PLOS ONE

RESEARCH ARTICLE

Anomaly detection over differential preserved privacy in online social networks

Randa Aljably[1,2☯], **Yuan Tian**[3☯], **Mznah Al-Rodhaan**[2*], **Abdullah Al-Dhelaan**[2☯]

1 Computer Department, Shaqra University, Druma, Kingdom of Saudi Arabia, **2** Computer Science Department, King Saud University, Riyadh, Kingdom of Saudi Arabia, **3** Nanjing Institute of Technology, Nanjing, China

☯ These authors contributed equally to this work.
* rodhaan@ksu.edu.sa

Citation: Aljably R, Tian Y, Al-Rodhaan M, Al-Dhelaan A (2019) Anomaly detection over differential preserved privacy in online social networks. PLoS ONE 14(4): e0215856. https://doi.org/10.1371/journal.pone.0215856

Editor: He Debiao, Wuhan University, CHINA

Received: December 7, 2018

Accepted: April 9, 2019

Published: April 25, 2019

Data Availability Statement: The data underlying this study have been uploaded to GitHub and are accessible using the following link: https://github.com/Randoot/Local-Differential-Privacy.

Funding: This work was supported by the Deanship of Scientific Research at King Saud University (https://dsrs.ksu.edu.sa/en), Riyadh, Saudi Arabia through research group no. RGP-264.

Competing interests: The authors have declared that no competing interests exist.

Abstract

The massive reach of social networks (SNs) has hidden their potential concerns, primarily those related to information privacy. Users increasingly rely on social networks for more than merely interactions and self-representation. However, social networking environments are not free of risks. Users are often threatened by privacy breaches, unauthorized access to personal information, and leakage of sensitive data. In this paper, we propose a privacy-preserving model that sanitizes the collection of user information from a social network utilizing restricted local differential privacy (LDP) to save synthetic copies of collected data. This model further uses reconstructed data to classify user activity and detect abnormal network behavior. Our experimental results demonstrate that the proposed method achieves high data utility on the basis of improved privacy preservation. Moreover, LDP sanitized data are suitable for use in subsequent analyses, such as anomaly detection. Anomaly detection on the proposed method's reconstructed data achieves a detection accuracy similar to that on the original data.

Introduction

Information sharing platforms, such as online social networks (OSNs), have experienced remarkable growth and recognition in recent years. Notably, OSN platforms have direct access to the public and private data of their users [1]. In some cases, these data are shared with other parties to carry out analytical and social research. Although the release of social network data is considered a severe breach of privacy, OSN platforms reassure their users by anonymizing their data before sharing it. Unfortunately, data mining techniques can be used to infer sensitive information from released data. Therefore, it is necessary to sanitize network data before releasing it [2].

Moreover, an increasing number of attacks target personal user information on OSNs [3]. Thus, there is an urgent need for radical improvements in OSN security and privacy measures. Most previous studies on the preserved privacy of published data deal only with relational data and cannot be applied to social network data [4].

Therefore, we have taken the initiative toward preserving privacy in social network data. For each user, we use an activity profile to represent his/her sequence of data. With our model, we aim to investigate the application of LDP to user activity logs. In this model, a data collection server uses a specific partitioning of privacy levels to create Laplacian random noise. However, not all user data are stored in SN repositories; only a predetermined set is selected amongst the salient points representing the data sequence. On the other hand, the data analyzer sub-model reverses these disrupted points to reconstruct the original stored data received from the repositories. Moreover, the data analyzer uses the resulting noisy data to detect anomalous behavior. The data analyzers in the proposed model utilize an extension of the conventional LDP to carry out anomaly detection on reconstructed SN data.

In this paper, our contributions are summarized as follows:

1. We propose a model that protects user privacy in SNs compared with other solutions where sensitive user information is poorly anonymized and can be inferred using data mining. We guarantee a stronger degree of privacy and a lower expected error caused by large data streams. Our privacy preserving model applies Laplace's probability distribution function (PDF) to generate random noise. To guarantee privacy for each user, this noise is calculated using the user's data. In addition, it protects not only user profiles but also user activity.
2. We achieve an improved estimation error of % 0.15 over direct LDP estimation [5]. In the direct application of LDP to data, the estimated error is linearly proportional to the size of the data set. Since SN data are highly scalable, the direct LDP approach results in relatively high expected error [6].
3. We conduct experiments on real-world datasets, showing that the proposed framework guarantees privacy and achieves modest overhead performance.
4. We significantly reduce analysis costs. In our algorithm, only selected data are sent to the detection model, which estimates the data required for classification.

This paper is organized as follows. In Section 2, we review related work in LDP privacy preservation in social networks. In Section 3, we formulate the problem and demonstrate a threat model. Section 4 introduces the scientific framework and preliminaries. The proposed model is explained in Section 5, followed by experimental results and a discussion in Section 6. We conclude and present our potential future work in Section 7.

Threat model

The data repositories in an SN collect and store everything related to its users. Logs may contain the user profiles, activities, and networks of other users and may also store information created without user involvement. Sometimes, the SN shares an anonymized version of this information with other parties for different purposes. Unfortunately, as several recent incidents have demonstrated, releasing even anonymized graphs may lead to the re-identification of individuals within the network [7] and the disclosure of confidential information, which has severe consequences for those involved.

Scientific analysis is known to be vulnerable to the identification of individuals and extraction of private data. However, when a specific breach of privacy was tackled with continuous research and proposed solutions, it was shown that data for analysis might be safely released under differential-privacy guarantees [5, 8]. Since privacy preservation in social networks is a relatively new research area, little work has been produced on the application of LDP to user profile data and activity logs. Fig 1 shows the motivational scenario of this research. The SN platform collects a pervasive amount of data and information and immediately stores it in its

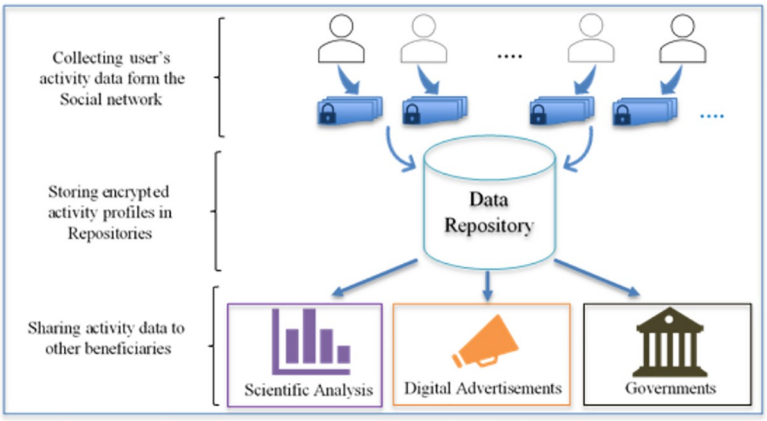

Fig 1. Data collection and sharing in an SN.

https://doi.org/10.1371/journal.pone.0215856.g001

repositories. The data and information are then shared with the governmental sector under certain agreements. The data may also be shared with analytical parties or even advertising companies to push specifically tailored digital advertisements.

Related work

Recently, the privacy of social network data has gained increasing attention and concern. Although these types of data are necessary for generating revenue and conducting social research, there is no guarantee that the implemented anonymization techniques will protect users' private information. In this section, we cover the state of the art application of local differential privacy (LDP) in social networks and other fields. LDP in social networks has become an alternative to simple graph anonymization and data aggregation. In one study, out-link privacy was implemented to protect information about individuals that is shared by other users. LDP has even been proposed to solve the non-uniformity problem in two-dimensional multimedia data [8]. Zhou et al. [9] claimed that calculating a standard deviation circle radius defines the divergence of a data grid and allows the dynamic allocation of noise. The results of their proposed model had lower relative errors than similar approaches, such as UG) algorithm. Kim et al. applied LDP to the collection of indoor positioning data and used differentiated data for estimating the density of a specified indoor area [10].

