Computers & Security 87 (2019) 101570

Contents lists available at ScienceDirect

Computers & Security

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

An efficient and scalable privacy preserving algorithm for big data and data streams

M.A.P. Chamikara[a,b,*], P. Bertok[a], D. Liu[b], S. Camtepe[b], I. Khalil[a]

[a] School of Science, RMIT University, Australia
[b] CSIRO Data61, Australia

ARTICLE INFO

Article history:
Received 27 November 2018
Revised 20 June 2019
Accepted 10 July 2019
Available online 22 July 2019

Keywords:
Privacy
Privacy-preserving data mining
Data streams
Smart cyber-physical systems
Internet of Things (IoT)
Web of Things (WoT)
Sensor data streams
Big data

ABSTRACT

A vast amount of valuable data is produced and is becoming available for analysis as a result of advancements in smart cyber-physical systems. The data comes from various sources, such as healthcare, smart homes, smart vehicles, and often includes private, potentially sensitive information that needs appropriate sanitization before being released for analysis. The incremental and fast nature of data generation in these systems necessitates scalable privacy-preserving mechanisms with high privacy and utility. However, privacy preservation often comes at the expense of data utility. We propose a new data perturbation algorithm, SEAL (Secure and Efficient data perturbation Algorithm utilizing Local differential privacy), based on Chebyshev interpolation and Laplacian noise, which provides a good balance between privacy and utility with high efficiency and scalability. Empirical comparisons with existing privacy-preserving algorithms show that SEAL excels in execution speed, scalability, accuracy, and attack resistance. SEAL provides flexibility in choosing the best possible privacy parameters, such as the amount of added noise, which can be tailored to the domain and dataset.

1. Introduction

Smart cyber-physical systems (SCPS) such as smart vehicles, smart grid, smart healthcare systems, and smart homes are becoming widely popular due to massive technological advancements in the past few years. These systems often interact with the environment to collect data mainly for analysis, e.g. to allow life activities to be more intelligent, efficient, and reliable (De Francisci Morales et al., 2016). Such data often includes sensitive details, but sharing confidential information with third parties can lead to a privacy breach. From our perspective, privacy can be considered as “Controlled Information Release” (Bertino et al., 2008). We can define a privacy breach as the release of private/confidential information to an untrusted environment. However, sharing the data with external parties may be necessary for data analysis, such as data mining and machine learning. Smart cyber-physical systems must have the ability to share information while limiting the disclosure of private information to third parties. Privacy-preserving data sharing and privacy-preserving data mining face significant challenges because of the size of the data and the speed at which data are produced. Robust, scalable, and efficient solutions are needed to preserve the privacy of big data and data streams generated by SCPS (Wen et al., 2018; Zhang et al., 2016a). Various solutions for privacy-preserving data mining (PPDM) have been proposed for data sanitization; they aim to ensure confidentiality and privacy of data during data mining (Backes et al., 2016; Vatsalan et al., 2017; Xue et al., 2011; Yang et al., 2017).

The two main approaches of PPDM are data perturbation (Chen and Liu, 2005b; 2011) and encryption (Kerschbaum and Härterich, 2017; Li et al., 2017). Although encryption provides a strong notion of security, due to its high computation complexity (Gai et al., 2016) it can be impractical for PPDM of SCPS-generated big data and data streams. Data perturbation, on the other hand, applies certain modifications such as randomization and noise addition to the original data to preserve privacy (Agrawal and Haritsa, 2005). These modification techniques are less complex than cryptographic mechanisms (Xu et al., 2014). Data perturbation mechanisms such as noise addition (Muralidhar et al., 1999) and randomization (Fox, 2015) provide efficient solutions towards PPDM. However, the utility of perturbed data cannot be 100% as data perturbation applies modifications to the original data, and the ability to infer knowledge from the perturbed data can result in a certain level of privacy leak as well. A privacy model (Machanavajjhala and Kifer, 2015) describes the limitations to the utility and privacy of a perturbation mechanism. Examples of such earlier

* Corresponding author at: School of Science, RMIT University, Building 14, Level 08, Room 17, Melbourne, VIC 3000, Australia.

E-mail address: pathumchamikara.mahawagaarachchige@rmit.edu.au (M.A.P. Chamikara).

https://doi.org/10.1016/j.cose.2019.101570
0167-4048/© 2019 Elsevier Ltd. All rights reserved.

privacy models include $k-anonymity$ (Niu et al., 2014; Zhang et al., 2016b) and $l-diversity$ (Machanavajjhala et al., 2006). However, it has been shown that older privacy models are defenseless against certain types of attacks, such as minimality attacks (Zhang et al., 2007), composition attacks (Ganta et al., 2008) and foreground knowledge (Wong et al., 2011) attacks. Differential privacy (DP) is a privacy model that provides a robust solution to these issues by rendering maximum privacy via minimizing the chance of private data leak (Dwork, 2009; Friedman and Schuster, 2010; Mohammed et al., 2011; Wang et al., 2015). Nevertheless, current DP mechanisms fail for small databases and have limitations on implementing efficient solutions for data streams and big data. When the database is small, the utility of DP mechanisms diminishes due to insufficient data being available for a reasonable estimation of statistics (Qin et al., 2016). At the other end of the scale, when the database is very large or continuously growing like in data streams produced by SCPS, the information leak of DP mechanisms is high due to the availability of too much information (Kellaris et al., 2014). Most perturbation mechanisms tend to leak information when the data is high-dimensional, which is a consequence of the dimensionality curse (Aggarwal, 2005). Moreover, the significant amount of randomization produced by certain DP algorithms results in low data utility. Existing perturbation mechanisms often ignore the connection between utility and privacy, even though improvement of one leads to deterioration of the other (Mivule and Turner, 2013). Furthermore, the inability to efficiently process high volumes of data and data streams makes the existing methods unsuitable for privacy-preservation in smart cyber-physical systems. New approaches that can appropriately answer the complexities in privacy preservation of SCPS generated data are needed.

The main contribution of this paper is a robust and efficient privacy-preserving algorithm for smart cyber-physical systems, which addresses the issues existing perturbation algorithms have. Our solution, SEAL (Secure and Efficient data perturbation Algorithm utilizing Local differential privacy), employs polynomial interpolation and notions of differential privacy. SEAL is a linear perturbation system based on Chebyshev polynomial interpolation, which allows it to work faster than comparable methods. We used generic datasets retrieved from the UCI data repository[1] to evaluate SEAL's efficiency, scalability, accuracy, and attack resistance. The results indicate that SEAL performs well at privacy-preserving data classification of big data and data streams. SEAL outperforms existing alternative algorithms in efficiency, accuracy, and data privacy, which makes it an excellent solution for smart system data privacy preservation.

The rest of the paper is organized as follows. Section 2 provides a summary of existing related work. The fundamentals of the proposed method are briefly discussed in Section 3. Section 4 describes the technical details of SEAL. Section 5 presents the experimental settings and provides a comparative analysis of the performance and security of PABIDOT. The results are discussed in Section 6, and the paper is concluded in Section 7. Detailed descriptions of the underlying concepts of SEAL are given in the Appendices.

2. Related work

Smart cyber-physical systems (SCPS) have become an important part of the IT landscape. Often these systems include IoT devices that allow effective and easy acquisition of data in areas such as healthcare, smart cities, smart vehicles, and smart homes (De Francisci Morales et al., 2016). Data mining and analysis are among the primary goals of collecting data from SCPS. The infrastructural extensions of SCPSs have contributed to the exponential growth in the number of IoT sensors, but security is often overlooked, and the devices become a source of privacy leak. The security and privacy concerns of big data and data streams are not entirely new, but require constant attention due to technological advancements of the environments and the devices used (Kieseberg and Weippl, 2018). Confidentiality, authentication, and authorization are just a few of the concerns (Balandina et al., 2015; Fernando et al., 2016; Sridhar et al., 2012). Many studies have raised the importance of privacy and security of SCPS due to their heavy use of personally identifiable information (PII) (Liu et al., 2012). Controlling access via authentication (Bertino, 2016), attribute-based encryption (Wang et al., 2014), temporal and location-based access control (Bertino, 2016) and employing constraint-based protocols (Kirkham et al., 2015) are some examples of improving privacy of SCPS.

In this paper, our primary target is maintaining privacy when sharing and mining data produced by SCPSs, and the focus is on “controlled information release”. Literature shows different attempts to impose constraints on data release and analysis in order to preserve privacy (Aldeen et al., 2015). Data encryption and data perturbation-based solutions have proven to be more viable for privacy-preserving data publishing and analysis than methods based on authentication and authorization (Verykios et al., 2004). Some recent examples for encryption based privacy-preserving approaches for cloud computing include PPM (Razaque and Rizvi, 2017), Sca-PBDA (Wu et al., 2016) and TDPP (Razaque and Rizvi, 2016), which provide scalable privacy-preserving data processing infrastructures. However, cryptographic mechanisms are less popular in PPDM for “controlled information release” due to the high computational complexity, hence not suitable for resource-constrained devices. Perturbing the instances of the original data by introducing noise or randomization is called input perturbation (Agrawal and Srikant, 2000; Aldeen et al., 2015), whereas perturbing the outputs of a query or analysis using noise addition or rule hiding is called output perturbation. Unidimensional perturbation and multidimensional perturbation (Agrawal and Srikant, 2000; Datta, 2004; Liu et al., 2006; Zhong et al., 2012) are the two main types of input perturbation classes. Examples for unidimensional perturbation include but are not limited to additive perturbation (Muralidhar et al., 1999), randomized response (Du and Zhan, 2003), and swapping (Estivill-Castro and Brankovic, 1999). Condensation (Aggarwal and Yu, 2004), random rotation (Chen and Liu, 2005b), geometric perturbation (Chen and Liu, 2011), random projection (Liu et al., 2006), and hybrid perturbation are types of multidimensional perturbation (Aldeen et al., 2015). Microaggregation(Domingo-Ferrer and Mateo-Sanz, 2002) can be considered as a hybrid perturbation mechanism that possesses both unidimensional and multidimensional perturbation capabilities.

As privacy models evolved, the limits of privacy imposed by particular mechanisms were evaluated (Machanavajjhala and Kifer, 2015), and new models were defined to overcome the issues of their predecessors. For example, $l-diversity$ (Machanavajjhala et al., 2006) was defined to overcome the shortcomings of $k-anonymity$ (Niu et al., 2014), then $(\alpha, k)-anonymity$ (Wong et al., 2006), $t-closeness$ (Li et al., 2007b) were proposed as further improvements. However, all these models eventually exhibited vulnerabilities to different attacks such as minimality (Zhang et al., 2007), composition (Ganta et al., 2008) and foreground knowledge (Wong et al., 2011) attacks. Moreover, these models were not scalable enough to address the curse of dimensionality presented by big data and data streams (Aggarwal, 2005; Cao et al., 2011), hence resulted in higher privacy leak (Aggarwal, 2005). In recent years differential privacy (DP) has drawn much attention as a powerful privacy model due to its inherent strong privacy guarantee. Differential privacy that is achieved via output perturbation is known as global differential privacy (GDP), whereas differential privacy

[1] https://archive.ics.uci.edu/ml/index.php.

achieved using input perturbation is known as local differential privacy (LDP). Laplacian mechanism and randomized response are two of the most frequently employed data perturbation methods used to achieve GDP and LDP (Dwork et al., 2014; Fox, 2015). LDP permits full or partial data release allowing the analysis of the perturbed data (Dwork, 2008; Kairouz et al., 2014), while GDP requires a trusted curator who enforces differential privacy by applying noise or randomization on results generated by running queries or by analysis of the original data (Dwork, 2008). Nevertheless, LDP algorithms are still at a rudimentary stage when it comes to full or partial data release of real-valued numerical data, and the complexity of selecting the domain of randomization with respect to a single data instance is still a challenge (Cormode et al., 2018; Erlingsson et al., 2014; Qin et al., 2016). Consequently, existing DP mechanisms are not suitable for differentially private data release.

Two important characteristics that determine the robustness of a particular perturbation mechanism are the ability to (1) protect against data reconstruction attacks and (2) perform well when high dimensional datasets and data streams are introduced. A data reconstruction attack tries to re-identify the individual owners of the data by reconstructing the original dataset from the perturbed dataset. Data reconstruction attacks are normally custom built for different perturbation methods using the properties of the perturbation mechanisms to restore the original data systematically. Different perturbation scenarios are vulnerable to different data reconstruction attacks. Principal component analysis (Wold et al., 1987), maximum likelihood estimation (Scholz, 2006), known I/O attack (Aggarwal and Philip, 2008), ICA attack (Chen and Liu, 2005a) and known sample attack (Aggarwal and Philip, 2008) are some examples of common data reconstruction attacks. For example, additive perturbation is vulnerable to principal component analysis (Huang et al., 2005) and maximum likelihood estimation (Huang et al., 2005), whereas multiplicative data perturbation methods are vulnerable to known input/output (I/O) attacks, known sample attacks and ICA attacks. These reconstruction attacks exploit the high information leak due to the dimensionality curse associated with high-dimensional data. In addition to providing extra information to the attacker, high-dimensional data also exponentially increases the amount of necessary computations (Aggarwal, 2005; Chen and Liu, 2005b; 2011; Machanavajjhala et al., 2006).

The literature shows methods that try to provide privacy-preserving solutions for data streams and big data by addressing the dimensionality curse in data streams. Recent attempts include a method proposing steered microaggregation to anonymize a data stream to achieve $k-anonymity$ (Domingo-Ferrer and Soria-Comas, 2017), but the problem of information leak inherent in $k-anonymity$ in case of high dimensional data can be a shortcoming of the method. Zhang et al. (2015) introduced a scalable data anonymization with MapReduce for big data privacy preservation. The proposed method first splits an original data set into partitions that contain similar data records in terms of quasi-identifiers and then locally recodes the data partitions by the proximity-aware agglomerative clustering algorithm in parallel, which limits producing sufficient utility for data streams such as produced by SCPS. Further, the requirement of advanced processing capabilities precludes its application to resource-constrained devices. Data condensation is another contender for data stream privacy preservation (Aggarwal and Yu, 2008). The problem in this case is that when the method parameters are set to achieve high accuracy (using small spatial locality), the privacy of the data often suffers. P^2RoCAl is a privacy preservation algorithm that provides high scalability for data streams and big data composed of millions of data records. P^2RoCAl is based on data condensation and random geometric transformations. Although P^2RoCAl provides linear complexity for the number of tuples, the computational complexity increases exponentially for the number of attributes (Chamikara et al., 2018). PABIDOT is a scalable privacy preservation algorithm for big data (Chamikara et al., 2019). PABIDOT comes with a new privacy model named $\Phi - separation$, which facilitates full data release with optimum privacy protection. PABIDOT can process millions of records efficiently as it has linear time complexity for the number of instances. However, PABIDOT also shows exponential time complexity for the number of attributes of a dataset (Chamikara et al., 2019). Other examples include the use of a Naive Bayesian Classifier for private data streams (Xu et al., 2008), and the method to efficiently and effectively track the correlation and autocorrelation structure of multivariate streams and leverage it to add noise to preserve privacy (Li et al., 2007a). The latter methods are also vulnerable to data reconstruction attacks such as principal component analysis-based attacks.

