Neural Computing and Applications (2021) 33:12397–12427
https://doi.org/10.1007/s00521-021-05860-8

ORIGINAL ARTICLE

Privacy protection of online social network users, against attribute inference attacks, through the use of a set of exhaustive rules

Khondker Jahid Reza[1] · Md Zahidul Islam[1] · Vladimir Estivill-Castro[2]

Received: 8 January 2020 / Accepted: 19 February 2021 / Published online: 8 April 2021

Abstract
A malicious data miner can infer users' private information in online social networks (OSNs) by data mining the users' disclosed information. By exploring the public information about a *target* user (i.e. an individual or a group of OSN users whose privacy is under attack), attackers can prepare a training data set holding similar information about other users who openly disclosed their data. Using a machine learning classifier, the attacker can input released information about users under attack as non-class attributes and extract the private information as a class attribute. Some techniques offer some privacy protection against specific classifiers;, however, the provided privacy can be under threat if an attacker uses a different classifier (rather than the one used by the privacy protection techniques) to infer sensitive information. In reality, it is difficult to predict the classifiers involved in a privacy attack. In this study, we propose a privacy-preserving technique which first prepares a training data set in a similar way that an attacker can prepare and then takes an approach independent of the classifiers to extract patterns (or logic rules) from the training data set. Based on the extracted rule set, it then suggests the target users to hide some non-class attribute values and/or modify some friendship links for protecting their privacy. We apply our proposed technique on two OSN data sets containing users' attribute values and their friendship links. For evaluating the performance of the proposed technique, we use conventional classifiers such as Naïve Bayes, Support Vector Machine and Random Forest on the privacy-protected data sets. The experimental results show that our proposed technique outperforms the existing privacy-preserving algorithms in terms of securing privacy while maintaining the data utility.

Keywords Attribute inference attack · Data mining · Online social networks · Privacy preserving technique

1 Introduction

Online social networks (OSNs), also known as social attribute networks (*SAN*s) [1], can contain a wide variety of users' data such as "*Age*", "*Relationship Status*", "*Religious View*", and "*Political View*". Data mining of the users' information can be useful for identifying interaction patterns among the users [2], information credibility assessment [3], and other research purposes. However, it can also create a serious privacy breach to the OSN users by leaking a sensitive attribute or private information about the users [4]. Such data mining technology has reached the public eye through media reports (e.g. published by ABC News [5] and the Boston Globe [6]) because extremely delicate data such as a user's "*Sexual Orientation*" can be correctly predicted by analysing their disclosed nonsensitive information [7].

✉ Khondker Jahid Reza
kreza@csu.edu.au

Md Zahidul Islam
zislam@csu.edu.au

Vladimir Estivill-Castro
vladimir.estivill@upf.edu

1 School of Computing and Mathematics, Charles Sturt University, Panorama Avenue, Bathurst, NSW 2795, Australia

2 Departament de Tecnologies de la Informació i les Comunicacions, Universitat Pompeu Fabra, Roc Boronat, 138, 08018 Barcelona, Spain

The ability to infer sensitive information about a target user by analysing the disclosed nonsensitive information is usually known as an *attribute inference attack* [8, 9]. By "target user", we refer to an individual or a group of OSN users whose privacy is currently under threat. It is important to note that attackers can take any approach to launch this attack. We now illustrate a data mining approach to carry out this type of attacks.

1.1 The privacy attack model

The *attribute inference attack* [8, 9] profits from the use of supervised learning. Well before the emergence of deep learning, most data-mining deployments centred around classification tasks [10]. At present, conventional classifiers are widely used in various applications and specific industrial settings such as construction [11], flood management systems (FMS) [12, 13], wind and solar energy applications [14], biodiesel production [15], hydrogen production [16], and evaporation prediction [17]. Today, fields such as computer vision [18] and Internet of Things [19] are dominated by applications that build and deploy classifiers. A classifier is the output of supervised learning, and the input is called a training set. It is called learning as it can make prediction on unseen information based on the labels in the training set. We show here the framing of the attack as a supervised learning model.

Consider a *SAN* model $G(N, \mathcal{A}, \mathcal{E})$ containing a set of user nodes N (where $N = \{u_1, u_2, \ldots u_p\}$), a set of attribute nodes $\mathcal{A}$ ($\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_n\}$), and a set of links $\mathcal{E}$. In the *SAN* model, users and attribute values are considered as vertices of a graph.