Recently, the application of LDP to crowdsourced data has received substantial attention [11–13]. In this context, LDP is mainly used to collect and build high dimensional data from distributed users [11]. These data are randomized using multi-variate joint distribution estimation on clusters of the dataset, and then the marginal distribution of these clusters is calculated to approximate a new dataset. When the model was tested, the Complexity Reduction Ratio (CRR) reached 0.512. In [12], an online aggregate monitoring framework was designed over infinite streams with a w-event privacy guarantee. The model was combined with a neural network to predict statistical values and test the utility of released data. The resulting mean absolute error (MAE) [0.2–16] and mean relative error (MRE) [0.2–0.6] indicate that the model improved the utility of real-time data publishing. The authors in [13] showed that LDP achieved an %89 assignment success rate in preserving the location of workers in Spatial Crowdsourcing (SC) through the random generation of 1000 work tasks from a dataset of 6100 users.

Privacy preservation in social networks is considered a relatively new research area. The work in [14–18] covers models generally dedicated to preserving privacy in social networks

(SNs). The model in [14] tackles the problems of determining data ownership in an SN and the vulnerability of SN metrics to changes in network structure. The authors claim it is necessary to develop an algorithm (such as minimal spanning tree, degree distribution, etc.) to compute results based on differential privacy (DP). Accordingly, modeling the complete 1-neighborhood structure as background knowledge was proven to protect privacy. The model focused on data that could be inferred from neighboring data and provided accurate answers to aggregate queries [15]. In addition to content, the correlation of a SN was investigated in [16]. The described algorithm labels vertexes in the dataset, uses dense clusters to populate an adjacency matrix, and applies a data partition to the matrix to identify dense regions. Lastly, DP is applied to obtain a noisy adjacency matrix. However, LDP has not always been preferred by researchers to preserve the privacy of sensitive attributes in SNs. Cai et al. and Backstrom et al. [17,18] suggest that, although LDP is generally suitable for inherent data, it is not the best choice for preventing inference attacks.

Furthermore, in [19], experiments showed that no LDP algorithm could fully preserve the persistent homology of high dimensional network features or fulfill all network graph metrics. Some proposed solutions for this issue include using Merging Barrels and Consistency Inference [20], deep neural networks with %73 accuracy [21] and neighborhood randomization [22]. An opposing opinion in [23] emphasizes the LDP's ethical and logistical capacity to protect organic data. The authors demonstrate that LDP can produce a differentially private synthetic dataset to be publicly distributed when combined with other privacy-protecting techniques, such as Ullman's Private Multiplicative Weights.

Local Differential Privacy Obfuscation (LDPO) is a variation of LDP tailored for IoT. LDPO substitutes homomorphic encryption to distill and aggregate data at edge servers with decreased computational overhead. The model is distributed over devices and both edge and cloud servers and provides an accuracy of %90.45 when using 30 features through feature distillation [24].

In summary, as privacy concerns are being raised ever more frequently, several local differential privacy models have been suggested and proven in many application areas for protecting user privacy from untrusted entities.

Preliminaries

Local differential privacy

LDP is a highly reliable and mathematically rigorous privacy standard [25, 26] that injects randomized noise into collected data or query results to hide sensitive details in a dataset. Thus, regardless of the experience level of an attacker, he/she cannot infer any knowledge from differentially elicited data [8, 27].

Definition (1): (ε−differential privacy)

Given two statistical datasets, D and $\dot{D}$, which satisfy $|D - \dot{D}| = 1$ (Hammingdistance), the randomized function A achieves ε−differential privacy on the condition that

$$e^{-\varepsilon} \leq \frac{\Pr(A(D) \in R)}{\Pr(A(\dot{D}) \in R)} \leq e^{\varepsilon}. \quad (1)$$

Definition (2) laplace mechanism

Using scientific data analysis, Dwork et al. [28] proposed the Laplace mechanism, which takes as inputs a database (or stream of data) D, function *f*, and privacy parameter ε (privacy budget) and returns the true output of *f* plus some Laplacian noise. This noise is drawn from a

Laplace distribution with the probability density function

$$P(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}. \tag{2}$$

where λ is determined by both GS(f) and the desired privacy level ε.

Theorem 1: For any function f: $\mathcal{D} \rightarrow \mathbb{R}^d$, the mechanism $\mathcal{A}$ for any dataset $D \in \mathcal{D}$,

$$\mathcal{A}(D) = f(D) + \langle Laplace\left(\frac{\Delta(f)}{\boldsymbol{\varepsilon}}\right)\rangle^d, \tag{3}$$

satisfies ε-differential privacy, where the noise, $\mathrm{Lap}\left(\frac{\Delta(f)}{\varepsilon}\right)$, is drawn from a Laplace distribution with a mean of zero and scale of Δ(f)/ε.

Probability Density Function (PDF)

A random variable has a {\displaystyle {\textrm {Laplace}}(\mu, b)}$Laplace(\mu Laplace(\mu,b)$ distribution if its probability density function is

$$f(x|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu}{b}\right). \tag{4}$$

Static / Uniform division

Involves the division of privacy level $\boldsymbol{\varepsilon}$ into smaller levels, $(\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \boldsymbol{\varepsilon}_3, \ldots\ldots, \boldsymbol{\varepsilon}_r)$, such that

$$\boldsymbol{\varepsilon}_1 = \boldsymbol{\varepsilon}_2 = \boldsymbol{\varepsilon}_3 = \ldots\ldots = \boldsymbol{\varepsilon}_r = \frac{\boldsymbol{\varepsilon}}{r}. \tag{5}$$

Dynamic / Adaptive division of privacy level

To divide the privacy level, a temporal scale must be introduced. Consider three successive SPs in ascending order: previous $(t_{s_{h-1}}, x_{s_{h-1}})$, current (t_{s_h}, x_{s_h}), and *next* $(t_{s_{h+1}}, x_{s_{h+1}})$. Given system parameter α, we calculate the temporal scale μ_h as [29, 30]

$$\mu_h = \left(\frac{|t_{s_h} - t_{s_{h-1}}| + |t_{s_h} - t_{s_{h+1}}|}{2}\right)^{\alpha}, \tag{6}$$

and, considering that privacy level $\boldsymbol{\varepsilon}$ is divided into $(\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \boldsymbol{\varepsilon}_3, \ldots\ldots, \boldsymbol{\varepsilon}_h)$

such that $\boldsymbol{\varepsilon} = \sum_{1 \le h \le r} \boldsymbol{\varepsilon}_h$ and $\mu_{sum} = \sum_{1 \le h \le r} \mu_h$, we calculate the individual small privacy level as

$$\boldsymbol{\varepsilon}_h = \boldsymbol{\varepsilon} \times \frac{\mu_h}{\mu_{sum}}. \tag{7}$$

Average error rate

The average error rate has been identified as follows.

Given the noisy salient points $S = (s'_1, s'_2, \ldots\ldots, s'_r)$ from r users [29], we can estimate the average values of the original x_n *at time* t_n, which requires averaging all the noisy values of x_n, as

$$AVG_{est}(x_n) = \frac{1}{r} \times \sum_{s'_i \in S} x'_n. \tag{8}$$

Conjugate Bayesian method [31–33]

The Bayesian probability function is given as

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}, \tag{9}$$

- where $P(c|x)$ is the posterior probability, and $P(c|x)$ is the likelihood;
- $P(c)$ is the class prior probability;
- $P(x)$ is the prediction prior probability.

A discrete random variable x is said to have a Poisson distribution with parameter $\lambda > 0$, if, for $k = 0,1,2,\ldots$, the probability mass function of X is given by

$$f(k;\lambda) = Pr(x = k) = \frac{\lambda^k e^{-\lambda}}{k!}. \tag{10}$$

For numerical stability, the Poisson probability mass function should be evaluated as

$$f(k;\lambda) = \exp\{k \ln \lambda - \lambda - \ln \Gamma(k+1)\}. \tag{11}$$

Choose P_{ij} = Poisson (λ_{ij}) for the unknown rate parameter $\lambda_{ij} > 0$.