The complex dynamics exhibited by SCPS require efficient privacy preservation methods scalable enough to handle exponentially growing databases and data streams such as IoT stream data. Existing privacy-preserving mechanisms have difficulties in maintaining the balance between privacy and utility while providing sufficient efficiency and scalability. This paper tries to fill the gap, and proposes a privacy-preserving algorithm for SCPS which solves the existing issues.

3. Fundamentals

In this section, we provide some background and discuss the fundamentals used in the proposed method (SEAL). Our approach is generating a privacy-preserved version of the dataset in question, and allowing only the generated dataset to be used in any application. We use Chebyshev interpolation based on least square fitting to model a particular input data series, and the model formation is subjected to noise addition using the Laplacian mechanism used in differential privacy. The noise integrated model is then used to synthesize a perturbed data series which approximate the properties of the original input data series.

3.1. Chebyshev polynomials of the first kind

For the interpolation of the input dataset, we use Chebyshev polynomials of the first kind. These polynomials are a set of orthogonal polynomials as given by Definition 3 (available in Appendix A) (Mason and Handscomb, 2002). Chebyshev polynomials are a sequence of orthonormal polynomials that can be defined recursively. Polynomial approximation and numerical integration are two of the areas where Chebyshev polynomials are heavily used (Mason and Handscomb, 2002). More details on Chebyshev polynomials of the first kind can be found in Appendix A.

3.2. Least squares fitting

Least squares fitting (LSF) is a mathematical procedure that minimizes the sum of squares of the offsets of the points from the curve to find the best-fitting curve to a given set of points. We can use vertical least squares fitting which proceeds by finding the sum of squares of the vertical derivations R^2 (refer to Eq. B.1 in Appendix B) of a set of n data points (Weisstein, 2002). To generate a linear fit considering $f(x) = mx + b$, we can minimize the expression of squared error between the estimated values and the original values (refer to Eq. B.5), which proceeds to obtaining the linear system shown in Eq. (1) (using Eqs. (B.8) and B.9). We can solve Eq. (1) to find values of a and b to obtain the corresponding linear fit of $f(x) = mx + b$ for a given data series.

$$\begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} n & \left(\sum_{i=1}^{n} x_i\right) \\ \left(\sum_{i=1}^{n} x_i\right) & \left(\sum_{i=1}^{n} x_i^2\right) \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix} \tag{1}$$

3.3. Differential privacy

Differential Privacy (DP) is a privacy model that defines the bounds to how much information can be revealed to a third party or adversary about someone's data being present or absent in a particular database. Conventionally, ϵ (epsilon) and δ (delta) are used to denote the level of privacy rendered by a randomized privacy preserving algorithm (M) over a particular database (D). Let us take two adjacent datasets of D, x and y, where y differs from x only by one person. Then M satisfies (ϵ, δ)-differential privacy if Eq. (2) holds.

Privacy Budget and Privacy Loss (ϵ): ϵ is called the privacy budget that provides an insight into the privacy loss of a DP algorithm. When the corresponding ϵ value of a particular differentially private algorithm A is increased, the amount of noise or randomization applied by A on the input data is decreased. The higher the value of ϵ, the higher the privacy loss.

Probability to Fail a.k.a. Probability of Error (δ): δ is the parameter that accounts for "bad events" that might result in high privacy loss; δ is the probability of the output revealing the identity of a particular individual, which can happen $n \times \delta$ times where n is the number of records. To minimize the risk of privacy loss, $n \times \delta$ has to be maintained at a low value. For example, the probability of a bad event is 1% when $\delta = \frac{1}{100 \times n}$.

Definition 1. A randomized algorithm M with domain $N^{|X|}$ and range R is (ϵ, δ)-differentially private for $\delta \geq 0$, if for every adjacent x, $y \in N^{|X|}$ and for any subset $S \subseteq R$

$$Pr[(M(x) \in S)] \leq \exp(\epsilon) Pr[(M(y) \in S)] + \delta \tag{2}$$

3.4. Global vs. local differential privacy

Global differential privacy (GDP) and local differential privacy (LDP) are the two main approaches to differential privacy. In the GDP setting, there is a trusted curator who applies carefully calibrated random noise to the real values returned for a particular query. The GDP setting is also called the trusted curator model (Chan et al., 2012). Laplace mechanism and Gaussian mechanism (Dwork et al., 2014) are two of the most frequently used noise generation methods in GDP (Dwork et al., 2014). A randomized algorithm, M provides ϵ-global differential privacy if for any two adjacent datasets x, y and $S \subseteq R$, $Pr[(M(x) \in S)] \leq \exp(\epsilon) Pr[(M(y) \in S)] + \delta$ (i.e. Eq. (2) holds). On the other hand, LDP eliminates the need of a trusted curator by randomizing the data before the curator can access them. Hence, LDP is also called the untrusted curator model (Kairouz et al., 2014). LDP can also be used by a trusted party to randomize all records in a database at once. LDP algorithms may often produce too noisy data, as noise is applied commonly to achieve individual record privacy. LDP is considered to be a strong and rigorous notion of privacy that provides plausible deniability. Due to the above properties, LDP is deemed to be a state-of-the-art approach for privacy-preserving data collection and distribution. A randomized algorithm A provides ϵ-local differential privacy if Eq. (3) holds (Erlingsson et al., 2014).

Definition 2. A randomized algorithm A satisfies ϵ-local differential privacy, if for all pairs of inputs v_1 and v_2, for all $Q \subseteq Range(A)$ and for ($\epsilon \geq 0$) Eq. (3) holds. $Range(A)$ is the set of all possible outputs of the randomized algorithm A.

$$Pr[A(v_1) \in Q] \leq \exp(\epsilon) Pr[A(v_2) \in Q] \tag{3}$$

3.5. Sensitivity

Sensitivity is defined as the maximum influence that a single individual data item can have on the result of a numeric query. Consider a function f, the sensitivity (Δf) of f can be given as in Eq. (4) where x and y are two neighboring databases (or in LDP, adjacent records) and $\|.\|_1$ represents the $L1$ norm of a vector (Wang et al., 2016).

$$\Delta f = max\{\| f(x) - f(y) \|_1\} \tag{4}$$

3.6. Laplace mechanism

The Laplace mechanism is considered to be one of the most generic mechanisms to achieve differential privacy (Dwork et al., 2014). Laplace noise can be added to a function output ($F(D)$) as given in Eq. (6) to produce a differentially private output. Δf denotes the sensitivity of the function f. In the local differentially private setting, the scale of the Laplacian noise is equal to $\Delta f / \epsilon$, and the position is the current input value ($F(D)$).

$$PF(D) = F(D) + Lap\left(\frac{\Delta f}{\epsilon}\right) \tag{5}$$

$$PF(D) = \frac{\epsilon}{2 \Delta f} e^{-\frac{|x - F(D)|}{\Delta F}} \tag{6}$$

4. Our approach

The proposed method, SEAL, is designed to preserve the privacy of big data and data streams generated by systems such as smart cyber-physical systems. One of our aims was balancing privacy and utility, as they may adversely affect each other. For example, the spatial arrangement of a dataset can potentially contribute to its utility in data mining, as the results generated by the analysis mechanisms such as data classification and clustering are often influenced by the spatial arrangement of the input data. However, the spatial arrangement can be affected when privacy mechanisms apply methods like randomization. In other words, while data perturbation mechanisms improve privacy, at the same time they may reduce utility. Conversely, an increasing utility can detrimentally affect privacy. To address these difficulties, SEAL processes the data in three steps: (1) determine the sensitivity of the dataset to calibrate how much random noise is necessary to provide sufficient privacy, (2) conduct polynomial interpolation with calibrated noise to approximate a noisy function over the original data, and (3) use the approximated function to generate perturbed data. These steps guarantee that SEAL applies enough randomization to preserve privacy while preserving the spatial arrangement of the original data. SEAL uses polynomial interpolation accompanied by noise addition, which is calibrated according to the instructions of differential privacy. We use the first four orders of the Chebyshev polynomial of the first kind in the polynomial interpolation process. Then we calibrate random Laplacian noise to apply a stochastic error to the interpolation process, in order to generate the perturbed data. Fig. 1 shows the integration of SEAL in the general purpose data flow of SCPS. As shown in the figure, the data perturbed by the SEAL layer comes directly from the SCPS. That means that the data in the storage module has already gone through SEAL's privacy preservation process and does not contain any original data.

Fig. 2 shows the flow of SEAL where the proposed noisy Chebyshev model (represented by a green note) is used to approximate each of the individual attributes of a particular input dataset or data stream. The approximated noisy function is used to synthesize perturbed data, which is then subjected to random tuple shuffling to reduce the vulnerability to data linkage attacks.

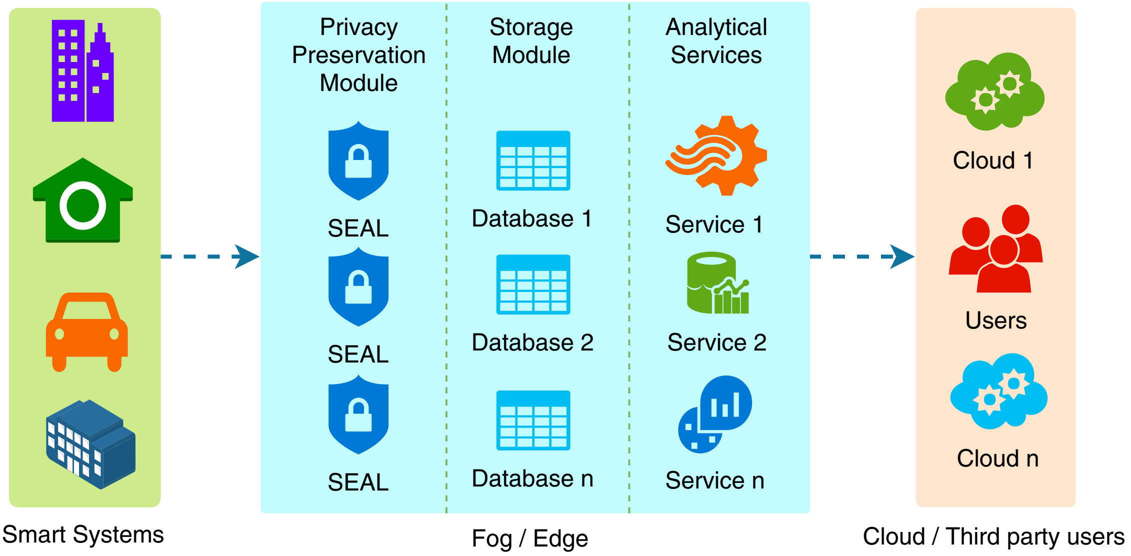


Fig. 1. Arrangement of SEAL in a smart system environment. In this setting, we assume that the original data are perturbed before reaching the storage devices. Any public or private services will have access only to the perturbed data.

4.1. Privacy-preserving polynomial interpolation for Noisy Chebyshev Model generation

We approximate an input data series (experimental data) by a functional expression with added randomization in order to inherit the properties of differential privacy. For approximation, our method uses the first four orders of Chebyshev polynomials of the first kind. We systematically add calibrated random Laplacian noise in the interpolation process, i.e. apply randomization to the approximation. Then we use the approximated function to re-generate the dataset in a privacy-preserving manner. We can denote an approximated function $\hat{f}$ of degree $(m-1)$ using Eq. (7), where the degree of (φ_k) is $k-1$. For the approximation, we consider the root mean square error (RMSE) E between the estimated values and the original values (refer to Eq. (C.14)). We use the first four Chebyshev polynomials of the first kind for the approximation, which limits the number of coefficients to four (we name the coefficients as a_1, a_2, a_3, and a_4). Now we can minimize E (the RMSE) to obtain an estimated function $\hat{f}^*(x)$, thus seeking to minimize the squared error $M(a_1, a_2, a_3, a_4)$. For more details refer to Eq. (C.15), where $a_1, a_2, \ldots, a_m$ are coefficients and $\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)$ are Chebyshev polynomials of first kind.

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \cdots + a_m\varphi_m(x) \tag{7}$$

4.1.1. Introducing privacy to the approximation process utilizing differential privacy (the determination of the sensitivity and the position of Laplacian noise)

We apply the notion of differential privacy to the private data generation process by introducing randomized Laplacian noise to the root mean square error (RMSE) minimization process. Random Laplacian noise introduces a calibrated randomized error in deriving the values for a_1, a_2, a_3, and a_4 with an error (refer to Eqs. (C.22), (C.25), (C.28) and (C.31)). We add Laplacian noise with a sensitivity of 1, as the input dataset is normalized within the bounds of 0 and 1, which restricts the minimum output to 0 and maximum output to 1 (refer to Eq. (C.20)). We select the position of Laplacian noise to be 0, as the goal is to keep the local minima of RMSE around 0. We can factorize the noise introduced squared error minimization equations to form a linear system which can be denoted by Eq. (8). C is the coefficient matrix obtained from the factorized expressions, A is the coefficient vector obtained from M, and B is the constant vector obtained from the factorized expressions (refer Eqs. (C.24), (C.27), (C.30), and (C.33) where m_{ij} denote the coefficients and b_i denote the constants).

$$CA = B \tag{8}$$

Where,

$$C = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \tag{9}$$

$$A = [a_1, a_2, a_3, a_4]^T \tag{10}$$

$$B = [b_1, b_2, b_3, b_4]^T \tag{11}$$

Now we solve the corresponding linear system (formed using Eqs. (C.35)-(C.37)), to obtain noisy values for a_1, a_2, a_3, and a_4 in order to approximate the input data series with a noisy function. The results will differ each time we calculate the values for a_1, a_2, a_3, and a_4 as we have randomized the process of interpolation by adding randomized Laplacian noise calibrated using a user-defined ϵ value.

4.2. Algorithmic steps of SEAL for static data and data streams

Algorithm 1 presents the systematic flow of steps in randomizing the data to produce a privacy-preserving output. The algorithm accepts input dataset (D), privacy budget ϵ (defined in Eq. (C.21)), window size (ws) and threshold (t) as the input parameters. The window size defines the number of data instances to be perturbed in one cycle of randomization. The window size of a data stream is essential to maintain the speed of the post-processing analysis/modification (e.g. data perturbation, classification, and clustering) done to the data stream (Hammad et al., 2003). For static data sets, the threshold is maintained with a default value of -1. For a static dataset, $t = -1$ ignores that a specific number of perturbed windows need to be released before the whole dataset is completed. In the case of data streams, the window size (ws) and the threshold t are useful as ws can be maintained as a data buffer and t can be specified with a certain number to let the algorithm know that it has to release every t number of processed windows.

