data processing using the GPU platform [41]. The fitness function computation step was parallelized on GPU threads to speed up the PPDM process. A difficult problem in the PPDM techniques is defining thresholds for minimum support. Wu et al. [42] defined thresholds based on the length of the sensitive patterns. This algorithm needs to delete a lot of transactions to reduce side-effects, leading to lose more essential knowledge.

Recently, some studies have focused on Multi-Objective Optimization (MOO) in data sanitization problem. In this case, all three common side effects of data sanitization, including hiding failure, lost information, and artificial knowledge, are considered as objectives. Lin et al. [6] employed multi-objective ant colony optimization with a transaction deletion technique to secure confidential and sensitive information. The authors also developed some other data sanitization mechanisms based on MOO [7,43].

6. Conclusions

IoT environments and wireless networks that allow information to be exchanged between devices expose confidential and sensitive information to security and privacy breaches. Fortunately, data sanitization mechanisms can mitigate these risks before data sharing and distribution. For large-scale IoT data applications, privacy protection mechanisms can however be extremely time-consuming, especially when evolutionary algorithms are involved. This paper presents an evolutionary PPDM algorithm called HEDS4IoT that uses a GPU platform to accelerate the data sanitization process in edge layer IoT environments. A GPU-based architecture was used with two contributions. The first is the development of a GPU-based parallel indexing engine to provide index retrieval lists for sensitive transactions and the elements to be used in the fitness calculation phase. The second contribution of the GPU is in the parallelization of the fitness computation, where the index lists are processed using a parallel implementation on GPU devices.

A total of four real benchmark datasets as well as five synthetic datasets were used to evaluate the performance of the HEDS4IoT in terms of side effects and speedup. Our findings demonstrated that the HEDS4IoT can provide promising side-effect reduction and high speedup. The HEDS4IoT was able to achieve a speedup of 47.7x over other PPDM algorithms. In addition, our developed GPU-enabled engines could accelerate their corresponding tasks 43.6x more than CPU-based implementations. As a result of distributing large index lists and transactions across threads, the HEDS4IoT provided higher scalability for dense datasets compared to sparse datasets. According to our results, IoT can benefit from certain privacy and security advantages provided by our framework. These benefits are the privacy protection in real-time using the application of GPU platform in edge computing of IoT networks and the minimization of side-effects using evolutionary algorithms.

In the future, PPDM can be implemented using other big data processing mechanisms such as Spark for large datasets based on the success of the hybrid CPU/GPU platform in accelerating the hiding process. Furthermore, the PF2E strategy can be designed to utilize both CPU and GPU platforms simultaneously, improving resource utilization and convergence.

Declaration of competing interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Data availability

Data will be made available on request.