Choose a gamma prior for λ_{ij} as this ensures that the posterior predictive distribution for a future period is calculable as a simple ratio of Poisson gamma mass functions.

Applying the Dirichlet process

Given a measurable set S, a base probability distribution H and a positive real number α, the Dirichlet process $DP(H,\alpha)$ is a stochastic process whose sample path is a probability distribution over S. For any measurable finite partition of $S : (B_i)_{i=1}^n$, If $X \sim DP(H,\alpha)$, we have

$$(X(B_1) \ldots\ldots X(B_n)) \sim Dir(\alpha H(B_1) \ldots.. H(B_n)). \tag{12}$$

The notation $X \sim DP(H,\alpha)$ indicates that the random variable X is distributed according to the distribution $DP(H,\alpha)$, i.e., according to a Dirichlet process with parameter base distribution H and real number α [31, 33].

The Dirichlet distribution of order $K \geq 2$ with parameters $\alpha_1,\ldots,\alpha_k > 0$ has a probability density function with respect to the Lebesgue measure on the Euclidean space $R^{k=1}$ given by

$$f(x_1 \ldots\ldots, x_k;\ \alpha_1, \ldots\ldots \alpha_k) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}. \tag{13}$$

Proposed approach

In this section, we describe the proposed scheme for sanitizing SN user activity logs using LDP. We then compare the results of applying anomaly detection to the original and reconstructed data. The model functions on two servers: a data collection server and a data-analyzing server. As shown in Fig 2, the data collection server represents each activity log as a data sequence. In each sequence, we determine specific salient points. After selecting these points, we use the user's data in addition to other parameters to create random noise. This noise is then added to the data to distort it from its original value. Finally, the data collection server stores it in data repositories.

In contrast, the data-analyzing server retrieves synthetic data from the repositories, reconstructs the original data streams, and searches the user's activity for abnormal behavior, as demonstrated in Fig 3.

The privacy standard model in the first sub-model avoids high error rates when applying LDP to large datasets. This model essentially groups salient points that represent similar actions (increasing, decreasing, constant) together, then applies LDP to selected points in these groups. Thus, a relatively small number of points are processed.

Users spend significant time on SNs performing all kinds of activities, such as sending messages, posting, liking posts, disliking posts, performing audio or video calling, and so on. If we consider a user's activity log per single action, it shows active periods vs. non-active periods. If we consider the sending and receiving of messages as an activity, the plot for a particular user's stream of data increases on days where a greater number of messages are sent and/or received, decreases on days where fewer messages are sent or received and remains constant on idle days.

The model operates in the following order:

Step 1: Calculate the salient points (SPs).

To obtain the representative points of a user's data sequence, we take the first order derivative of each value in his/her sequence at a specific timestamp. The user's sequence is represented as values collected at particular time intervals reflecting increasing, decreasing, or constant activity. In Fig 4, the user's calling activity is represented as a curve over a ten day time period.

Calculating the first order derivative allows us to determine increasing (derivative >0) and decreasing (derivative <0) periods. We exclude the constant periods where the user's activity value is the same. As explained in Algorithm 1, we store the points where the derivative of each value is not equal to zero.

```
Algorithm 1: Pseudo-code for calculating salient points
Input: S_i = ((t_1,x_1),(t_2,x_2),.....,(t_n,x_n)). // Activity stream for a
specific action performed by the i^th user.
Output: dS_i = ((t_1,dx_1),(t_2,dx_2),.....,(t_n,xd_n))
// Calculate first order derivative
n = size (S_i)
For i← 2: n
      dx_i ← (x_i − x_{i−1}) / (t_i − t_{i−1})
End For
// Exclude x_d = 0;
C_list ← NULL;
For i←1: n
      IF (dx_i ~ = 0)
            add dx_i to C_list;
      End IF
End For
```

Fig 5 shows the salient points calculated from the data stream of user calls in Fig 4 (when $dx_m \neq 0$). As shown in the Figure, the number of points can be further reduced while maintaining accurate data representation.

Step 2: Reduce the set of salient points to only those indicating the beginning of an increasing or decreasing period following the rule described in Table 1.

In situations where the data sequence is very long (thousands of values), or the data's time intervals are very small (seconds), the set of salient points will be considerably large and in need of further reduction. To achieve this, any successive points belonging to the same movement can be removed. Therefore, if three successive points belong to an increasing period, we merge their time interval, retaining only the beginning and end of the interval. We continue

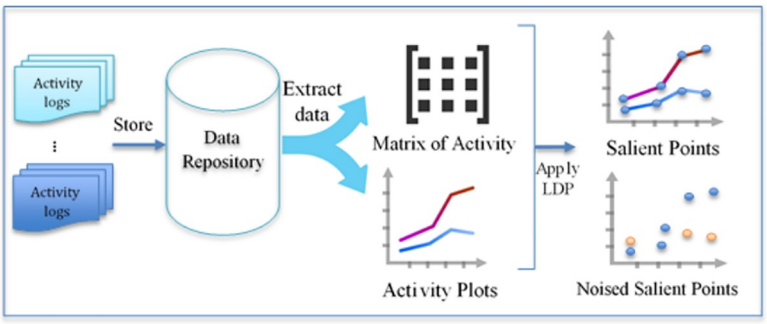

Fig 2. An overview of applying LDP to collected data.

https://doi.org/10.1371/journal.pone.0215856.g002

this reduction process until no two adjacent time intervals have the same movement. Algorithm 2 depicts these steps in detail.

```
Algorithm 2: Pseudo code for minimizing the number of salient points
Input: C_list; //First row in C_list is the derivative of the SP.
//The second row is the timestamp.
// Selecting points at the beginning of an ascending or descending
period
[~, n] ←size(C_list);
While (TRUE)
      interval_min ← ∞;
      For h ← 2: n-1
      // Obtaining the Pre-element
            Dx_pre ← C_list (1, h-1)
            T_pre ← C_list (2, h-1)
      // Obtaining the current element
            Dx_cur← C_list (1, h)
            T_cur← C_list (2, h)
      // Obtaining the next element
            Dx_next← C_list (1, h+1)
            T_next← C_list (2, h+1)
      //Applying the selection condition
            IF (Dx_pre>0 && Dx_cur>0 &&Dx_next >0) OR
                  (Dx_pre<0 && Dx_cur<0 &&Dx_next<0)
                  interval_cur ← |T_cur-T_pre|+|T_cur-T_next|;
                  IF (interval_cur < interval_min)
                  interval_min ← interval_cur;
                  t_min ← h
```

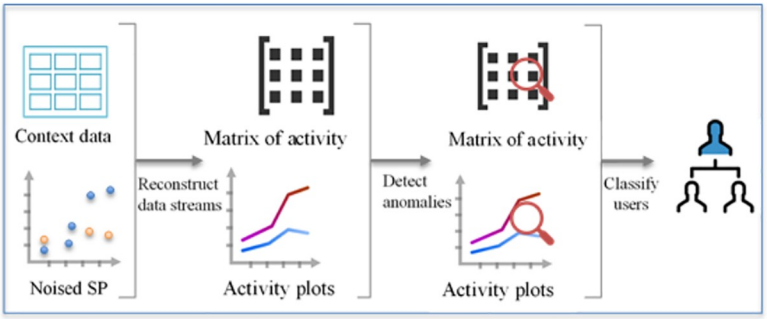

Fig 3. An overview of reconstructing the received data and classifying anomalous behavior.