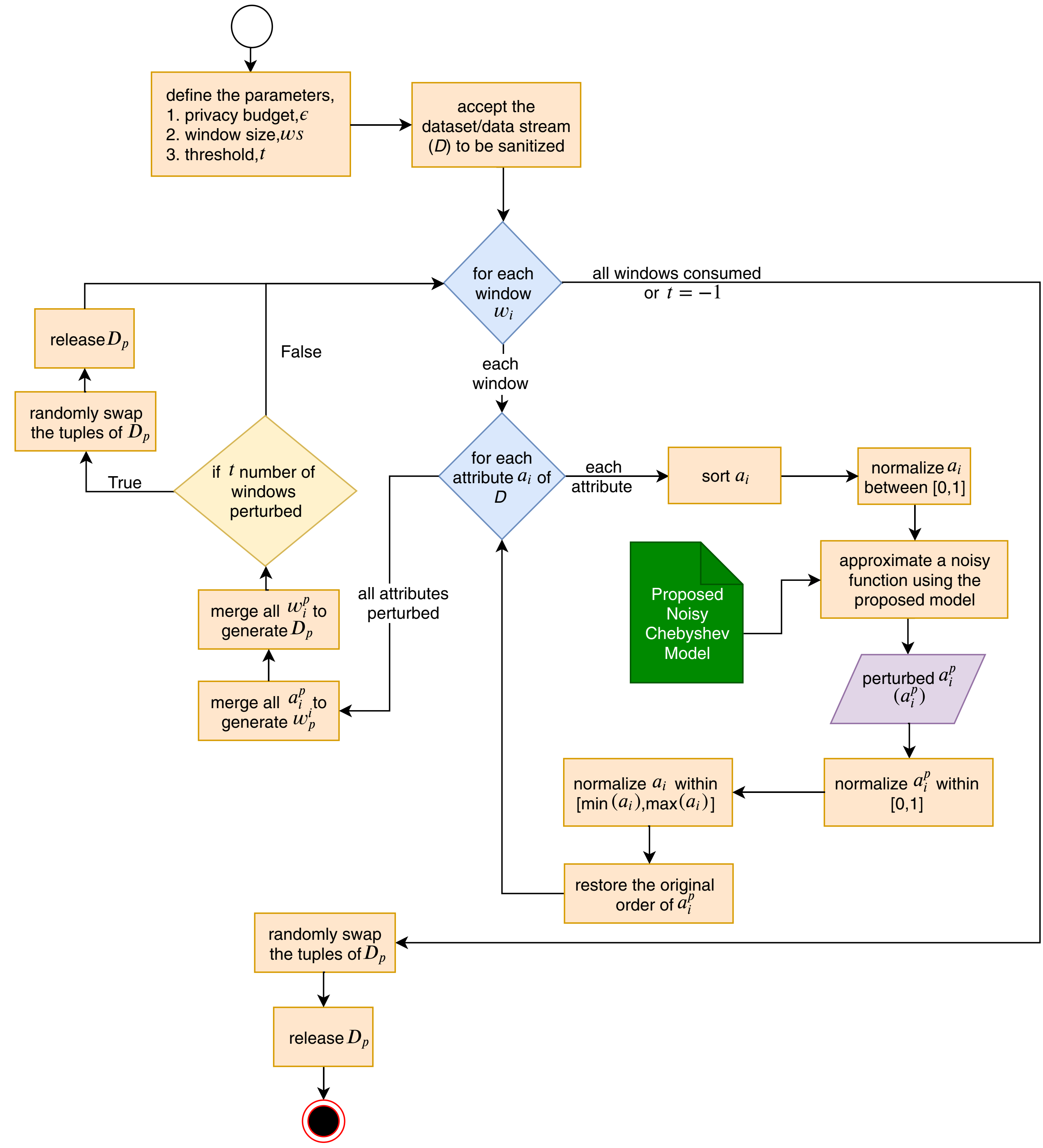

define the parameters,
1. privacy budget,ϵ
2. window size,ws
3. threshold,t
accept the dataset/data stream (D) to be sanitized
for each window w_i
all windows consumed or $t=-1$
release D_p
False
each window
randomly swap the tuples of D_p
True
if t number of windows perturbed
for each attribute a_i of D
each attribute
sort a_i
normalize a_i between [0,1]
Proposed Noisy Chebyshev Model
approximate a noisy function using the proposed model
merge all w_i^p to generate D_p
all attributes perturbed
perturbed a_i^p (a_i^p)
merge all a_i^p to generate w_p^i
normalize a_i within [min(a_i),max(a_i)]
normalize a_i^p within [0,1]
restore the original order of a_i^p
randomly swap the tuples of D_p
release D_p

Fig. 2. The basic flow of SEAL. The users can calibrate the level of privacy using the privacy budget parameter (ϵ). The smaller the ϵ, the higher the privacy. It is recommended to use an ϵ in the interval (0,10), which is considered to be the default range to provide a sufficient level of privacy.

Maintaining t is important for data streams because data streams are growing infinitely in most cases, and the algorithm makes sure that the data is released in predefined intervals.

According to conventional differential privacy, the acceptable values of ϵ should be within a small range, ideally in the interval of (0,9] (Abadi et al., 2016). Due to the lower sensitivity of the interpolation process, increasing ϵ greater than 2 may lower privacy. It is the users' responsibility to decrease or increase ϵ depending on the requirements. We suggest an ϵ of 1 to have a balance between privacy and utility. If the user chooses an ϵ value less than 1, the algorithm will provide higher randomization, hence providing higher privacy and lower utility, whereas lower privacy and a higher utility will be provided in case of an ϵ value higher than 1. The selection of *ws* depends specifically on the size of the particular dataset. A comparably larger *ws* can be chosen for a large dataset, while *ws* can be smaller for a small dataset. For a static dataset, *ws* can range from a smaller value such as one-tenth the *size of the dataset* to the *full size of the dataset*. The min-

Algorithm 1 Steps of the perturbation algorithm: SEAL

```
Inputs :
    $D$         ← input dataset (numeric)
    $\epsilon$  ← scale of Laplacian noise
    $ws$        ← data buffer/window size
    $t$         ← threshold for the maximum number
                  of windows processed
                  before a data release (default value of $t = -1$)
Outputs :  $D^p$ ← perturbed dataset
divide $D$ in to data partitions ($w_i$) of size $ws$
$x = [1, \dots, ws]$
normalize $x$ within the bounds of $[0, 1]$
for each $w_i$ do
    rep=rep+1
    $D^p = []$                                   ▷ empty matrix
    normalize data of $w_i$ within the bounds of $[0, 1]$
    for each attribute $a_i$ in $w_i$ do
        $sa_i = sort(a_i)$                       ▷ sorted in ascending order
        generate $M$                             ▷ according to Eq. C.35
        generate $B$                             ▷ according to Eq. C.37
        $A = B * M^{-1}$
        use $A = [a_1, a_2, a_3, a_4]$ to generate perturbed data ($a_i^p$)
        using $f(\hat{x})$                       ▷ refer Eq. C.12
        normalize $a_i^p$ within the bounds of $[0, 1]$ to generate $a_i^N$
        normalize $a_i^N$ within the bounds of $[min(a_i), max(a_i)]$
        resort $a_i^N$ to the original order of $a_i$ to generate $a_i^o$
    end for
    merge all $a_i^o$ to generate $w_i^p$
    $D^p = merge(D^p, w_i^p)$
    if rep==t then
        randomly swap the tuple of $D^p$
        release $D^p$
        rep=0
    end if
end for
if t==-1 then
    randomly swap the tuple of $D^p$
    return $D^p$
end if
End Algorithm
```

imum value of *ws* should not go down to a small value (e.g. < 100) because it increases the number of perturbation cycles and introduces an extreme level of randomization to the input dataset, resulting in poor utility. For a data stream, *ws* is considered as the buffer size and can range from a smaller value to any number of tuples that fit in the memory of the computer. Further discussions on selecting suitable values for ϵ and *ws* is provided in Section 5.2.1.

4.3. A use case: SEAL integration in a healthcare smart cyber-physical system

Biomedical and healthcare systems provide numerous opportunities and challenges for SCPS. Intelligent operating rooms and hospitals, image-guided surgery and therapy, fluid flow control for medicine and biological assays and the development of physical and neural prostheses are some of the examples for biomedical and healthcare systems which can be effectively facilitated and improved using SCPS (Baheti and Gill, 2011). However, biomedicine and healthcare data can contain a large amount of sensitive, personal information. SEAL provides a practical solution and can impose privacy in such scenarios to limit potential privacy leak from such systems (Baheti and Gill, 2011).

Fig. 3 shows a use case for SEAL integration in a healthcare smart cyber-physical system. Patients can have several sensors attached to them for recording different physical parameters. The recorded data are then transmitted to a central unit which can be any readily available digital device such as a smartphone, a personal computer, or an embedded computer. A large variety of sensors are available today, e.g. glucose monitors, blood pressure monitors (Alhayajneh et al., 2018). In the proposed setting, we assume that the processing unit that runs SEAL, perturbs all sensitive inputs forwarded to the central unit. As shown in the figure, we assume that the central units do not receive any unperturbed sensitive information, and the data repositories will store only perturbed data, locally or in a remote data center. Data analysts can access and use only the perturbed data to conduct their analyses. Since the data is perturbed, adversarial attacks on privacy will not be successful.

5. Experimental results

In this section, we discuss the experimental setup, resources used, experiments, and their results. The experiments were conducted using seven datasets retrieved from the UCI data repository.[2] We compare the results of SEAL against the results of rotation perturbation (RP), geometric perturbation (GP) and data condensation (DC). For performance comparison with SEAL, we selected GP and RP when using static datasets, while DC was used with data streams. The main reason for selecting GP, RP, and DC is that they are multidimensional perturbation mechanisms that correlate with the technique used in the linear system of SEAL as given in Eq. (C.34). Fig. 4 shows the analytical setup which was used to test the performance of SEAL. We perturbed the input data using SEAL, RP, GP, and DC and conducted data classification experiments on the perturbed data using five different classification algorithms to test and compare utility. We used the default settings of 10 iterations with a noise factor (sigma) of 0.3 to perturb the data using RP and GP. Next, SEAL was tested for attack resistance against naive inference, known I/O attacks, and ICA-based attacks that are based on data reconstruction. These attacks are more successful against perturbation methods that use matrix multiplications. The attack resistance results of SEAL were then compared with the attack resistance results of RP, GP, and DC. Subsequently, we tested and compared SEAL's computational complexity and scalability by using two large datasets. Finally, we tested the performance of SEAL on data streams and compared the results with the results of DC.

5.1. Experimental setup

For the experiments we used a Windows 10 (Home 64-bit, Build 17134) computer with Intel (R) i5-6200U (6th generation) CPU (2 cores with 4 logical threads, 2.3 GHz with turbo up to 2.8 GHz) and 8192 MB RAM. The scalability of the proposed algorithm was tested using a Linux (SUSE Enterprise Server 11 SP4) SGI UV3000 supercomputer, with 64 Intel Haswell 10-core processors, 25 MB cache and 8TB of global shared memory connected by SGI's NUMAlink interconnect. We implemented SEAL in MATLAB R2016b. The experiments on data classification were carried out on Weka 3.6 (Witten et al., 2016), which is a collection of machine learning algorithms for data mining tasks.

5.1.1. Datasets used in the experiments

A brief description of the datasets is given in Table 1. The dimensions of the datasets used for the performance testing vary

[2] https://archive.ics.uci.edu/ml/index.php.

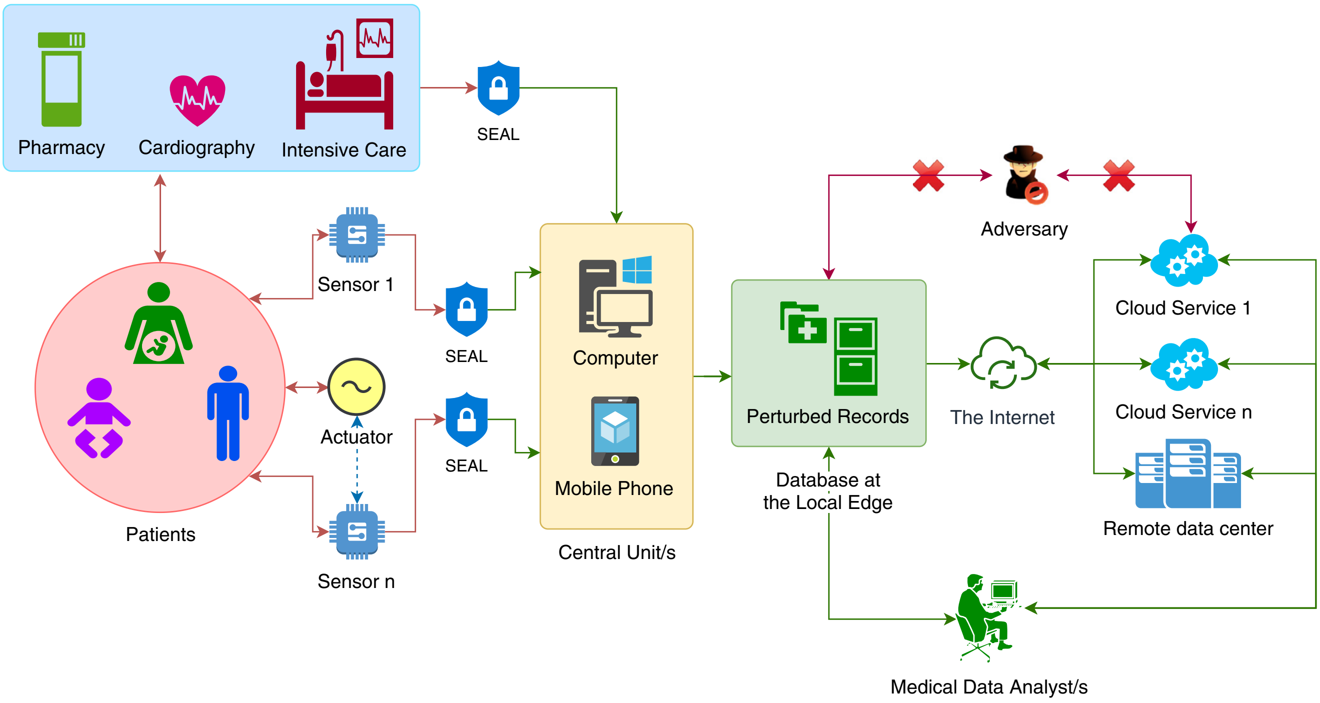


Fig. 3. A use case: The integration of SEAL in a healthcare smart cyber-physical system. As shown in the figure, SEAL perturbs data as soon as they leave the source (medical sensors, medical devices, etc.). In the proposed setting, SEAL assumes that there is no trusted party.