References

[1] C.W. Tsai, C.F. Lai, M.C. Chiang, L.T. Yang, Data mining for internet of things: A survey, IEEE Commun. Surv. Tutor. 16 (1) (2013) 77–97.
[2] A. Amiri, Dare to share: Protecting sensitive knowledge with data sanitization, Decis. Support Syst. 43 (1) (2007) 181–191.
[3] D. Huang, Q. Gan, X. Wang, M.R. Ogiela, X.A. Wang, Privacy preserving IoT-based crowd-sensing network with comparable homomorphic encryption and its application in combating COVID19, Internet Things 20 (2022) 100625.
[4] W. Lu, Z. Ren, J. Xu, S. Chen, Edge blockchain assisted lightweight privacy-preserving data aggregation for smart grid, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1246–1259.
[5] A. Telikani, A. Shahbahrami, Data sanitization in association rule mining: An analytical review, Expert Syst. Appl. 96 (2018) 406–426.
[6] J.C.W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, M. Aloqaily, Privacy-preserving multiobjective sanitization model in 6g IoT environments, IEEE Internet Things J. 8 (7) (2020) 5340–5349.
[7] J.C.W. Lin, J.M.T. Wu, P. Fournier-Viger, Y. Djenouri, C.H. Chen, Y. Zhang, A sanitization approach to secure shared data in an iot environment, IEEE Access 7 (2019) 25359–25368.
[8] D. Takabi, R. Podschwadt, J. Druce, C. Wu, K. Procopio, Privacy preserving neural network inference on encrypted data with GPUs, 2019, arXiv preprint arXiv:1911.11377.
[9] S. Tan, B. Knott, Y. Tian, D.J. Wu, CRYPTGPU: Fast privacy-preserving machine learning on the GPU, 2021, arXiv preprint arXiv:2104.10949.
[10] H. Hashemi, Y. Wang, M. Annavaram, DarKnight: An accelerated framework for privacy and integrity preserving deep learning using trusted hardware, in: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, 2021, pp. 212–224.
[11] M.G. Herabad, Communication-efficient semi-synchronous hierarchical federated learning with balanced training in heterogeneous IoT edge environments, Internet Things (2022) 100642.
[12] M.Z. Uddin, A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system, J. Parallel Distrib. Comput. 123 (2019) 46–53.
[13] Y. Sun, T. Fei, X. Li, A. Warnecke, E. Warsitz, N. Pohl, Real-time radar-based gesture detection and recognition built in an edge-computing platform, IEEE Sens. J. 20 (18) (2020) 10706–10716.
[14] A. Telikani, A.H. Gandomi, A. Shahbahrami, M.N. Dehkordi, Privacy-preserving in association rule mining using an improved discrete binary artificial bee colony, Expert Syst. Appl. 144 (2020) 113097.
[15] J.M.T. Wu, J. Zhan, J.C.W. Lin, Ant colony system sanitization approach to hiding sensitive itemsets, IEEE Access 5 (2017) 10024–10039.
[16] S. Li, S. Zhao, G. Min, L. Qi, G. Liu, Lightweight privacy-preserving scheme using homomorphic encryption in industrial internet of things, IEEE Internet Things J. 9 (16) (2021) 14542–14550.
[17] M. Ma, D. He, N. Kumar, K.-K.R. Choo, J. Chen, Certificateless searchable public key encryption scheme for industrial internet of things, IEEE Trans. Ind. Inform. 14 (2) (2017) 759–767.
[18] C. Yin, J. Xi, R. Sun, J. Wang, Location privacy protection based on differential privacy strategy for big data in industrial internet of things, IEEE Trans. Ind. Inform. 14 (8) (2017) 3628–3636.
[19] Z. Chang, S. Liu, X. Xiong, Z. Cai, G. Tu, A survey of recent advances in edge-computing-powered artificial intelligence of things, IEEE Internet Things J. 8 (18) (2021) 13849–13875.
[20] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[21] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional, 2010.
[22] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[23] A. Telikani, A.H. Gandomi, A. Shahbahrami, A survey of evolutionary computation for association rule mining, Inform. Sci. 524 (2020) 318–352.
[24] I. Lera, C. Guerrero, C. Juiz, YAFS: A simulator for IoT scenarios in fog computing, IEEE Access 7 (2019) 91745–91758.
[25] L.K. Ng, S.S. Chow, A.P. Woo, D.P. Wong, Y. Zhao, Goten: GPU-outsourcing trusted execution of neural network training, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 17, 2021, pp. 14876–14883.
[26] A. Telikani, A. Shahbahrami, Optimizing association rule hiding using combination of border and heuristic approaches, Appl. Intell. 47 (2) (2017) 544–557.
[27] H. Surendra, H. Mohan, Hiding sensitive itemsets without side effects, Appl. Intell. 49 (4) (2019) 1213–1227.
[28] P. Cheng, J.F. Roddick, S.C. Chu, C.W. Lin, Privacy preservation through a greedy, distortion-based rule-hiding method, Appl. Intell. 44 (2) (2016) 295–306.
[29] G. Srivastava, J.C.W. Lin, Y. Djenouri, U. Yun, C.F. Cheng, G. Lin, Security protocol of sensitive high utility itemset hiding in shared IoT environments, Digit. Commun. Netw. (2021).
[30] S. Li, N. Mu, J. Le, X. Liao, Privacy preserving frequent itemset mining: Maximizing data utility based on database reconstruction, Comput. Secur. 84 (2019) 17–34.
[31] N. Domadiya, U.P. Rao, Privacy preserving distributed association rule mining approach on vertically partitioned healthcare data, Procedia Comput. Sci. 148 (2019) 303–312.
[32] C.W. Lin, T.P. Hong, K.T. Yang, S.L. Wang, The GA-based algorithms for optimizing hiding sensitive itemsets through transaction deletion, Appl. Intell. 42 (2) (2015) 210–230.
[33] J.M.T. Wu, G. Srivastava, A. Jolfaei, P. Fournier-Viger, J.C.W. Lin, Hiding sensitive information in ehealth datasets, Future Gener. Comput. Syst. 117 (2021) 169–180.
[34] J.C.W. Lin, Q. Liu, P. Fournier Viger, T.P. Hong, M. Voznak, J. Zhan, A sanitization approach for hiding sensitive itemsets based on particle swarm optimization, Eng. Appl. Artif. Intell. 53 (2016) 1–18.
[35] M.H. Afshari, M.N. Dehkordi, M. Akbari, Association rule hiding using cuckoo optimization algorithm, Expert Syst. Appl. 64 (2016) 340–351.
[36] S.M. Darwish, R.M. Essa, M.A. Osman, A.A. Ismail, Privacy preserving data mining framework for negative association rules: An application to healthcare informatics, IEEE Access 10 (2022) 76268–76280.
[37] G. Navale, S. Mali, Lossless and robust privacy preservation of association rules in data sanitization, Cluster Comput. 22 (1) (2019) 1415–1428.
[38] B. Suma, G. Shobha, Fractional salp swarm algorithm: An association rule based privacy-preserving strategy for data sanitization, J. Inf. Secur. Appl. 68 (2022) 103224.
[39] G. Shailaja, C.G. Rao, Opposition intensity-based cuckoo search algorithm for data privacy preservation, J. Intell. Syst. 29 (1) (2019) 1441–1452.
[40] G. Navale, S. Mali, A multi-analysis on privacy preservation of association rules using hybridized approach, Evol. Intell. 15 (2022) 1051–1065.
[41] A. Telikani, A. Shahbahrami, A.H. Gandomi, High-performance implementation of evolutionary privacy-preserving algorithm for big data using GPU platform, Inform. Sci. 579 (2021) 251–265.
[42] J.M.T. Wu, G. Srivastava, U. Yun, S. Tayeb, J.C.W. Lin, An evolutionary computation-based privacy-preserving data mining model under a multithreshold constraint, Trans. Emerg. Telecommun. Technol. 32 (3) (2021) e4209.
[43] J.C.W. Lin, Y. Zhang, B. Zhang, P. Fournier-Viger, Y. Djenouri, Hiding sensitive itemsets with multiple objective optimization, Soft Comput. 23 (23) (2019) 12779–12797.