https://doi.org/10.1371/journal.pone.0215856.g003

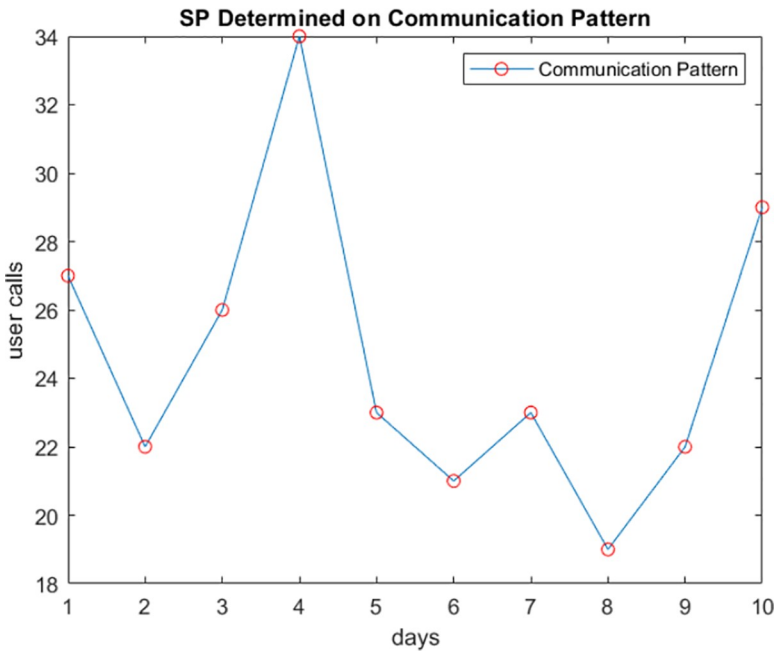

Fig 4. The calling activity of a user.

https://doi.org/10.1371/journal.pone.0215856.g004

```
        End IF
    End IF
End For
IF (interval_min ← ∞)
        break
End IF
```

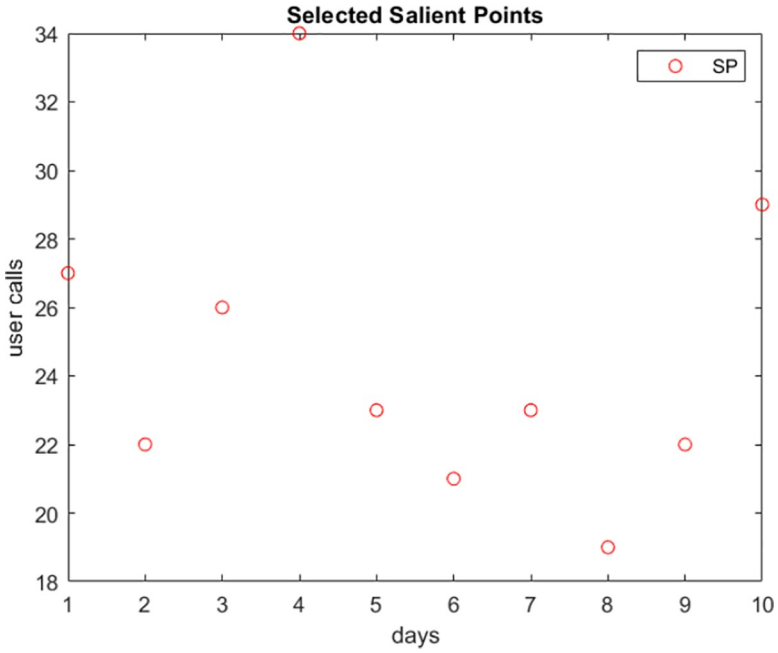

Fig 5. The representative salient points computed from the calling activity in Fig 4.

https://doi.org/10.1371/journal.pone.0215856.g005

Table 1. Rule for minimizing salient points.

Condition	Action
If three successive time intervals are all increasing or all decreasing	If the third point is a beginning of an opposite movement (increase after a decrease or decrease after an increase) 1. Add the first two together. 2. Remove the first two points from the list and keep the third.
	Else 1. Add them together. 2.Remove their corresponding points from the List of Points.
Otherwise	1. Keep the corresponding point in the list of points.

https://doi.org/10.1371/journal.pone.0215856.t001

```
      Remove element at C_list (1, t_min)
End while
```

Step 3: Calculate the privacy level in uniform and adaptive distributions.

Given the reduced set of salient points, we now partition the privacy level to generate random noise values. This step, uses Algorithm 3. The algorithm divides privacy ε into equal levels with each level ε_i satisfying the condition $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \ldots\ldots = \varepsilon_r = \frac{\varepsilon}{r}$.

```
Algorithm 3: Pseudo code for uniformly dividing the privacy level
Input: n // Length of the data sequence;
Epsilon // Parameter
Output: ε_i
// Uniform partitioning
For i←1: n
        ε_i← (Epsilon / n);
End For
```

In this step, we calculate a temporal scale for each salient point in the set that controls the privacy level of each point, thus regulating the amount of noise added to it. We use three timestamps representing the current, previous and next SP. We then calculate the temporal sum for all SPs in the sequence of a specific user. Then, we divide the privacy level based on this temporal sum. Algorithm 4 shows the steps of this procedure.

```
Algorithm 4: Pseudo code for the adaptive division of privacy level
Input:
Selected_SP // List of selected salient points
Epsilon // Parameter
Alpha // System parameter
Beta // Parameter
Output: Privacy level for each timestamp ε_i.
// Calculating Temporal scale μ_h = ((|t_{s_h} − t_{s_{h−1}}| + |t_{s_h} − t_{s_{h+1}}|)/2)^α
[m, n] ←size (Selected_SP);
For i←2: n-1
        Uniform_Up ←|Selected_SP(i)-Selected_SP(i-1) |
                +
                |Selected_SP (2, i)-Selected_SP (2, i+1) |
        Fraction ← Uniform_Up/2;
        Temporal_scale(i) ← (Fraction)^Alpha
End For
// The last element in the selected points does not have 'next'
        Uniform_Up ← |Selected_SP (2, n)-Selected_SP (2, n-1) |
        Fraction ← Uniform_Up/2;
        Temporal_scale(n) ← (Fraction)^Alpha;
// Calculating the Temporal sum μ_sum = ∑_{1≤h≤r} μ_h
temporal_sum←0
For i←1: n
```

```
        temporal_sum ← temporal_sum + Temporal_scale(i)
End For
// Calculating the privacy level ε_h = ε × μ_h/μ_sum
For i←1: n
    ε_i = Epsilon. (Temporal_scale(i)/temporal_sum);
End For
```

Step 4: Add Laplacian noise to the selected salient points.

If we consider the list of salient points $SP_i = ((t_{s_1}, x_{s_1}), (t_{s_2}, x_{s_2}), \ldots.., (t_{s_r}, x_{s_r}))$, we can obtain the noisy salient points $S^{'}P_i = ((t_{s_1}, \acute{x}_{s_1}), (t_{s_2}, \acute{x}_{s_2}), \ldots.., (t_{s_r}, \acute{x}_{s_r}))$, where $\acute{x}_{s_h}$ is obtained using the probability distribution function (PDF) of the Laplacian distribution

$$\acute{x}_{si} = x_{si} + Lap\left(\frac{\Delta s}{\varepsilon_i}\right). \tag{14}$$

The Laplacian generated noise depends on the privacy level. Therefore, using a uniform distribution generates noise different from adaptive noise. The higher the value of the privacy level, the higher the generated noise is. Therefore, it differs from one user to another. Knowing that the Laplace distribution performs a simple translation, it perfectly fits with the definition of differential privacy. The steps are shown in Algorithm 5. Since uniform privacy levels are the same, the same PDF generates the noise, whereas, in adaptive privacy partitioning, different PDFs are used to create the noise for each SP. Each different PDF incorporates a different privacy level due to dynamic partitioning, and the PDFs' are independently calculated for each SP.

```
Algorithm 5: Calculating Laplacian noise
Input:
Selected_SP [] // List of Selected Salient points of length n.
S_max // Maximum value of the data stream.
S_min // Minimum value of the data stream
Mean_Mu; // Mean variable for the PDF function.
Scale_b; // Standard Deviation for the PDF function
Uniform_privacy [] // List of uniform privacy levels for each SP.
Adaptive_privacy [] // List of adaptive privacy level for each SP.
Output:
Uniform_Noisy_stream // List of distorted SPs using a uniform privacy
distribution.
Adaptive_Noisy_stream // List of distorted SPs using an adaptive pri-
vacy distribution.
Delta_s = s_max-s_min; // Data sensitivity (low sensitivity causes
higher noise values)
[m, n]←size(Selected_SP);
For i←1: n
    Uniform_Up ← Delta_s/Uniform_privacy(i) // Delta over uniform
epsilon
    Adaptive_Up ← Delta_s/ Adaptive_privacy(i) // Delta over adaptive
epsilon
    Uniform_Noise ← pdf ('Normal', Uniform_Up, Mean_Mue, Scale_b)
    Adaptive_Noise ← pdf ('Normal', Adaptive_Up, Mean_Mue, Scale_b)
    Uniform_Noisy_stream (i) ← Selected_SP(i)+ Uniform_Noise
    Adaptive_Noisy_stream (i) ← Selected_SP(i)+ Adaptive_Noise
End For
```

Step 3: Store the 'noised' SP for analytical or other purposes. The repositories contain a sanitized representation of the SP with no indication of the original data.