Table 1
Short descriptions of the datasets selected for testing.

Dataset	Abbreviation	Number of records	Number of attributes	Number of classes
Wholesale customers[a]	WCDS	440	8	2
Wine Quality[b]	WQDS	4898	12	7
Page Blocks Classification[c]	PBDS	5473	11	5
Letter Recognition[d]	LRDS	20,000	17	26
Statlog (Shuttle)[e]	SSDS	58,000	9	7
HEPMASS[f]	HPDS	3,310,816	28	2
HIGGS[g]	HIDS	11,000,000	28	2

[a] https://archive.ics.uci.edu/ml/datasets/Wholesale+customers
[b] https://archive.ics.uci.edu/ml/datasets/Wine+Quality
[c] https://archive.ics.uci.edu/ml/datasets/Page+Blocks+Classification
[d] https://archive.ics.uci.edu/ml/datasets/Letter+Recognition
[e] https://archive.ics.uci.edu/ml/datasets/Statlog+ %28Shuttle%29
[f] https://archive.ics.uci.edu/ml/datasets/HEPMASS#
[g] https://archive.ics.uci.edu/ml/datasets/HIGGS#

from small to extremely large, to check the performance of SEAL in different circumstances. Since the current version of SEAL conducts perturbation only on numerical data; we selected only numerical datasets in which the only attribute containing non-numerical data is the class attribute.

5.1.2. Perturbation methods used for comparison

Random rotation perturbation (RP), geometric data perturbation (GP), and data condensation (DC) are three types of matrix multiplicative perturbation approaches which are considered to provide high utility in classification and clustering (Okkalioglu et al., 2015). In RP, the original data matrix is multiplied using a random rotation matrix which has the properties of an orthogonal matrix. A rotational matrix R follows the property of $R \times R^T = R^T \times R = I$, where I is the identity matrix. The application of rotation is repeated until the algorithm converges at the desired level of privacy (Chen and Liu, 2005b). In GP, a random translation matrix is added to the process of perturbation in order to enhance privacy. The method accompanies three components: rotation perturbation, translation perturbation, and distance perturbation (Chen and Liu, 2011). Due to the isometric nature of transformations, the perturbation process preserves the distance between the tuples, resulting in high utility for classification and clustering. RP and GP can only be used for static datasets in their current setting, due to their recursive approach to deriving the optimal perturbation. DC is specifically introduced for data streams. In DC, data are divided into multiple homogeneous groups of predefined size (accepted as user input) in such a way that the difference between the records in a particular group is minimal, and a certain level of statistical information about different records is maintained. The sanitized data is generated using a uniform random distribution based on the eigenvectors which are generated using the eigendecomposition of the characteristic covariance matrices of each homogeneous group (Aggarwal and Yu, 2004).

5.1.3. Classification algorithms used in the experiments

Different classes of classification algorithms employ different classification strategies (Lessmann et al., 2015). To investigate the performance of SEAL with diverse classification methods, we chose five different algorithms as the representative of different classes,

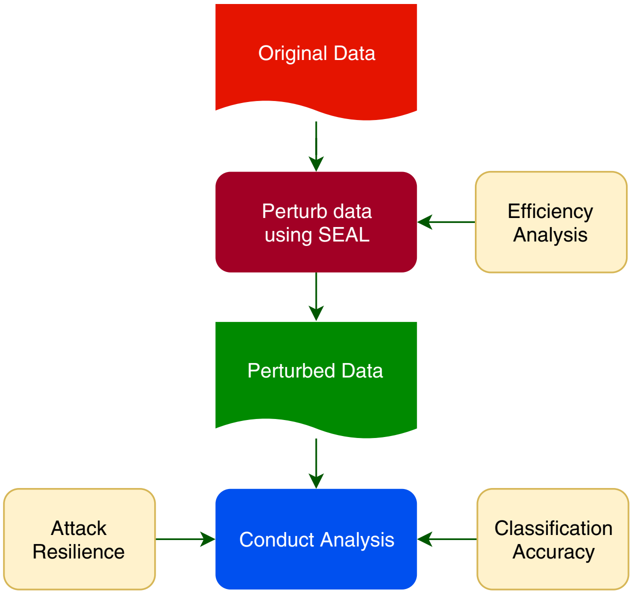


Fig. 4. The analytical setup used for SEAL. The figure shows the different levels of performance analysis of SEAL. First, the time consumption of perturbation was recorded under the efficiency analysis. Next, the attack resistance and classification accuracy were analyzed upon the perturbed data.

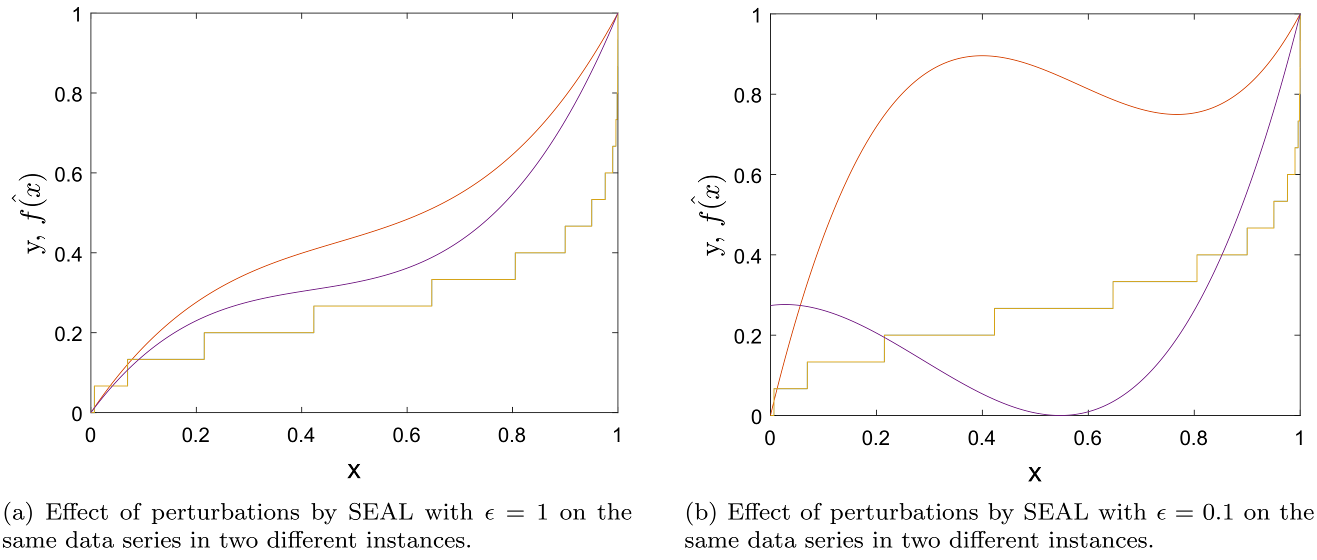
namely: Multilayer Perceptron (MLP) (Witten et al., 2016), k-Nearest Neighbor (kNN) (Witten et al., 2016), Sequential Minimal Optimization (SMO) (Schölkopf et al., 1999), Naive Bayes (Witten et al., 2016), and J48 (Quinlan, 1993), and tested SEAL for its utility in terms of classification accuracy. MLP uses back-propagation to classify instances (Witten et al., 2016), kNN is a non-parametric method used for classification (Witten et al., 2016), and SMO is an implementation of John Platt's sequential minimal optimization algorithm for training a support vector classifier (Schölkopf et al., 1999). Naive Bayes is a fast classification algorithm based on probabilistic classifiers (Witten et al., 2016), while J48 is an implementation of the decision tree based classification algorithm (Witten et al., 2016).

5.2. Performance evaluation of SEAL

We evaluated the performance of SEAL with regard to classification accuracy, attack resistance, time complexity, scalability, and also looked at data streams. First, we generated perturbed data using SEAL, RP, GP, and DC for the datasets: WCDS, WQDS, PBDS, LRDS, and SSDS (refer to Table 1) under the corresponding settings. The perturbed data were then used to determine classification accuracy and attack resistance for each perturbed dataset. During the classification accuracy experiments, k of k-nearest neighbor (kNN) classification algorithm was kept at 1. The aggregated results were rated using the nonparametric statistical comparison test, Friedman's rank test, which is analogous to a standard one-way repeated-measures analysis of variance (Howell, 2016). We recorded the statistical significance values, and the Friedman's mean ranks (FMR) returned by the rank test. The time consumption of SEAL was evaluated using runtime complexity analysis. We ran SEAL on two large-scale datasets, HPDS and HIDS, to test its scalability. Finally, the performance of SEAL was tested on data streams by running it on the LRDS dataset, and the results were compared with those produced by DC.

5.2.1. Effect of randomization on the degree of privacy

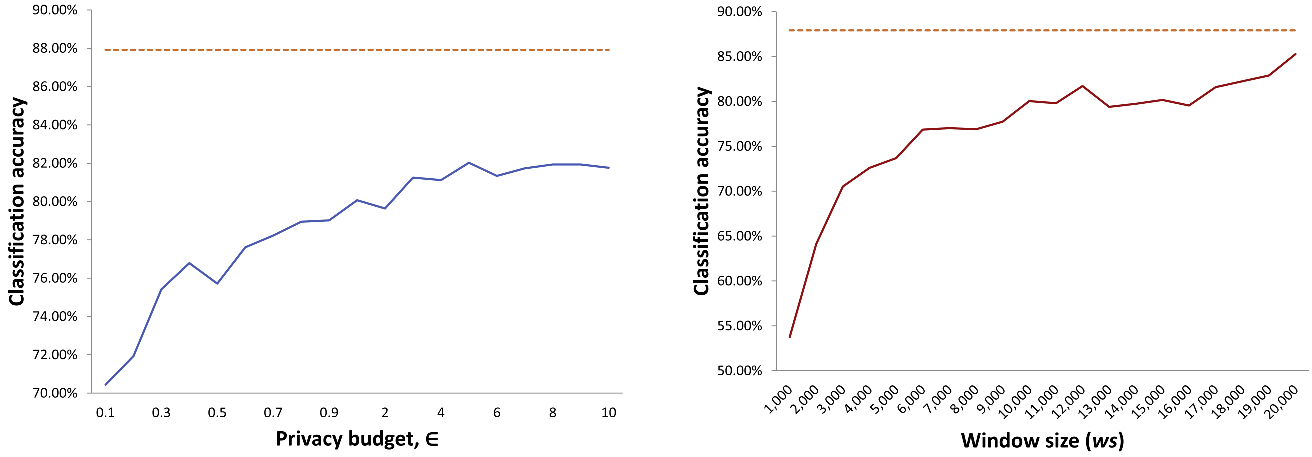
One of the main features of SEAL is its ability to perturb a dataset while preserving the original shape of data distribution. We ran SEAL on the same data series to detect the effect of randomization in two different instances of perturbation. This experiment is to check and guarantee that SEAL does not publish similar perturbed data when it is applied with the same ϵ value to the same data on different occasions. This feature enables SEAL to prevent privacy leak via data linkage attacks that are exploiting multiple data releases. As depicted in Fig. 5, in two separate applications, SEAL generates two distinct randomized data series, while preserving the shape of the original data series. The left-hand plot of Fig. 5 shows the data generated under an ϵ of 1, whereas the right-hand plot shows the data generated under an ϵ of 0.1. The right plot clearly shows the effect of high randomization under the extreme level of privacy generated by a strict privacy budget (ϵ) of 0.1.



(a) Effect of perturbations by SEAL with $\epsilon = 1$ on the same data series in two different instances.

(b) Effect of perturbations by SEAL with $\epsilon = 0.1$ on the same data series in two different instances.

Fig. 5. Effect of perturbation by SEAL. The plot with the staircase pattern represents the original data series (first attribute of the LRDS dataset). The two plots that were plotted above the original data series represent two instances of perturbation conducted by SEAL on the original data series.



(a) Change of classification accuracy of the LRDS dataset perturbed by SEAL, where the window size (ws) was maintained at $10,000$ tuples.

(b) Change of classification accuracy (J48) of the LRDS dataset perturbed by SEAL where the privacy budget (ϵ) was maintained at 1.

Fig. 6. Classification accuracy of SEAL. The classification accuracy was obtained by classifying the corresponding datasets using the J48 classification algorithm. The orange dotted horizontal lines on the two plots represent the classification accuracy of the original dataset. The window size (*ws*) is measured in number of tuples.

5.2.2. *Dynamics of privacy budget (ϵ) and window size (*ws*)*

As explained in Section 5.2.1, smaller ϵ means higher randomization, which results in decreased utility. Fig. 6a shows the change of classification accuracy against an increasing ϵ. As shown in the figure, classification accuracy increases with an increasing privacy budget (ϵ). Fig. 6a shows a more predictable pattern of increasing utility (classification accuracy) against increasing ϵ. The choice of a proper ϵ depends on the application requirements: a case that needs higher privacy should have a smaller ϵ, while a larger ϵ will provide better utility. As it turns out, two-digit ϵ values provide no useful privacy. Given that SEAL tries to preserve the shape of the original data distribution, we recommend a range of 0.4 to 3 for ϵ to limit unanticipated privacy leaks. We showed that SEAL provides better privacy and utility than comparable methods under a privacy budget of 1.

Next, we tested the effect of window size (*ws*) on classification accuracy and the magnitude of randomization performed by SEAL. As shown on Fig. 6b, classification accuracy increases when *ws* increases. When *ws* is small, the dataset is divided into more groups than when *ws* is large. When there is more than one group to be perturbed, SEAL applies randomization on each group distinctly. Since each of the groups is subjected to distinct randomization, the higher the number of groups, the larger the perturbation of the dataset. For smaller sizes of *ws*, SEAL will produce higher perturbation, resulting in more noise, reduced accuracy, improved privacy, and better resistance to data reconstruction attacks.

5.2.3. *Classification accuracy*

Table 2 provides the classification accuracies when using the original dataset and the datasets perturbed by the three methods. During the experiments for classification accuracy, we maintained ϵ at 1 and *ws* at the total length of the dataset. For example, if the dataset contained n number of tuples, *ws* was maintained at n. After producing the classification accuracies, Friedman's rank test was conducted on the data available in Table 2 to rank the three methods: GP, RP, and SEAL. The mean ranks produced by Friedman's rank (FMR) test[3] are presented in the last row of Table 2. The p-value suggests that the difference between the classification accuracies of RP, GP, and SEAL are significantly different. When evaluating FMR values on classification accuracies, a higher rank means that the corresponding method tends to produce better classification results. The mean ranks indicate that SEAL provides comparatively higher classification accuracy. SEAL is capable of providing higher utility in terms of classification accuracy due to its ability to maintain the shape of the original data distribution despite the introduced randomization. Although SEAL provides better performance overall than the other two methods, we can notice that in a few cases (as shown in Table 2) SEAL has produced slightly lower classification accuracies. We assume that this is due to the effect of variable random noise applied by SEAL. However, these lower accuracies are still on par with accuracies produced by the other two methods.