Step 4: The requesting sub-model receives the noised SP and attempts to reconstruct the stream of data using linear estimation, as explained in the preliminaries section. The sub-model uses the linear equation of a straight line to draw segments between every two points. The general equation is

$$y = ax + b, \tag{15}$$

where a is the slope of the line, represented as the $(\frac{\text{change in the y-value}}{\text{change in the x value}})$. The y-intercept parameter b is the intersection point between the linear line and the y-axis, which is represented as

$$b = \acute{x}_{si} - a.t_{si}. \tag{16}$$

In our case, the slope is calculated as $\frac{\text{change in noisy points}}{\text{change in timestamps}}$. Algorithm 6 shows the code steps.

```
Algorithm 6: Reconstructing the original data stream using linear
estimation
Input: Uniform_Noisy_stream containing a sequence of the noised SP and
a sequence of timestamps for each noised SP.
Output: SP // List of the reconstructed SP.
[~, n] ← size (Uniform_Noisy_stream);
For i←1: n-1
       a(i) ← (Uniform_Noisy_stream(1,i+1) − Uniform_Noisy_stream(1,i)) / (Uniform_Noisy_stream(2,i+1) − Uniform_Noisy_stream(2,i))
       b(i) ← Uniform_Noisy_stream (1, i)- a(i). Uniform_Noisy_stream
(2, i);
End For
       a(n) ← Uniform_Noisy_stream (1, n)/Noisy_stream (2, n);
       b(n) ← Uniform_Noisy_stream (1, n)- a(n). Noisy_stream (2, n);
// Reconstructing the original SP
For i ← 1: n
SP(i) ← (a(i). Uniform_Noisy_stream (1, i)) + b(i);
End For
```

Step 5: For the activity dataset, the anomaly detection sub-model extracts the number of communications between pairs of nodes as a Bayesian counting process [31] and represents the number of interactions as weights assigned to communicating nodes in the network. The anomaly detection sub-model then applies Bernoulli, Markov chain and Dirichlet processes to find the nonparametric Bayesian inference.

Step 6: Perform individual-based analysis. In this step, we assume $N_{ij}(t)$ to be the adjacency of node i to node j at time t. The increments determine the out-degree and in-degree of node i, and we represent the number of outgoing nodes as

$$N_{i.}(t) = \sum_{j \neq i} N_{ij}(t), \tag{17}$$

while the incoming communications over time for individual i are represented as

$$N_{.i}(t) = \sum_{j \neq i} N_{ji}(t). \tag{18}$$

We then calculate the total activity by finding the degree sum of the network over time $N_{..}(t)$.

Step 6: A sample of size n is selected from the population. The random variable of interest, X, is the number of anomalous individuals in the sample, while M is the number of anomalous

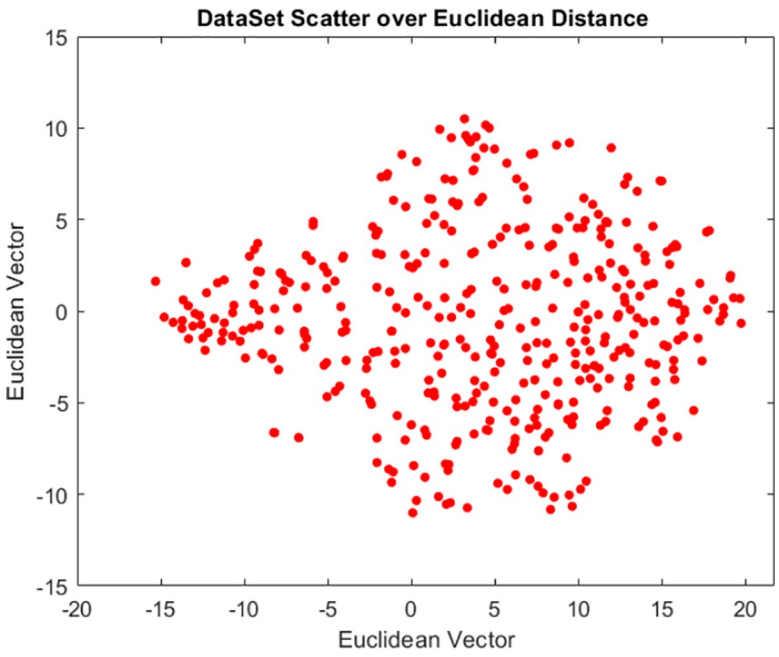

Fig 6. Dataset visualization using MATLAB's T-SNE function for dimension reduction over Euclidean distance metrics.

https://doi.org/10.1371/journal.pone.0215856.g006

individuals in the population, and N is the set of communicating individuals

$$P(X = x) = h(x; n, M, n) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}. \quad (19)$$

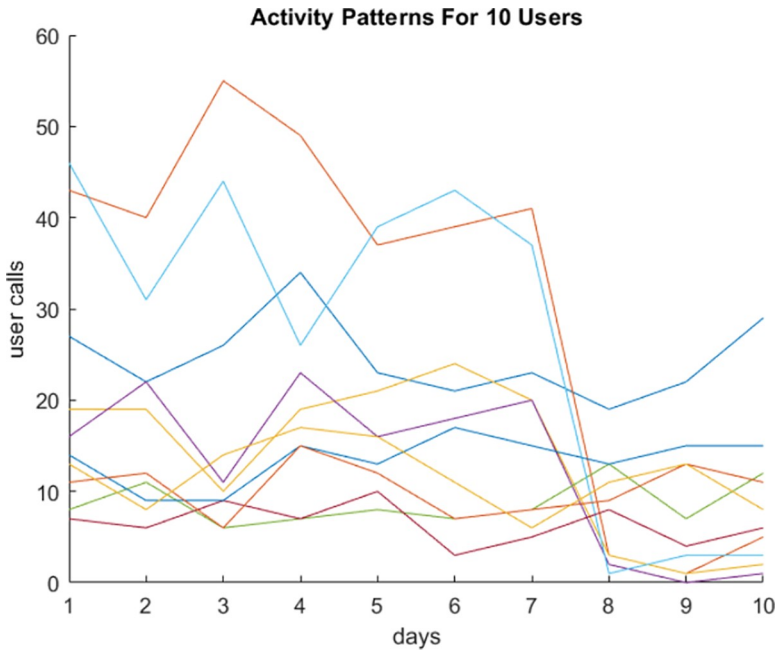

Fig 7. Plotting the activity of 10 users in the dataset.

https://doi.org/10.1371/journal.pone.0215856.g007

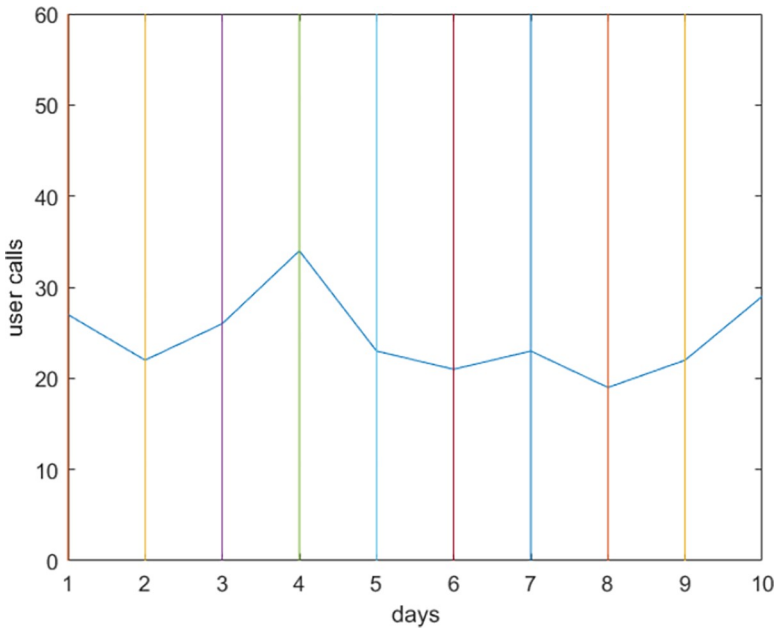

Fig 8. Selecting an SP to represent non-zero derivative periods. All SPs are selected on the curve.

https://doi.org/10.1371/journal.pone.0215856.g008

Experimental setup

In this section, we describe the simulation process, including the dataset, parameters, and evaluation metrics. We explain the setup and discuss the results in the second sub-section.

We conducted our experiments by applying the LDP privacy preservation distribution to a set of user activity sequences. The data sequence was adapted from the VAST Challenge 2008

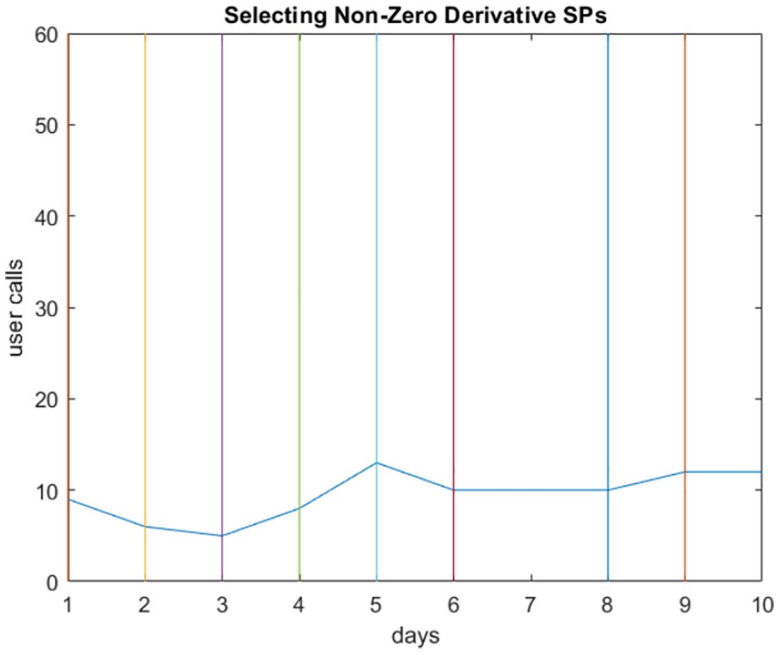

Fig 9. Selecting an SP to represent non-zero derivative periods. The SP for timestamp 7 was ($dx_i = 0$).

https://doi.org/10.1371/journal.pone.0215856.g009

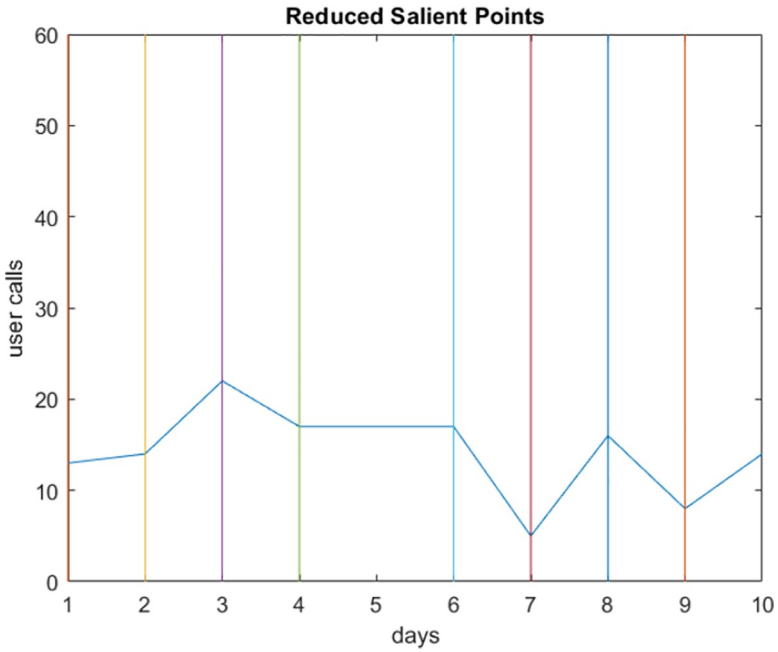

Fig 10. Reduced SPs for a user communication pattern.

https://doi.org/10.1371/journal.pone.0215856.g010

[32]. The LDP model was applied to simulated cell phone data from the Mini Challenge on social network analysis. Data were collected from 400 individuals located at 30 locations in the network over a period of ten days. Fig 6 shows a simple visualization of the data after projecting the high dimensional communication log into low-dimensional points.

Each data stream represents a user's calling activity over ten days. Each day represents a timestamp. Fig 7 illustrates the data sequences (streams) of ten users.

Results and discussion

We applied the steps explained in the proposed approach in Section 5. We first determined the salient points in each user's data stream. The user's activity in Fig 8 does not contain a constant period (having the same number of calls), so all points are selected. However, the user in Fig 9 makes the same number of calls on the sixth and seventh days ($dx_i = 0$). Since the first order derivative for timestamp 7 is zero, the salient point at this timestamp is removed. The same applies to timestamp 8; however, since it represents the beginning of a decreasing period, it is

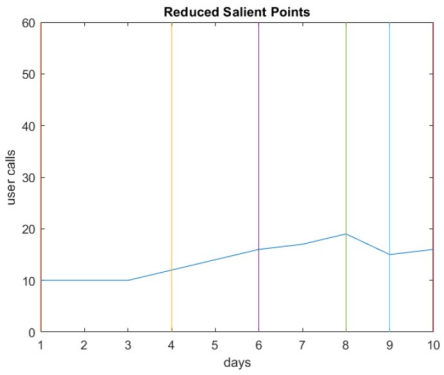

Fig 11. Another example of reduced SPs.

https://doi.org/10.1371/journal.pone.0215856.g011

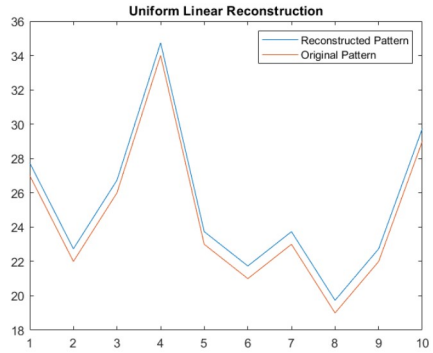

Fig 12. Plotting original and reconstructed curves in a uniform privacy distribution.

https://doi.org/10.1371/journal.pone.0215856.g012

retained. The colored lines parallel to the y-axis represent the timestamps, and the intersection point between each line and the curve is a salient point.

The next step is to reduce the salient points to represent points at the beginning of an increasing or decreasing period. Figs 10 and 11 show the reduced sets of two different users. The user in Fig 10 does not have consecutive intervals with all increasing or decreasing values, as the minimum interval is 2. The same scenario for the other user is shown in Fig 11.