5.2.4. *Attack resistance*

Table 3 shows the three methods' (RP, GP, and SEAL) resistance to three attack methods: naive snooping (NI), independent component analysis (ICA) and known I/O attack (IO) (Chen and Liu, 2005b; Okkalioglu et al., 2015). We used the same parameter settings of SEAL ($\epsilon = 1$ and *ws*=number of tuples) which were used in classification accuracy experiments for attack resistance analysis as well. IO and ICA data reconstruction attacks try to restore the original data from the perturbed data and are more successful in attacking matrix multiplicative data perturbation. FastICA package (Gävert et al., 2005) was used to evaluate the effectiveness of ICA-based reconstruction of the perturbed data. We obtained the attack resistance values as standard deviation values of (i) the difference between the normalized original data and the perturbed data for NI, and (ii) the difference between the normalized original data and reconstructed data for ICA and IO. During the IO attack analysis, we assume that around 10% of the original data is known to the adversary. The "*min*" values under each test indicate the minimum guarantee of resistance while "*avg*" values give an impression of the overall resistance.

We evaluated the data available in Table 3 using Friedman's rank test to generate the mean ranks for GP, RP, and SEAL. The mean ranks produced by Friedman's rank test[4] are given in the last row of Table 3. The p-value implies that the difference between the attack resistance values is significantly different. As for

[3] The FMR test returned a χ^2 value of 27.6566, a degree of freedom of 2 and a p-value of 9.8731e-07.

[4] The test statistics: χ^2 value of 14.6387, a degree of freedom of 2 and a p-value of 6.6261e-04.

Table 2
Classification accuracies obtained when using the original dataset and the datasets perturned by three methods. During the experiments conducted using SEAL, the privacy budget ϵ was maintained at 1. The window size *ws* was maintained at full length of the corresponding dataset. For example, *ws* of SEAL for the LRDS dataset was maintained at 20,000 during the experiments presented in this table. The last row shows the mean ranks returned by the nonparametric statistical comparison test (Friedman's rank test on the classification accuracies) of the three methods. A larger FMR value represents better classification accuracy.

Dataset	Algorithm	MLP	IBK	SVM	Naive Bayes	J48
LRDS	Original	82.20%	95.96%	82.44%	64.01%	87.92%
	RP	74.04%	87.19%	71.07%	48.41%	64.89%
	GP	79.12%	93.05%	77.92%	59.89%	70.54%
	SEAL	80.59%	93.67%	81.71%	63.10%	85.28%
PBDS	Original	96.25%	96.02%	92.93%	90.85%	96.88%
	RP	92.00%	95.52%	89.99%	35.76%	95.61%
	GP	90.24%	95.67%	89.93%	43.10%	95.49%
	SEAL	96.34%	96.73%	95.59%	86.97%	96.34%
SSDS	Original	99.72%	99.94%	96.83%	91.84%	99.96%
	RP	96.26%	99.80%	88.21%	69.04%	99.51%
	GP	98.73%	99.81%	78.41%	79.18%	99.59%
	SEAL	99.70%	99.21%	98.51%	89.94%	99.87%
WCDS	Original	90.91%	87.95%	87.73%	89.09%	90.23%
	RP	89.09%	85.00%	82.27%	84.55%	86.82%
	GP	91.82%	86.59%	85.00%	84.32%	88.86%
	SEAL	89.32%	86.82%	89.09%	88.41%	86.59%
WQDS	Original	54.94%	64.54%	52.14%	44.67%	59.82%
	RP	47.65%	53.29%	44.88%	32.32%	45.53%
	GP	48.86%	56.88%	44.88%	32.16%	46.43%
	SEAL	53.92%	64.02%	52.02%	47.83%	84.15%
FMR Values		RP: 1.34		GP: 1.86	SEAL: 2.80	

Table 3
Attack resistance of the algorithms. During the experiments conducted using SEAL, the privacy budget ϵ was maintained at 1. The window size *ws* was maintained at full length of the corresponding dataset. For example, *ws* of SEAL for the LRDS dataset was maintained at 20,000 during the experiments presented in this table. The last row provides Friedman's mean ranks returned by the nonparametric statistical comparison test on the three methods.

Dataset	Algorithm	NI_{min}	NI_{avg}	ICA_{min}	ICA_{avg}	IO_{min}	IO_{avg}
LRDS	RP	0.8750	1.4490	0.4057	0.6942	0.0945	0.2932
	GP	1.3248	1.6175	0.6402	0.7122	0.0584	0.4314
	SEAL	1.4061	1.4148	0.7024	0.7062	0.6986	0.7056
PBDS	RP	0.7261	1.3368	0.5560	0.6769	0.0001	0.1242
	GP	0.2845	1.4885	0.1525	0.6834	0.0000	0.1048
	SEAL	1.3900	1.4084	0.7008	0.7056	0.6932	0.7031
SSDS	RP	1.2820	1.5015	0.1751	0.5909	0.0021	0.0242
	GP	1.4490	1.6285	0.0062	0.3240	0.0011	0.0111
	SEAL	1.4065	1.4119	0.7038	0.7068	0.7027	0.7068
WCDS	RP	1.0105	1.3098	0.6315	0.7362	0.0000	0.0895
	GP	1.4620	1.7489	0.1069	0.6052	0.0000	0.1003
	SEAL	1.3130	1.3733	0.6775	0.7053	0.6557	0.6930
WQDS	RP	1.2014	1.4957	0.4880	0.7062	0.0057	0.4809
	GP	1.3463	1.6097	0.3630	0.6536	0.0039	0.4025
	SEAL	1.3834	1.4138	0.7018	0.7053	0.6859	0.7026
FMR Values		RP: 1.68		GP: 1.75		SEAL: 2.57	

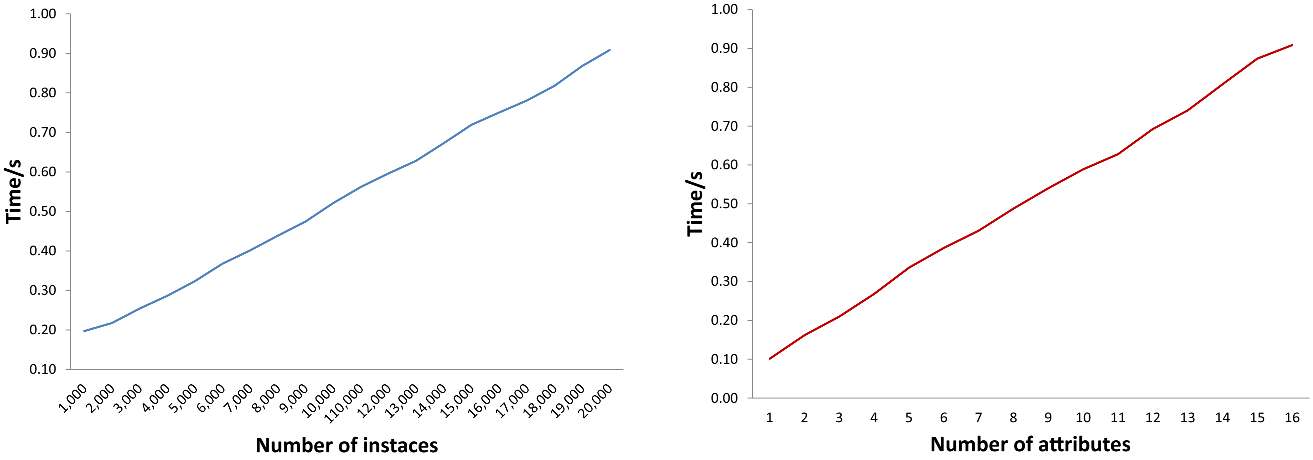
the FMR values on attack resistance, a higher rank means that the corresponding method tends to be more attack-resistant. The mean ranks suggest that SEAL provides comparatively higher security than the comparable methods against the privacy attacks.

5.2.5. Time complexity

Algorithm 1 (SEAL) has two loops. One loop is controlled by the number of data partitions resulting from the window size (*ws*), and the number of attributes controls the other loop. In a particular instance of perturbation, these two parameters (*ws* and the number of attributes) remain constants. Let us take the contribution of both loops to the computational complexity as a constant (k). If we evaluate the steps of SEAL from step 9 to step 23, we can see that the highest computational complexity in these steps is $O(n)$, where n is the number of tuples. From this, we can estimate the time complexity of Algorithm 1 to be $O(kn) = O(n)$. We investigated the time consumption against the number of instances and the number of attributes to determine the computational complexity empirically. Fig. 7 confirms that the time complexity of SEAL is in fact $O(n)$.

5.2.6. Time complexity comparison

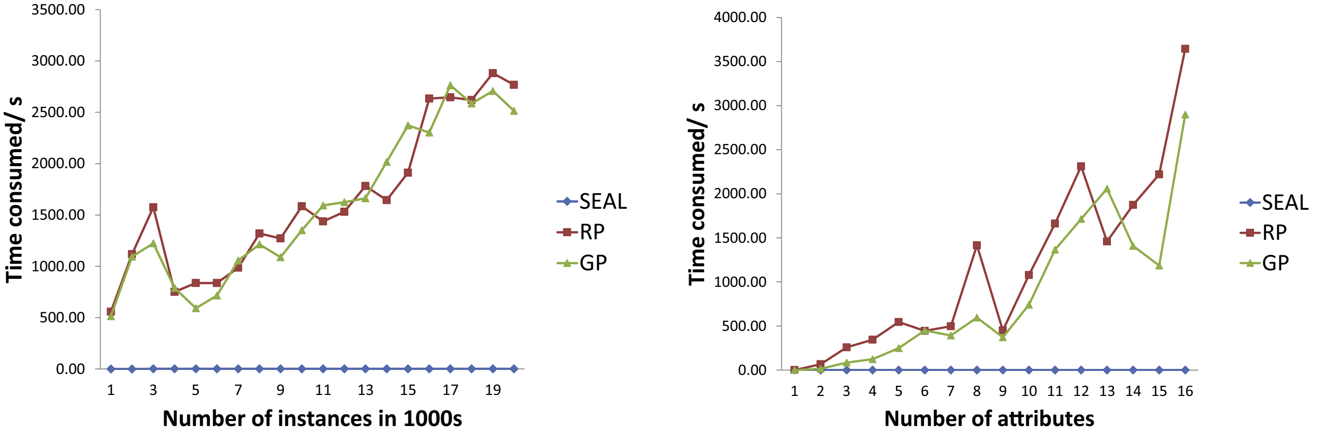
Both RP and GP show $O(n^2)$ time complexity to perturb one record with n attributes. The total complexity to perturb a dataset of m records is $O(m \times n^2)$. However, both RP and GP run for r number of iterations (which is taken as a user input) to find the optimal perturbation instance of the dataset within the r iterations. Therefore, the overall complexity is $O(m \times r \times n^2)$. Under each iteration of r, the algorithms run data reconstruction using ICA and known IO attacks to find the vulnerability level of the perturbed



(a) Change of the time elapsed for the LRDS dataset with an increasing number of instances.

(b) Change of the time elapsed for the LRDS dataset with an increasing number of attributes.

Fig. 7. Time consumption of SEAL. During the runtime analysis, the window size (ws) was maintained at a full length of the corresponding instance of LRDS. The privacy budget (ϵ) was maintained at 1.



(a) Increase of time consumption of SEAL, RP, and GP against the number of tuples.

(b) Time consumption of SEAL, RP, and GP against the number of attributes.

Fig. 8. Time consumption comparison of SEAL, RP, and GP. The time consumption plots available in Fig. 7 are plotted in comparison with the time consumption plots of RP and GP. Due to the extremely low time consumption of SEAL, its curves lie almost on the x-axis when drawn in a plot together with the others.

dataset. Each attack runs another k number of iterations (which is another user input) to reconstruct k number of instances. Usually, k is much larger than r. For one iteration of k, IO and ICA contribute a complexity of $O(n \times m)$ (Zarzoso et al., 2006). Hence, the overall complexity of RP or GP in producing an optimal perturbed dataset is equal to $O(m^2 \times r \times k \times n^3)$ which is a much larger computational complexity compared to the linear computational complexity of SEAL. Fig. 8 shows the time consumption plots of the three methods plotted together on the same figure. As shown on the figures, the curves of SEAL lie almost on the x-axis due to its extremely low time consumption compared to the other two methods.

5.2.7. *Scalability*

We conducted the scalability analysis of SEAL on an SGI UV3000 supercomputer (a detailed specification of the supercomputer is given in Section 5.1). SEAL was tested for its scalability on two large datasets: HPDS and HIDS. The results are given in Table 4. It is apparent that SEAL is more efficient than RP, GP, and DC; in fact, RP and GP did not even converge after 100 hours (the time limit of the batch scripts were set to 100 h). Both RP and GP use recursive loops to achieve optimal perturbation, which slows down the perturbation process. Therefore, RP and GP are not suitable for perturbing big data and data streams. DC is effective in perturbing big data, but SEAL performs better by providing better efficiency and utility.