After reducing the set of salient points, we calculate the individual privacy level for each point using uniform or adaptive division and assign Epsilon values of 2 and 5. The Epsilon value is used to generate the random noise applied to the reduced set. We use mean $\mu = 0.8$ *scale* $b = 0.2$ for the PDF function, and noise is added to each point in different data-set sizes: 50, 100, 200 and 400 users.

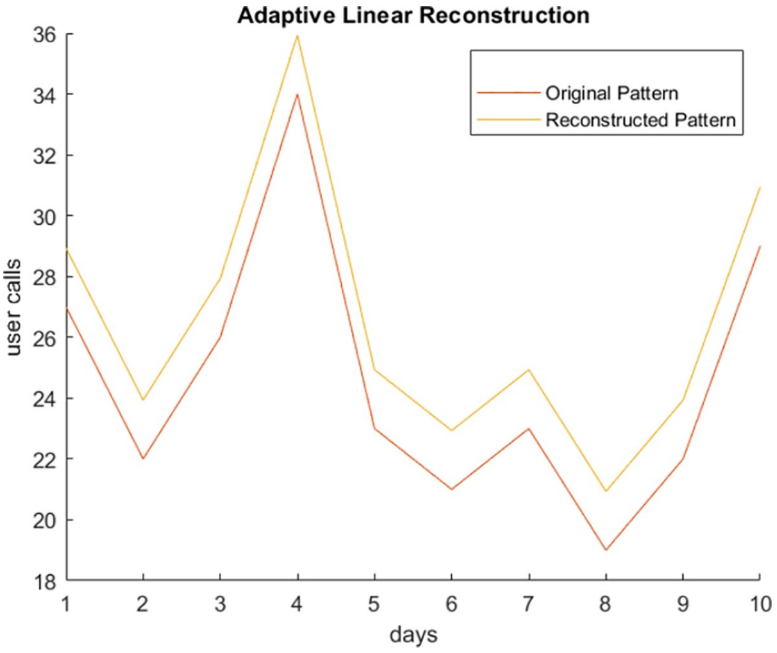

Fig 13. Plotting original and reconstructed curves in adaptive privacy distribution (low noise).

https://doi.org/10.1371/journal.pone.0215856.g013

Fig 14. Varying data size ($\varepsilon = 2$).

https://doi.org/10.1371/journal.pone.0215856.g014

Having created and stored the synthetic data on the data collection server, we next demonstrate the reconstruction process using linear estimation. Figs 12 and 13 show original and reconstructed data streams. In Fig 12, the red curve represents original user activity, and the blue curve represents the activity generated for uniform privacy levels. In Fig 13, the red curve represents the original data stream, and the yellow curve is the linear reconstruction for adaptive distributed privacy levels. Note that the reconstruction of data preserves the structure of the activity pattern, this is very important for anomaly detection.

We calculate the error rate for the combination of uniform-privacy division with linear estimation and adaptive-privacy division regenerated using linear estimation. Fig 14 plots the average error rate for the two different approaches on various data sizes. The error rate

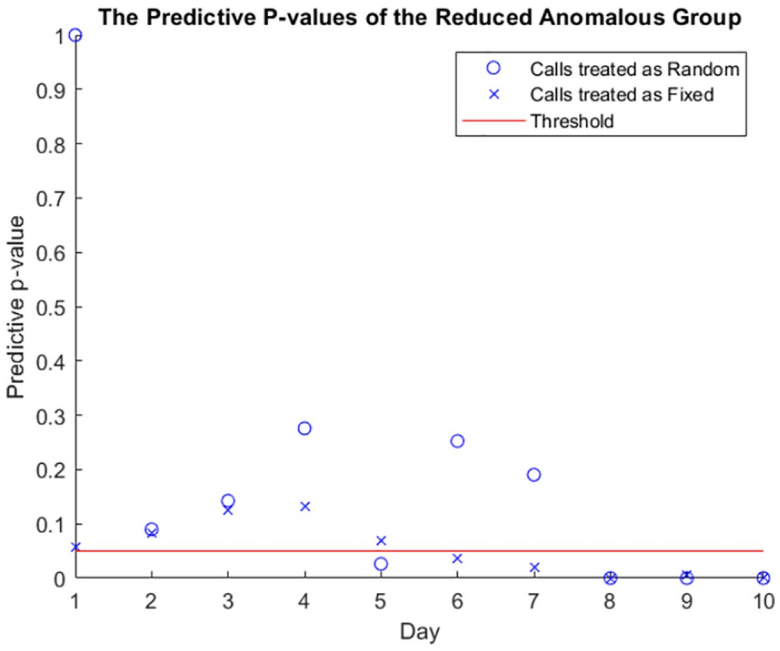

Fig 15. The p-values of the anomalous nodes under the multinomial model.

https://doi.org/10.1371/journal.pone.0215856.g015

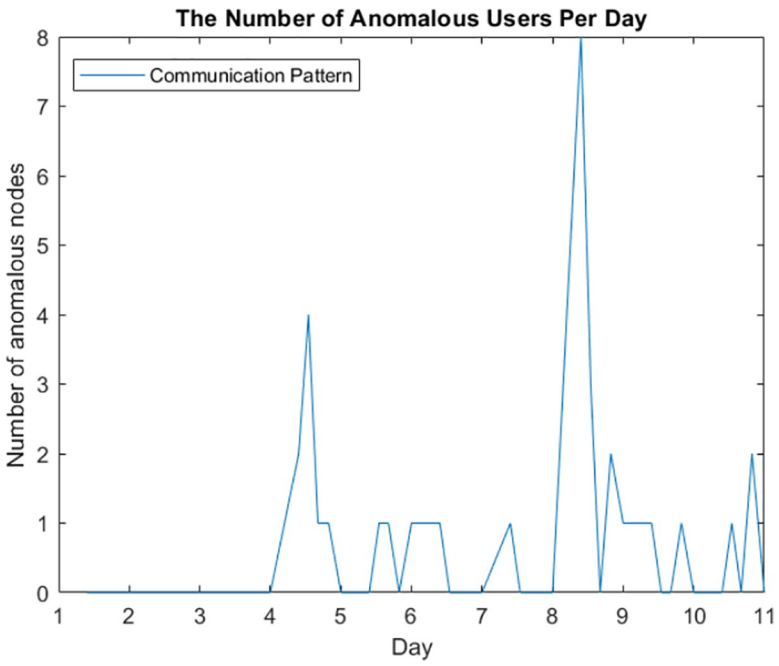

Fig 16. Number of anomalous nodes in each daily time interval under a Markov Bernoulli model.

https://doi.org/10.1371/journal.pone.0215856.g016

equation is

$$e = \frac{1}{seq\_length} \times \sum_{time_{stamp}=1}^{n} \frac{Avg(x_d) - Avg(\acute{x}_d)}{Avg(x_d)}. \quad (20)$$

where $Avg(x_d)$ is the average of the actual values in the data stream for timestamp d, and $Avg(\acute{x}_d)$ is the average of the reconstructed values of the data stream for the same timestamp.

We next apply the Bayesian anomaly detection technique to the reconstructed stream of users. In this experiment, we detect outliers with respect to the duration of calls between individuals. The duration variables are treated in the same manner as the calling activity described earlier. During the first analysis phase, the model checks all 30 locations for anomalous users to apply the multinomial model with the sequential Dirichlet process model with an uninformative negative binomial base measure [7].

We apply the Bernoulli process and Markov chain to all network users, with mean values of [0.63, 0.48] using a threshold of 0.05, to obtain a better understanding of the messaging patterns and their variability. This phase extracts the predictive P-values of the users from their communication patterns, as shown in Fig 15. The detection phase of the reconstructed data is the same as that of the original data. The same users have predictive p-values below the threshold and are flagged by the detection sub-model, which implies that the application of LDP to preserve data privacy succeeds in sanitizing the data. In addition, the data structure is maintained for further use by the anomaly detection sub-model.

In Fig 16, the abnormal activities peak on the eighth day, the same day the original activities peak, suggesting that the reconstructed data do not lower the performance of subsequent analyses, which can incorporate all the data into real-time anomaly detection.