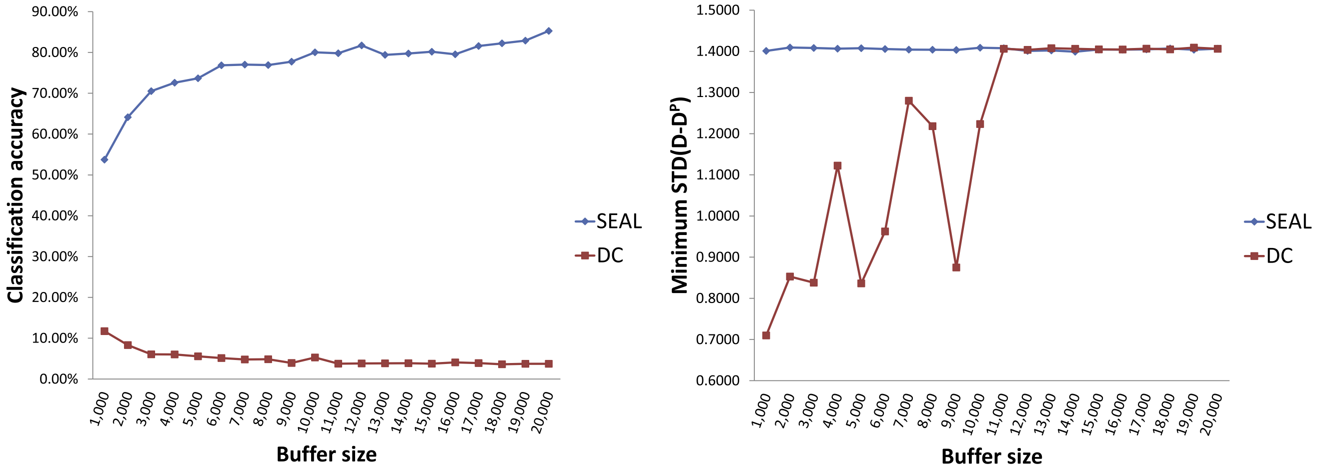
5.2.8. *Performance on data streams*

We checked the performance of SEAL on data streams with regard to (i) classification accuracy and (ii) *Minimum* $STD(D - D^p)$. The latter provides evidence to the minimum guarantee of attack resistance provided under a particular instance of perturbation. As shown in Fig. 9a, the classification accuracy of SEAL increases with increasing buffer size. This property is valuable for the perturbation of infinitely growing data streams generated by systems such as smart cyber-physical systems. The figure indicates that when a data stream grows infinitely, the use of smaller window sizes would negatively affect the utility of the perturbed data. When the window size is large, the utility of the perturbed data is closer to the utility of the original data stream. We can also notice that DC performs poorly in terms of classification accuracy compared to SEAL. It was previously noticed that DC works well only for tiny buffer sizes such as 5 or 10 (Chamikara et al., 2018). However, according to Fig. 9b, the minimum guarantee of attack resistance

Table 4
Scalability results (in seconds) of the three methods for high dimensional data.

Dataset	RP	GP	DC (k = 10,000)	SEAL (*ws* = 10,000)
HPDS	NC within 100 h	NC within 100 h	526.1168	97.8238
HIDS	NC within 100 h	NC within 100 h	6.42E+03	1.02E+03

NC: Did not converge.



(a) Change of classification accuracy against the buffer size.

(b) Change of *minimum* $STD(D - D^p)$ consumption against the buffer size.

Fig. 9. Dynamics of classification accuracy and the minimum STD(D-D^p) against increasing buffer size. During the experiments, the privacy budget (ϵ) was maintained at 1. The *minimum* $STD(D - D^p)$ represents the minimum (attribute) value of the standard deviation of the difference between the normalized attributes of the original data (LRDS dataset) and the perturbed data. The classification accuracy was obtained by classifying each perturbed datasets using the J48 classification algorithm.

drops when the buffer size decreases, which restricts the use of DC with smaller buffer sizes. According to Fig. 9b, however, SEAL still provides a consistent minimum guarantee of attack resistance, which allows SEAL to be used with any suitable buffer size.

6. Discussion

The proposed privacy-preserving mechanism (named SEAL) for big data and data streams performs data perturbation based on Chebyshev polynomial interpolation and the application of a Laplacian mechanism for noise addition. SEAL uses the first four orders of Chebyshev polynomials of the first kind for polynomial interpolation of a particular dataset. Although Legendre polynomials would offer a better approximation of the original data during interpolation, Chebyshev polynomials are simpler to calculate and provide improved privacy; a higher interpolation error, i.e. increased deviation from the original data would intuitively provide greater privacy than Legendre polynomials. Moreover, we intend to maintain the spatial arrangement of the original data, and this requirement is fully satisfied by Chebyshev interpolation. During the interpolation, SEAL adds calibrated noise using the Laplacian mechanism to introduce randomization, and henceforth privacy, to the perturbed data. The Laplacian noise allows the interpolation process to be performed with an anticipated random error for the root mean squared error minimization. We follow the conventions of differential privacy for noise addition, the introduction of noise is in accordance with the characteristic privacy budget ϵ. The privacy budget (ϵ) allows users (data curators) of SEAL to adjust the amount of noise. Smaller values of ϵ (usually less than 1 but greater than 0) add more noise to generate more randomization, whereas large values of ϵ add less noise and generate less randomization. The privacy budget is especially useful for multiple data release, where the data curator can apply proper noise in the perturbation process in consecutive data releases. SEAL's ability to maintain the shape of the original data distribution after noise addition is a clear advantage, and enables SEAL to provide convincingly higher utility than a standard local differentially private algorithm. This characteristic may come at a price, and the privacy enforced by a standard differentially private mechanism can be a little higher than that of SEAL.

The experimental results of SEAL show that it performs well on both static data and data streams. We evaluated SEAL in terms of classification accuracy, attack resistance, time complexity, scalability, and data stream performance. We tested each of these parameters using seven datasets, five classification algorithms, and three attack methods. SEAL outperforms the comparable methods: RP, GP, and DC in all these areas, proving that SEAL is an excellent choice for privacy preservation of data produced by SCPS and related technologies. SEAL produces high utility perturbed data in terms of classification accuracy, due to its ability to preserve the underlying characteristics such as the shape of the original data distribution. Although we apply an extensive amount of noise by using a small ϵ value, SEAL still tries to maintain the shape of the original data. The experiments show that even in extremely noisy perturbation environments, SEAL can provide higher utility compared to similar perturbation mechanisms, as shown in Section 5.1. SEAL shows excellent resistance with regard to data reconstruction attacks, proving that it offers excellent privacy. SEAL takes several steps to enhance the privacy of the perturbed data, namely (1) approximation through noisy interpolation, (2) scaling/normalization, and (3) data shuffling. These three steps help it outperform the other, similar perturbation mechanisms in terms of privacy.

In Section 5.1 we showed that SEAL has linear time complexity, $O(n)$. This characteristic is crucial for big data and data streams. The scalability experiments confirm that SEAL processes big datasets and data streams very efficiently. As shown in Fig. 9, SEAL also offers significantly better utility and attack resistance than data condensation. The amount of time spent by SEAL in processing one data record is around 0.03 to 0.09 milliseconds, which means that SEAL can perturb approximately 11110 to 33330 records per second. We note that runtime speed depends on the computing environment, such as CPU speed, memory speed, and disk IO speeds. The processing speed of SEAL in our experimental setup suits many practical examples of data streams, e.g. Sense your City (CITY)[5] and NYC Taxi cab (TAXI)[6] (Shukla and Simmhan, 2016). The results clearly demonstrate that SEAL is an efficient and reliable privacy preserving mechanism for practical big data and data stream scenarios.

7. Conclusion

In this paper, we proposed a solution for maintaining data privacy in large-scale data publishing and analysis scenarios, which is becoming an important issue in various environments, such as smart cyber-physical systems. We proposed a novel algorithm named SEAL to perturb data to maintain data privacy. Linear time complexity ($O(n)$) of SEAL allows it to work efficiently with continuously growing data streams and big data. Our experiments and comparisons indicate that SEAL produces higher classification accuracy, efficiency, and scalability while preserving better privacy with higher attack resistance than similar methods. The results prove that SEAL suits the dynamic environments presented by smart cyber-physical environments very well. SEAL can be an effective privacy-preserving mechanism for smart cyber-physical systems such as vehicles, grid, healthcare systems, and homes, as it can effectively perturb continuous data streams generated by sensors monitoring an individual or group of individuals and process them on the edge/fog devices before transmission to cloud systems for further analysis.

The current configuration of SEAL does not allow distributed data perturbation, and it limits the utility only to privacy-preserving data classification. A potential future extension of SEAL can address a distributed perturbation scenario that would allow SEAL to perturb sensor outputs individually while capturing the distinct latencies introduced by the sensors. SEAL could then combine the individually perturbed data using the corresponding timestamps and latencies to produce the privacy-protected data records. Further investigation on privacy parameter tuning would allow extended utility towards other areas such as descriptive statistics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Chebyshev polynomials of the first kind

Definition 3. The Chebyshev polynomial $T_n(x)$ of the first kind is a polynomial in x of degree n, defined by the relation,

$$T_n(x) = \cos n\theta \quad \text{when} \quad x = \cos\theta \tag{A.1}$$

From Eq. (A.1), we can deduce the first five ($n = 0, 1, 2, 3, 4$) Chebyshev polynomials using Eqs. (A.2) to (A.6), which are normalized such that $T_n(1) = 1$, and $x \in [-1, 1]$.

$$T_0(x) = 1 \tag{A.2}$$

$$T_1(x) = x \tag{A.3}$$

$$T_2(x) = 2x^2 - 1 \tag{A.4}$$

$$T_3(x) = 4x^3 - 3x \tag{A.5}$$

$$T_4(x) = 8x^4 - 8x^2 + 1 \tag{A.6}$$

Furthermore, we can represent any Chebyshev polynomial of the first kind using the recurrence relation given in Eq. (A.7), where $T_0(x) = 1$ and $T_1(x) = x$.

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{A.7}$$

Appendix B. Least square fitting

In least square fitting, vertical least squares fitting proceeds by finding the sum of squares of the vertical derivations R^2 (refer Eq. (B.1)) of a set of n data points (Weisstein, 2002).

$$R^2 \equiv \sum [f(x_i, a_1, a_2, \ldots, a_n) - y_i]^2 \tag{B.1}$$

Now, we can choose to minimize the quantity given in Eq. (B.2), which can be considered as an average approximation error. This is also referred to as the root mean square error in approximating (x_i, y_i) by a function $f(x_i, a_1, a_2, \ldots, a_n)$.

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[f(x_i, a_1, a_2, \ldots, a_n) - y_i]^2} \tag{B.2}$$

Let's assume that $f(x)$ is in a known class of functions, C. It can be shown that a function $\hat{f}*$ which is most likely to equal to f will also minimize Eq. (B.3) among all functions $\hat{f}(x)$ in C. This is called the least squares approximation to the data (x_i, y_i).

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\hat{f}(x_i, a_1, a_2, \ldots, a_n) - y_i\right]^2} \tag{B.3}$$

Minimizing E is equivalent to minimizing R^2, although the minimum values will be different. Thus we seek to minimize Eq. (B.1), which result in the condition given in Eq. (B.4) for $i = 1, \ldots, n$.

$$\frac{\partial(R^2)}{\partial a_i} = 0 \tag{B.4}$$

Let's consider $f(x) = mx + b$ for a linear fit. Thus we attempt to minimize Eq. (B.5), where b and m are allowed to vary arbitrarily.

$$R^2 = \sum_{i=1}^{n}[mx_i + b - y_i]^2 \tag{B.5}$$

Now, according to Eq. (B.4), the choices of b and m that minimize R^2 satisfy, Eqs. (B.6) and (B.7).

$$\frac{\partial R^2}{\partial b} = 0 \tag{B.6}$$

$$\frac{\partial R^2}{\partial m} = 0 \tag{B.7}$$

Eqs. (B.6) and (B.7) result in Eqs. (B.8) and (B.9).

$$\frac{\partial R^2}{\partial b} = \sum_{i=1}^{n}2[mx_i + b - y_i] \tag{B.8}$$

$$\frac{\partial R^2}{\partial m} = \sum_{i=1}^{n}2\left[mx_i^2 + bx_i - x_iy_i\right] \tag{B.9}$$

[5] Sense your City is an urban environmental monitoring project that used crowdsourcing to deploy sensors at 7 cities across 3 continents in 2015 with about 12 sensors per city, and it generates 7000 messages/sec.

[6] NYC Taxi cab (TAXI) produces a stream of smart transportation messages at the rate of 4000 messages/sec. The messages arrive from 2M trips taken on 20,355 New York city taxis equipped with GPS in 2013.

From Eqs. (B.8) and (B.9), we can generate the linear system shown in Eq. (B.10) which can be represented by the matrix form shown in Eq. (B.11). Now, we can solve Eq. (B.12), to find values of a and b to obtain the corresponding linear fit of $f(x) = mx + b$.

$$nb + \left(\sum_{i=1}^{n} x_i\right)m = \sum_{i=1}^{n} y_i \left(\sum_{i=1}^{n} x_i\right)b + \left(\sum_{i=1}^{n} x_i^2\right)m = \sum_{i=1}^{n} x_i y_i \tag{B.10}$$

$$\begin{bmatrix} n & \left(\sum_{i=1}^{n} x_i\right) \\ \left(\sum_{i=1}^{n} x_i\right) & \left(\sum_{i=1}^{n} x_i^2\right) \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix} \tag{B.11}$$

So,

$$\begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} n & \left(\sum_{i=1}^{n} x_i\right) \\ \left(\sum_{i=1}^{n} x_i\right) & \left(\sum_{i=1}^{n} x_i^2\right) \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix} \tag{B.12}$$

Appendix C. Privacy-preserving polynomial model generation

Consider a dataset $\{(x_i, y_i)|1 \leq i \leq n\}$, and let

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \cdots + a_m\varphi_m(x) \tag{C.1}$$

where, $a_1, a_2, \ldots, a_m$ are coefficients and $\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)$ are Chebeshev polynomials of first kind,

$$\varphi_1(x) = T_0(x) = 1 \tag{C.2}$$

$$\varphi_2(x) = T_1(x) = x \tag{C.3}$$

$$\varphi_n(x) = T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{C.4}$$

Assume that the data $\{x_i\}$ are chosen from an interval $[\alpha, \beta]$. The Chebyshev polynomials can be modified as given in Eq. (C.5),

$$\varphi_k(x) = T_{k-1}\left(\frac{2x - \alpha - \beta}{\beta - \alpha}\right) \tag{C.5}$$

The approximated function $\hat{f}$ of degree $(m-1)$ can be given by Eq. (C.1), where the degree of (φ_k) is $k-1$. We will assume the interval $[\alpha, \beta] = [0, 1]$ and construct the model accordingly. According to Eq. (C.6) when $[\alpha, \beta] = [0, 1]$, we get Eq. (C.6).

$$\varphi_k(x) = T_{k-1}\left(\frac{2x - \alpha - \beta}{\beta - \alpha}\right) = T_{k-1}(2x - 1) \tag{C.6}$$

From Eqs. (C.1) and (C.6) we have the following equations for $m = 4$.

$$\varphi_1(x) = T_0(2x - 1) = 1 \tag{C.7}$$

$$\varphi_2(x) = T_1(2x - 1) = 2x - 1 \tag{C.8}$$

$$\varphi_3(x) = T_2(2x - 1) = 8x^2 - 8x + 1 \tag{C.9}$$

$$\varphi_4(x) = T_3(2x - 1) = 32x^3 - 48x^2 + 18x - 1 \tag{C.10}$$

Eq. (C.12) defines $\hat{f}(x)$ when $m = 4$.

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + a_3\varphi_3(x) + a_4\varphi_4(x) \tag{C.11}$$

$$\hat{f}(x) = a_1(1) + a_2(2x - 1) + a_3(8x^2 - 8x + 1) + a_4(32x^3 - 48x^2 + 18x - 1) \tag{C.12}$$

Let the actual input be y_i, where $i = 1$ to n. The error of the approximated input can be determined by Eq. (C.13).

$$e_i = \hat{f}(x_i) - y_i \tag{C.13}$$

We need to determine the values of a_1, a_2, a_3, and a_4 in such a way that the errors (e_i) are small. In order to determine the best values for a_1, a_2, a_3, and a_4, we use the root mean square error given in Eq. (C.14).

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\hat{f}(x_i) - y_i\right]^2} \tag{C.14}$$

Let's take the least squares fitting of $\hat{f}(x)$ of the class of functions C which minimizes E as $\hat{f}^*(x)$. We can obtain $\hat{f}^*(x)$ by minimizing E. Thus we seek to minimize $M(a_1, a_2, a_3, a_4)$ which is given in Eq. (C.15).

$$M(a_1, a_2, a_3, a_4) = \sum_{i=1}^{n}[a_1 + a_2(2x - 1) + a_3(8x^2 - 8x + 1) + a_4(32x^3 - 48x^2 + 18x - 1) - y_i]^2 \tag{C.15}$$

The values of a_1, a_2, a_3, and a_4 that minimize $M(a_1, a_2, a_3, a_4)$ will satisfy the expressions given in Eqs. (C.16)–(C.19).

$$\frac{\partial M(a_1, a_2, a_3, a_4)}{\partial a_1} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) - y_i\right]^2\right)}{\partial a_1} = 0 \tag{C.16}$$

$$\frac{\partial M(a_1, a_2, a_3, a_4)}{\partial a_2} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) - y_i\right]^2\right)}{\partial a_2} = 0 \tag{C.17}$$

$$\frac{\partial M(a_1, a_2, a_3, a_4)}{\partial a_3} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) - y_i\right]^2\right)}{\partial a_3} = 0 \tag{C.18}$$

$$\frac{\partial M(a_1, a_2, a_3, a_4)}{\partial a_4} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) - y_i\right]^2\right)}{\partial a_4} =0 \tag{C.19}$$

C1. Utilizing differential privacy to introduce randomization to the approximation process

To decide the amount of noise, we have to determine the sensitivity of the noise addition process. Given that we add the noise to the approximated values of $f(\hat{x})$, the sensitivity (Δf) can be defined using Eq. (C.20), which is the maximum difference between the highest and the lowest possible output values of $f(\hat{x})$. Since the input dataset is normalized within the bounds of 0 and 1, the minimum possible input or output is 0 while the maximum possible input or output is 1. Therefore, we define the sensitivity of the noise addition process to be 1.

$$\Delta f = \|max(y_i) - min(y_{i+1})\|_1 = (1 - 0) = 1 \tag{C.20}$$

Now we add random Laplacian to each expression given in Eqs. (C.16)–(C.19), according to Eq. (C.21) with a sensitivity (Δf) of 1 and a privacy budget of ϵ as shown in Eqs. (C.22), (C.25), (C.28) and (C.31). In the process of adding the Laplacian noise, we choose Laplacian noise with a position (location) of 0 as the idea is to keep the local minima of RMSE around 0 during the process of interpolation. Here we try to randomize the process of obtaining the local minima of the mean squared error to generate the value for the coefficients (a_1, a_2, a_3, a_4) with randomization. The differentially private Laplacian mechanism can be represented by Eq. (C.21), where Δf is the sensitivity of the process, and ϵ is the privacy budget.

$$PF(D) = \frac{\epsilon}{2\Delta f} e^{-\frac{|x-F(D)|}{\Delta F}} \tag{C.21}$$

Eq. (C.22) shows the process of using the Laplacian mechanism to introduce noise to the RMSE minimization of the polynomial interpolation process. Here, we try to introduce an error to the partial derivative of Eq. (C.15) with respect to a_1. By doing so, it guarantees that Eq. (C.22) contributes with an error to the process of finding the coefficients for a_1, a_2, a_3, and a_4, which is given in Eq. (C.36). Since the sensitivity (Δf) of the noise addition process is 1, as defined in Eq. (C.20), the scale (spread) of the Laplacian noise is $1/\epsilon$. We restrict the position (μ) of the Laplacian noise to 0 as the goal is to achieve the global minima keeping the RMSE at 0 after the randomization.

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_1} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i(\frac{\Delta f}{\epsilon}) - y_i\right]^2\right)}{\partial a_1} = 0 \tag{C.22}$$

After applying the partial derivation on Eq. (C.22) with respect to a_1, we can obtain Eq. (C.23) which leads to obtaining Eq. (C.24).

$$\sum_{i=1}^{n} 2\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i\left(\frac{\Delta f}{\epsilon}\right) - y_i\right] = 0 \tag{C.23}$$

Let's use m_{ij} to denote the coefficients, and b_i to represent the constants in the right hand side of the equal symbol in the factorised Eqs. (C.24), (C.27), (C.30), and (C.33).

$$\begin{aligned} & a_1 \underbrace{n}_{m_{11}} + a_2 \underbrace{\left(2\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{12}} + a_3 \underbrace{\left(8\left(\sum_{i=1}^{n} x_i^2\right) - 8\left(\sum_{i=1}^{n} x_i\right) + n\right)}_{m_{13}} \\ & \quad + a_4 \underbrace{\left(32\left(\sum_{i=1}^{n} x_i^3\right) - 48\left(\sum_{i=1}^{n} x_i^2\right) + 18\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{14}} = \underbrace{\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} Lap_i\left(\frac{\Delta f}{\epsilon}\right)}_{b_1} \end{aligned} \tag{C.24}$$

We repeat the same process of applying noise in Eq. (C.22) to apply noise to Eq. (C.17).

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_2} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i(\frac{\Delta f}{\epsilon}) - y_i\right]^2\right)}{\partial a_2} = 0 \tag{C.25}$$

After solving Eq. (C.25), we can obtain Eq. (C.26) which leads to Eq. (C.27).

$$\sum_{i=1}^{n} 2\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i\left(\frac{\Delta f}{\epsilon}\right) - y_i\right](2x_i - 1) = 0 \tag{C.26}$$

$$\begin{aligned} & a_1 \underbrace{\left(2\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{21}} + a_2 \underbrace{\left(4\left(\sum_{i=1}^{n} x_i^2\right) - 4\left(\sum_{i=1}^{n} x_i\right) + n\right)}_{m_{22}} + a_3 \underbrace{\left(16\left(\sum_{i=1}^{n} x_i^3\right) - 24\left(\sum_{i=1}^{n} x_i^2\right) + 10\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{23}} \\ & \quad + a_4 \underbrace{\left(64\left(\sum_{i=1}^{n} x_i^4\right) - 128\left(\sum_{i=1}^{n} x_i^3\right) + 84\left(\sum_{i=1}^{n} x_i^2\right) - 20\left(\sum_{i=1}^{n} x_i\right) + n\right)}_{m_{24}} \\ & \quad = \underbrace{2\left(\sum_{i=1}^{n} x_i y_i\right) - \sum_{i=1}^{n} y_i - \left(2\left(\sum_{i=1}^{n} x_i Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) - \sum_{i=1}^{n} Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right)}_{b_2} \end{aligned} \tag{C.27}$$

Similarly, we can obtain Eqs. (C.30) and (C.33) after introducing the calibrated Laplacian noise to Eqs. (C.18) and (C.19) respectively.

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_3} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i(\frac{\Delta f}{\epsilon}) - y_i\right]^2\right)}{\partial a_3} = 0 \tag{C.28}$$

$$\sum_{i=1}^{n} 2\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i\left(\frac{\Delta f}{\epsilon}\right) - y_i\right](8x^2 - 8x + 1) = 0 \tag{C.29}$$

$$a_1 \underbrace{\left(8\left(\sum_{i=1}^{n} x_i^2\right) - 8\left(\sum_{i=1}^{n} x_i\right) + n\right)}_{m_{31}} + a_2 \underbrace{\left(16\left(\sum_{i=1}^{n} x_i^3\right) - 24\left(\sum_{i=1}^{n} x_i^2\right) + 10\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{32}}$$

$$+ a_3 \underbrace{\left(64\left(\sum_{i=1}^{n} x_i^4\right) - 128\left(\sum_{i=1}^{n} x_i^3\right) + 80\left(\sum_{i=1}^{n} x_i^2\right) - 16\left(\sum_{i=1}^{n} x_i\right) + n\right)}_{m_{33}}$$

$$+ a_4 \underbrace{\left(256\left(\sum_{i=1}^{n} x_i^5\right) - 640\left(\sum_{i=1}^{n} x_i^4\right) + 560\left(\sum_{i=1}^{n} x_i^3\right) - 200\left(\sum_{i=1}^{n} x_i^2\right) + 26\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{34}}$$

$$= \underbrace{\begin{gathered} 8\left(\sum_{i=1}^{n} x_i^2 y_i\right) - 8\left(\sum_{i=1}^{n} x_i Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) + \sum_{i=1}^{n} y_i \\ -\left(8\left(\sum_{i=1}^{n} x_i^2 Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) - 8\left(\sum_{i=1}^{n} x_i Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) + \sum_{i=1}^{n} Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) \end{gathered}}_{b_3} \tag{C.30}$$

$$\frac{\partial M(a_1, a_2, a_3, a_4)}{\partial a_4} = \frac{\partial\left(\sum_{i=1}^{n}\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i\left(\frac{\Delta f}{\epsilon}\right) - y_i\right]^2\right)}{\partial a_4} = 0 \tag{C.31}$$

$$\sum_{i=1}^{n} 2\left[a_1 + a_2(2x-1) + a_3(8x^2-8x+1) + a_4(32x^3-48x^2+18x-1) + Lap_i\left(\frac{\Delta f}{\epsilon}\right) - y_i\right](32x^3-48x^2+18x-1) = 0 \tag{C.32}$$

$$a_1 \underbrace{\left(32\left(\sum_{i=1}^{n} x_i^3\right) - 48\left(\sum_{i=1}^{n} x_i^2\right) + 18\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{41}} + a_2 \underbrace{\left(64\left(\sum_{i=1}^{n} x_i^4\right) - 128\left(\sum_{i=1}^{n} x_i^3\right) + 84\left(\sum_{i=1}^{n} x_i^2\right) - 20\left(\sum_{i=1}^{n} x_i\right) + n\right)}_{m_{42}}$$

$$+ a_3 \underbrace{\left(256\left(\sum_{i=1}^{n} x_i^5\right) - 640\left(\sum_{i=1}^{n} x_i^4\right) + 560\left(\sum_{i=1}^{n} x_i^3\right) - 200\left(\sum_{i=1}^{n} x_i^2\right) + 26\left(\sum_{i=1}^{n} x_i\right) - n\right)}_{m_{43}}$$

$$+ a_4 \underbrace{\left(1024\left(\sum_{i=1}^{n} x_i^6\right) - 3072\left(\sum_{i=1}^{n} x_i^5\right) + 3456\left(\sum_{i=1}^{n} x_i^4\right) - 1792\left(\sum_{i=1}^{n} x_i^3\right) + 420\left(\sum_{i=1}^{n} x_i^2\right) - 36\left(\sum_{i=1}^{n} x\right) + n\right)}_{m_{44}}$$

$$= \underbrace{\begin{gathered} 32\left(\sum_{i=1}^{n} x_i^3 y_i\right) - 48\left(\sum_{i=1}^{n} x_i^2 y_i\right) + 18\left(\sum_{i=1}^{n} x_i y_i\right) - \sum_{i=1}^{n} y_i \\ -\left(32\left(\sum_{i=1}^{n} x_i^3 Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) - 48\left(\sum_{i=1}^{n} x_i^2 Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) + 18\left(\sum_{i=1}^{n} x_i Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) - \sum_{i=1}^{n} Lap_i\left(\frac{\Delta f}{\epsilon}\right)\right) \end{gathered}}_{b_4} \tag{C.33}$$

Let's consider the coefficients (m_{ij}) and the constants (b_i) in the Eqs. (C.24), (C.27), (C.30), and (C.33). Using m_{ij} and b_i we can form a linear system which can be denoted by Eq. (C.34).

$$CA = B \tag{C.34}$$

Where,

$$C = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \tag{C.35}$$

$$A = [a_1, a_2, a_3, a_4]^T \tag{C.36}$$

$$B = [b_1, b_2, b_3, b_4]^T \tag{C.37}$$

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cose.2019.101570.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L., 2016. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 308–318.

Aggarwal, C.C., 2005. On *k*-anonymity and the curse of dimensionality. In: Proceedings of the 31st International Conference on Very Large Data Bases. VLDB Endowment, pp. 901–909.

Aggarwal, C.C., Philip, S.Y., 2008. A general survey of privacy-preserving data mining models and algorithms. In: Privacy-Preserving Data Mining. Springer, pp. 11–52.

Aggarwal, C.C., Yu, P.S., 2004. A condensation approach to privacy preserving data mining. In: EDBT, Volume 4. Springer, pp. 183–199.

Aggarwal, C.C., Yu, P.S., 2008. On static and dynamic methods for condensation-based privacy-preserving data mining. ACM Trans. Database Syst. (TODS) 33 (1), 2.

Agrawal, R., Srikant, R., 2000. Privacy-preserving data mining. In: ACM Sigmod Record, Volume 29. ACM, pp. 439–450.

Agrawal, S., Haritsa, J.R., 2005. A framework for high-accuracy privacy-preserving mining. In: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. IEEE, pp. 193–204.

Aldeen, Y.A.A.S., Salleh, M., Razzaque, M.A., 2015. A comprehensive review on privacy preserving data mining. SpringerPlus 4 (1), 694.

Alhayajneh, A., Baccarini, A., Weiss, G., Hayajneh, T., Farajidavar, A., 2018. Biometric authentication and verification for medical cyber physical systems. Electronics 7 (12), 436.

Backes, M., Berrang, P., Goga, O., Gummadi, K.P., Manoharan, P., 2016. On profile linkability despite anonymity in social media systems. In: Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society. ACM, pp. 25–35.

Baheti, R., Gill, H., 2011. Cyber-physical systems. ImpactControl Technol. 12 (1), 161–166.

Balandina, E., Balandin, S., Koucheryavy, Y., Mouromtsev, D., 2015. Iot use cases in healthcare and tourism. In: Business Informatics (CBI), 2015 IEEE 17th Conference on, Volume 2. IEEE, pp. 37–44.

Bertino, E., 2016. Data privacy for IOT systems: concepts, approaches, and research directions. In: Big Data (Big Data), 2016 IEEE International Conference on. IEEE, pp. 3645–3647.

Bertino, E., Lin, D., Jiang, W., 2008. A survey of quantification of privacy preserving data mining algorithms. In: Privacy-Preserving Data Mining. Springer, pp. 183–205.

Cao, J., Carminati, B., Ferrari, E., Tan, K.L., 2011. Castle: Continuously anonymizing data streams. IEEE Trans. Dependable Secure Comput. 8 (3), 337–352.