As seen in the simulation results in Fig 16, the proposed model improves the estimation error while being applied to large-scale data. The model conducts anomaly detection on a

subset of the data without disclosing the actual values, which guarantees privacy and reduces the cost of further analyses.

Conclusion and future work

In this paper, we presented a model for privacy preservation in social networks. The model sanitizes the collected data and sensitive information of SN users using LDP and then attempts to reconstruct the original sequences and perform analyses using sets of selected salient points. We conserve the social structure of each user's communication pattern. The error rate of the estimated data compared to the original data is acceptable for large datasets with small time-intervals. Our simulation results show that conducting anomaly detection on synthetic data results in determining the same anomalous users and activities as those in the original data. In the future, we plan to extend the proposed privacy model to include estimating noisy data with non-linear approximation.

Author Contributions

Formal analysis: Randa Aljably, Abdullah Al-Dhelaan.

Funding acquisition: Mznah Al-Rodhaan, Abdullah Al-Dhelaan.

Investigation: Randa Aljably.

Methodology: Randa Aljably.

Resources: Randa Aljably.

Supervision: Yuan Tian, Mznah Al-Rodhaan.

Validation: Randa Aljably, Mznah Al-Rodhaan.

Visualization: Randa Aljably, Yuan Tian.

Writing – review & editing: Yuan Tian, Mznah Al-Rodhaan, Abdullah Al-Dhelaan.

References

1. Ayalon O, Toch E. Not even past: Information aging and temporal privacy in online social networks. Human–Computer Interaction. 2017 Mar 4; 32(2):73–102.
2. Cheng Y, Park J, Sandhu R. An access control model for online social networks using user-to-user relationships. IEEE transactions on dependable and secure computing. 2016 Jul 1; 13(4):424–36.
3. Adhikari K, Panda RK. Users' Information Privacy Concerns and Privacy Protection Behaviors in Social Networks. Journal of Global Marketing. 2018 Mar 15; 31(2):96–110.
4. Vishwanath A, Xu W, Ngoh Z. How people protect their privacy on Facebook: A cost-benefit view. Journal of the Association for Information Science and Technology. 2018 May; 69(5):700–9.
5. Phan N, Wu X, Hu H, Dou D. Adaptive Laplace mechanism: differential privacy preservation in deep learning. In Data Mining (ICDM), 2017 IEEE International Conference on 2017 Nov 18 (pp. 385–394). IEEE.
6. Dwork C, Naor M, Pitassi T, Rothblum GN. Differential privacy under continual observation. In Proceedings of the forty-second ACM symposium on Theory of computing 2010 Jun 5 (pp. 715–724). ACM.
7. Dabeer O. Joint Probability Mass Function Estimation from Asynchronous Samples. IEEE Transactions on Signal Processing. 2013 Jan 15; 61(2):355–64.
8. Task C, Clifton C. A guide to differential privacy theory in social network analysis. In Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012) 2012 Aug 26 (pp. 411–417). IEEE Computer Society.
9. Zhou G, Qin S, Zhou H, Cheng D. A differential privacy noise dynamic allocation algorithm for big multimedia data. Multimedia Tools and Applications. 2018:1–9.
10. Kim J, Kim DH, Jang B. Application of local differential privacy to collection of indoor positioning data. IEEE Access. 2018; 6:4276–86.

11. Ren X, Yu CM, Yu W, Yang S, Yang X, McCann JA, Philip SY. $\textsf {LoPub} $: High-Dimensional Crowdsourced Data Publication with Local Differential Privacy. IEEE Transactions on Information Forensics and Security. 2018 Sep; 13(9):2151–66.
12. Wang, Zhang Y, Lu X, Wang Z, Qin Z, Ren K. Real-time and spatiotemporal crowd-sourced social network data publishing with differential privacy. IEEE Transactions on Dependable and Secure Computing. 2018 Jul 1; 15(4):591–606.
13. Dai J Qiao K. A Privacy-Preserving Framework for Worker's Location in Spatial Crowdsourcing Based on Local Differential Privacy. Future Internet. 2018 Jun 14; 10(6):53.
14. Rajaei M, Haghjoo MS, Miyaneh EK. Ambiguity in social network data for presence, sensitive-attribute, degree and relationship privacy protection. PloS one. 2015 Jun 25; 10(6):e0130693. https://doi.org/10.1371/journal.pone.0130693 PMID: 26110762
15. Zhou, Pei J. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. Knowledge and Information Systems. 2011 Jul 1; 28(1):47–77.
16. Chen R Fung BC, Yu PS, Desai BC. Correlated network data publication via differential privacy. The VLDB Journal—The International Journal on Very Large Data Bases. 2014 Aug 1; 23(4):653–76.
17. Cai Z, He Z, Guan X, Li Y. Collective data-sanitization for preventing sensitive information inference attacks in social networks. IEEE Transactions on Dependable and Secure Computing. 2018 Jul 1; 15 (4):577–90.
18. Backstrm L, Dwork C, Kleinberg J. Wherefore art thou R3579x?: anonymized social networks, hidden patterns, and structural steganography. In Proceedings of the 16th international conference on World Wide Web 2007 May 8 (pp. 181–190). ACM.
19. Gao T, Li F. Preserving Graph Utility in Anonymized Social Networks? A Study on the Persistent Homology. In Mobile Ad Hoc and Sensor Systems (MASS), 2017 IEEE 14th International Conference on 2017 Oct 22 (pp. 348–352). IEEE.
20. Li X, ang J, Sun Z, Zhang J. Differential privacy for edge weights in social networks. Security and Communication Networks. 2017; 2017.
21. Abadi, Chu A, Good fellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security 2016 Oct 24 (pp. 308–318). ACM.
22. Fard M, Wang K. Neighborhood randomization for link privacy in social network analysis. World Wide Web. 2015 Jan 1; 18(1):9–32.
23. Garfinkle SL. Privacy and security concerns when social scientists work with administrative and operational data. The ANNALS of the American Academy of Political and Social Science. 2018 Jan; 675 (1):83–101.
24. Xu C, Ren J, Zhang D, Zhang Y. Distilling at the Edge: A Local Differential Privacy Obfuscation Framework for IoT Data Analytics. IEEE Communications Magazine. 2018 Aug; 56(8):20–5.
25. Shin, Kim S, Shin J, Xiao X. Privacy Enhanced Matrix Factorization for Recommendation with Local Differential Privacy. IEEE Transactions on Knowledge and Data Engineering. 2018 Feb 12.
26. Dwork C, Lei J. Differential privacy and robust statistics. In Proceedings of the forty-first annual ACM symposium on Theory of computing 2009 May 31 (pp. 371–380). ACM.
27. Horawalavithana S, Gandy C, Flores JA, Skvoretz J, Iamnitchi A. Diversity, Topology, and the Risk of Node Re-identification in Labeled Social Graphs. arXiv preprint arXiv:1808.10837. 2018 Aug 31.
28. Dwork C, Roth A. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science. 2014 Aug 11; 9(3–4):211–407.
29. Dwork C, Smith A. Differential privacy for statistics: What we know and what we want to learn. Journal of Privacy and Confidentiality. 2009 Jan 14; 1(2):135–54.
30. Kim J. W., Jang B., & Yoo H. (2018). Privacy-preserving aggregation of personal health data streams. PloS one, 13(11), e0207639. https://doi.org/10.1371/journal.pone.0207639 PMID: 30496200
31. Heard NA, Weston DJ, Platanioti K, Hand DJ. Bayesian anomaly detection methods for social networks. The Annals of Applied Statistics. 2010; 4(2):645–62.
32. Conn PB, Johnson DS, Williams PJ, Melin SR, Hooten MB. A guide to Bayesian model checking for ecologists. Ecological Monographs. 2018 Nov; 88(4):526–42.
33. Grinstein G, Plaisant C, Laskowski S, O'Connell T, Scholtz J, Whiting M. VAST 2008 Challenge: Introducing mini-challenges. InVisual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on 2008 Oct 19 (pp. 195–196). IEEE.