Chamikara, M.A.P., Bertok, P., Liu, D., Camtepe, S., Khalil, I., 2018. Efficient data perturbation for privacy preserving and accurate data stream mining. Pervasive Mob. Comput. doi:10.1016/j.pmcj.2018.05.003.

Chamikara, M.A.P., Bertok, P., Liu, D., Camtepe, S., Khalil, I., 2019. Efficient privacy preservation of big data for accurate data mining. Inf. Sci. doi:10.1016/j.ins.2019.05.053.

Chan, T.H.H., Li, M., Shi, E., Xu, W., 2012. Differentially private continual monitoring of heavy hitters from distributed streams. In: International Symposium on Privacy Enhancing Technologies Symposium. Springer, pp. 140–159.

Chen, K., Liu, L., 2005a. Privacy preserving data classification with rotation perturbation. In: Data Mining, Fifth IEEE International Conference on. IEEE, pp. 1–4.

Chen K., Liu L.. A random rotation perturbation approach to privacy preserving data classification2005b;https://corescholar.libraries.wright.edu/knoesis/916/.

Chen, K., Liu, L., 2011. Geometric data perturbation for privacy preserving outsourced data mining. Knowl. Inf. Syst. 29 (3), 657–695.

Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., Wang, T., 2018. Privacy at scale: local differential privacy in practice. In: Proceedings of the 2018 International Conference on Management of Data. ACM, pp. 1655–1658.

Datta, S., 2004. On random additive perturbation for privacy preserving data mining. University of Maryland, Baltimore County.

De Francisci Morales, G., Bifet, A., Khan, L., Gama, J., Fan, W., 2016. IOT big data stream mining. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 2119–2120.

Domingo-Ferrer, J., Mateo-Sanz, J.M., 2002. Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans. Knowl. Data Eng. 14 (1), 189–201.

Domingo-Ferrer, J., Soria-Comas, J., 2017. Steered microaggregation: a unified primitive for anonymization of data sets and data streams. In: Data Mining Workshops (ICDMW), 2017 IEEE International Conference on. IEEE, pp. 995–1002.

Du, W., Zhan, Z., 2003. Using randomized response techniques for privacy-preserving data mining. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 505–510.

Dwork, C., 2008. Differential privacy: a survey of results. In: International Conference on Theory and Applications of Models of Computation. Springer, pp. 1–19.

Dwork, C., 2009. The differential privacy frontier. In: Theory of Cryptography Conference. Springer, pp. 496–502.

Dwork, C., Roth, A., et al., 2014. The algorithmic foundations of differential privacy. Found. Trends®Theor. Comput. Sci. 9 (3–4), 211–407.

Erlingsson, U., Pihur, V., Korolova, A., 2014. Rappor: Rr aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 1054–1067.

Estivill-Castro, V., Brankovic, L., 1999. Data swapping: balancing privacy against precision in mining for logic rules. In: DaWaK, Volume 99. Springer, pp. 389–398.

Fernando, R., Ranchal, R., An, B., Othman, L.B., Bhargava, B., 2016. Consumer oriented privacy preserving access control for electronic health records in the cloud. In: Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on. IEEE, pp. 608–615.

Fox, J.A., 2015. Randomized response and related methods: surveying sensitive data, volume 58. SAGE Publications.

Friedman, A., Schuster, A., 2010. Data mining with differential privacy. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 493–502.

Gai, K., Qiu, M., Zhao, H., Xiong, J., 2016. Privacy-aware adaptive data encryption strategy of big data in cloud computing. In: Cyber Security and Cloud Computing (CSCloud), 2016 IEEE 3rd International Conference on. IEEE, pp. 273–278.

Ganta, S.R., Kasiviswanathan, S.P., Smith, A., 2008. Composition attacks and auxiliary information in data privacy. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 265–273.

Gävert, H., Hurri, J., Särelä, J., Hyvärinen, A., 2005. The fastICA package for MATLAB. Lab. Comput. Inf. Sci. Helsinki Univ. Technol.. https://research.ics.aalto.fi/ica/fastica/.

Hammad, M.A., Franklin, M.J., Aref, W.G., Elmagarmid, A.K., 2003. Scheduling for shared window joins over data streams. In: Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, pp. 297–308.

Howell, D.C., 2016. Fundamental Statistics for the Behavioral Sciences. Nelson Education.

Huang, Z., Du, W., Chen, B., 2005. Deriving private information from randomized data. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. ACM, pp. 37–48.

Kairouz, P., Oh, S., Viswanath, P., 2014. Extremal mechanisms for local differential privacy. In: Advances in Neural Information Processing Systems, pp. 2879–2887.

Kellaris, G., Papadopoulos, S., Xiao, X., Papadias, D., 2014. Differentially private event sequences over infinite streams. Proc. VLDB Endow. 7 (12), 1155–1166.

Kerschbaum, F., Härterich, M., 2017. Searchable encryption to reduce encryption degradation in adjustably encrypted databases. In: IFIP Annual Conference on Data and Applications Security and Privacy. Springer, pp. 325–336.

Kieseberg, P., Weippl, E., 2018. Security challenges in cyber-physical production systems. In: International Conference on Software Quality. Springer, pp. 3–16.

Kirkham, T., Sinha, A., Parlavantzas, N., Kryza, B., Fremantle, P., Kritikos, K., Aziz, B., 2015. Privacy aware on-demand resource provisioning for IOT data processing. In: International Internet of Things Summit. Springer, pp. 87–95.

Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C., 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. Eur. J. Oper. Res. 247 (1), 124–136.

Li, F., Sun, J., Papadimitriou, S., Mihaila, G.A., Stanoi, I., 2007a. Hiding in the crowd: privacy preservation on evolving streams through correlation tracking. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, pp. 686–695.

Li, J., Lin, D., Squicciarini, A.C., Li, J., Jia, C., 2017. Towards privacy-preserving storage and retrieval in multiple clouds. IEEE Trans. Cloud Comput. 5 (3), 499–509. doi:10.1109/TCC.2015.2485214.

Li, N., Li, T., Venkatasubramanian, S., 2007b. t-closeness: privacy beyond k-anonymity and l-diversity. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, pp. 106–115.

Liu, J., Xiao, Y., Li, S., Liang, W., Chen, C.L.P., 2012. Cyber security and privacy issues in smart grids. IEEE Commun. Surv. Tutor. 14 (4), 981–997.

Liu, K., Kargupta, H., Ryan, J., 2006. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. IEEE Trans. Knowl. Data Eng. 18 (1), 92–106.

Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M., 2006. L-diversity: Privacy beyond k-anonymity. In: Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on. IEEE. 24–24

Machanavajjhala, A., Kifer, D., 2015. Designing statistical privacy for your data. Commun. ACM 58 (3), 58–67.

Mason, J.C., Handscomb, D.C., 2002. Chebyshev Polynomials. Chapman and Hall/CRC.

Mivule, K., Turner, C., 2013. A comparative analysis of data privacy and utility parameter adjustment, using machine learning classification as a gauge. Procedia Comput. Sci. 20, 414–419.

Mohammed, N., Chen, R., Fung, B., Yu, P.S., 2011. Differentially private data release for data mining. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 493–501.

Muralidhar, K., Parsa, R., Sarathy, R., 1999. A general additive data perturbation method for database security. Manage. Sci. 45 (10), 1399–1415.

Niu, B., Li, Q., Zhu, X., Cao, G., Li, H., 2014. Achieving k-anonymity in privacy-aware location-based services. In: INFOCOM, 2014 Proceedings IEEE. IEEE, pp. 754–762.

Okkalioglu, B.D., Okkalioglu, M., Koc, M., Polat, H., 2015. A survey: deriving private information from perturbed data. Artif. Intell. Rev. 44 (4), 547–569.

Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., Ren, K., 2016. Heavy hitter estimation over set-valued data with local differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 192–203.

Quinlan, J.R., 1993. C4. 5: Programming for Machine Learning, 38. Morgan Kauffmann.

Razaque, A., Rizvi, S.S., 2016. Triangular data privacy-preserving model for authenticating all key stakeholders in a cloud environment. Comput. Secur. 62, 328–347.

Razaque, A., Rizvi, S.S., 2017. Privacy preserving model: a new scheme for auditing cloud stakeholders. J. Cloud Comput. 6 (1), 7.

Schölkopf, B., Burges, C.J.C., Smola, A.J., 1999. Advances in Kernel Methods: Support Vector Learning. MIT press.

Scholz, F.W., Maximum Likelihood Estimation; Wiley Online Library. doi:10.1002/0471667196.ess1571.pub2.

Shukla, A., Simmhan, Y., 2016. Benchmarking distributed stream processing platforms for IOT applications. In: Technology Conference on Performance Evaluation and Benchmarking. Springer, pp. 90–106.

Sridhar, S., Hahn, A., Govindarasu, M., et al., 2012. Cyber-physical system security for the electric power grid. Proc. IEEE 100 (1), 210–224.

Vatsalan, D., Sehili, Z., Christen, P., Rahm, E., 2017. Privacy-preserving record linkage for big data: current approaches and research challenges. In: Handbook of Big Data Technologies. Springer, pp. 851–895.

Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y., 2004. State-of-the-art in privacy preserving data mining. ACM SIGMOD Record 33 (1), 50–57.

Wang, W., Chen, L., Zhang, Q., 2015. Outsourcing high-dimensional healthcare data to cloud with personalized privacy preservation. Comput. Netw. 88, 136–148.

Wang, X., Zhang, J., Schooler, E.M., Ion, M., 2014. Performance evaluation of attribute-based encryption: toward data privacy in the IOT. In: Communications (ICC), 2014 IEEE International Conference on. IEEE, pp. 725–730.

Wang, Y., Wu, X., Hu, D., 2016. Using randomized response for differential privacy preserving data collection. EDBT/ICDT Workshops, Volume 1558.

Weisstein E.W.. Least squares fitting2002;.

Wen, Y., Liu, J., Dou, W., Xu, X., Cao, B., Chen, J., 2018. Scheduling workflows with privacy protection constraints for big data applications on cloud. Future Gener. Comput. Syst. doi:10.1016/j.future.2018.03.028. http://www.sciencedirect.com/science/article/pii/S0167739X17307379.

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., 2016. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

Wold, S., Esbensen, K., Geladi, P., 1987. Principal component analysis. Chemom. Intell. Lab. Syst. 2 (1-3), 37–52.

Wong, R.C.W., Fu, A.W.C., Wang, K., Yu, P.S., Pei, J., 2011. Can the utility of anonymized data be used for privacy breaches? ACM Trans. Knowl. Discov. Data (TKDD) 5 (3), 16.

Wong, R.C.W., Li, J., Fu, A.W.C., Wang, K., 2006. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 754–759.

Wu, D., Yang, B., Wang, R., 2016. Scalable privacy-preserving big data aggregation mechanism. Digit. Commun. Netw. 2 (3), 122–129.

Xu, H., Guo, S., Chen, K., 2014. Building confidential and efficient query services in the cloud with rasp data perturbation. IEEE Trans. Knowl. Data Eng. 26 (2), 322–335.

Xu, Y., Wang, K., Fu, A.W.C., She, R., Pei, J., 2008. Privacy-preserving data stream classification. In: Privacy-Preserving Data Mining. Springer, pp. 487–510.

Xue, M., Papadimitriou, P., Raïssi, C., Kalnis, P., Pung, H.K., 2011. Distributed privacy preserving data collection. In: International Conference on Database Systems for Advanced Applications. Springer, pp. 93–107.

Yang, K., Han, Q., Li, H., Zheng, K., Su, Z., Shen, X., 2017. An efficient and fine-grained big data access control scheme with privacy-preserving policy. IEEE Internet Things J. 4 (2), 563–571.

Zarzoso, V., Comon, P., Kallel, M., 2006. How fast is fastICA? In: 2006 14th European Signal Processing Conference. IEEE, pp. 1–5.

Zhang, L., Jajodia, S., Brodsky, A., 2007. Information disclosure under realistic assumptions: privacy versus optimality. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. ACM, pp. 573–583.

Zhang, Q., Yang, L.T., Chen, Z., 2016a. Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans. Comput. 65 (5), 1351–1362.

Zhang, X., Dou, W., Pei, J., Nepal, S., Yang, C., Liu, C., Chen, J., 2015. Proximity-aware local-recoding anonymization with mapreduce for scalable big data privacy preservation in cloud. IEEE Trans. Comput. 64 (8), 2293–2307.

Zhang, Y., Tong, W., Zhong, S., 2016b. On designing satisfaction-ratio-aware truthful incentive mechanisms for k-anonymity location privacy. IEEE Trans. Inf. ForensicsSecur. 11 (11), 2528–2541.

Zhong, J., Mirchandani, V., Bertok, P., Harland, J., 2012. μ-fractal based data perturbation algorithm for privacy protection. In: PACIS, p. 148.

M.A.P. Chamikara is a Ph.D. researcher in Computer Science and Software Engineering at the School of Science, RMIT University, Australia. He is also a researcher at CSIRO Data61, Melbourne, Australia. He received his M.Phil. in Computer Science from the University of Peradeniya, Sri Lanka in 2015. His research interests include information privacy and security, data mining, artificial neural networks, and fuzzy logic.

P. Bertok is an associate professor in the School of Science at RMIT University, Melbourne, Australia, where he is a member of the Cyberspace & Security Group (CSG). He received his Ph.D. in computer engineering from the University of Tokyo, Japan. His research interests include access control, privacy protection and communication security.

D. Liu is a senior research scientist at CSIRO Data61. He received his Ph.D. in Computer Science and Engineering from Shanhai Jiao Tong University, China. Dongxi Liu joined CSIRO in March 2008. Before that, he was a Researcher in the University of Tokyo from Feb 2004 to March 2008, and a Research Fellow in National University of Singapore from December 2002 to December 2003. His current research focuses on lightweight encryption for IoT security and encrypted data processing for cloud security.

S. Camtepe is a senior research scientist at CSIRO Data61. He received his Ph.D. in computer science from Rensselaer Polytechnic Institute, New York, USA, in 2007. From 2007 to 2013, he was with the Technische Universitaet Berlin, Germany, as a Senior Researcher and Research Group Leader in Security. From 2013 to 2017, he worked as a lecturer at the Queensland University of Technology, Australia. His research interests include mobile and wireless communication, pervasive security and privacy, and applied and malicious cryptography.

I. Khalil is an associate professor in the School of Science at RMIT University, Melbourne, Australia. Ibrahim obtained his Ph.D. in 2003 from the University of Berne in Switzerland. Before joining RMIT University Ibrahim also worked for EPFL and University of Berne in Switzerland and Osaka University in Japan. He has several years of experience in Silicon Valley based companies working on Large Network Provisioning and Management software. His research interests are in scalable efficient computing in distributed systems, network and data security, secure data analysis including big data security, steganography of wireless body sensor networks and highspeed sensor streams and smart grids.