A link between a user u_i and a node $\mathcal{A}_n = v$ represents the user u_i having the value v for the attribute $\mathcal{A}_n$. For example, if u_i discloses *Married* as his/her "*Relationship Status*", then a connection is placed between u_i and *Relationship Status* = *Married*. Therefore, u_i does not have any connection with other "*Relationship Status*" values. The *SAN* model also places an edge between two users if they are friends in the OSN. Figure 1 shows a sample *SAN* model. We first present our notation and its corresponding descriptions in Table 1.

Consider an attacker $\mathcal{Z}$ who wants to know the "*Political View*" of a user u_6 (shown in Fig. 1) who considers this information as sensitive and therefore did not disclose it on his/her OSN profile. Therefore, there is no link between u_6 and the values (i.e. *Labour* or *Liberal*) of the "*Political View*" attribute. In this example, the user u_6 is seen as a target user and for simplicity we use u_6 and u interchangeably from this point forward.

To launch the attack, $\mathcal{Z}$ can prepare a training data set D_{tr}. At first, $\mathcal{Z}$ may collect some information such as

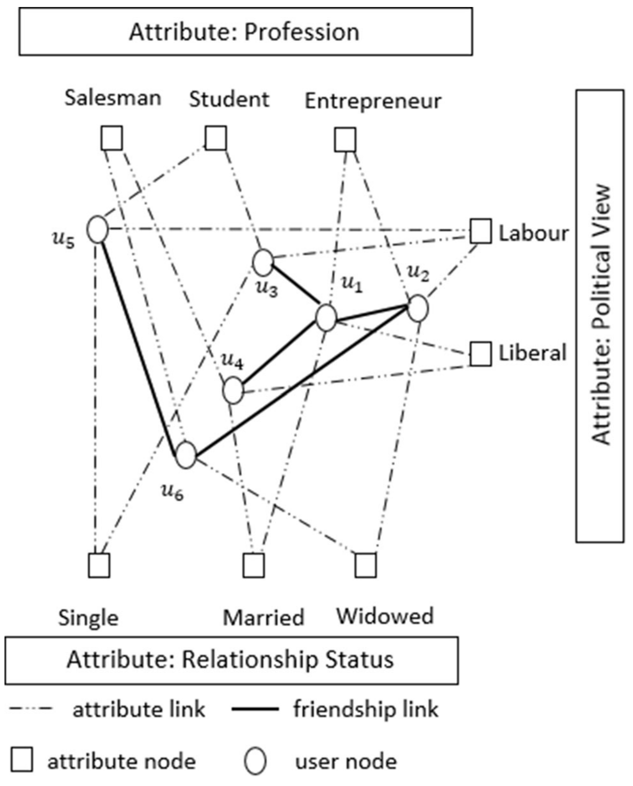


Fig. 1 A sample *SAN* model

"*Profession*" and "*Relationship Status*" about u_6 by exploring G (shown in Fig. 1). $\mathcal{Z}$ then can select a set of users who disclose their "*Political View*" information as well as the other information (i.e. "*Profession*" and "*Relationship Status*" in this example) that u_6 discloses. $\mathcal{Z}$ can also utilise the users' friendship information to launch the attack. $\mathcal{Z}$ may utilise metrics that evaluate principles of social influence that suggest users mimic those they are connected to [9, 20]. Also, the inverse also happens [21], and users with common attributes are likely to be connected [22, 23]. Equation (1) shows one such metric, the Adamic–Adar metric [24],

$$m(u_i, \mathcal{A}_n = v) = \sum_{t \in \Gamma_{s+}(u_i) \cap \Gamma_{s+}(\mathcal{A}_n = v)} \frac{w(t)}{log|\Gamma_+(t)|}. \qquad (1)$$

If a user assigns v as a value for $\mathcal{A}_n$, then we denote $\mathcal{A}_n = v$. $\Gamma_{s+}(u_i)$ is a set of user nodes in G connected to a user u_i and $\Gamma_{\mathcal{A}+}(u_i)$ is a set of neighbour attribute-value nodes connected to user u_i. For example, in Fig. 1, $\Gamma_{\mathcal{A}+}(u_1) = \{Entrepreneur, Married, Liberal\}$ and $\Gamma_{\mathcal{A}+}(u_2) = \{Entrepreneur, Widowed, Labour\}$.

On the other hand, $\Gamma_{s+}(\mathcal{A}_n = v)$ is the set of users in G who assign v as a value for the attribute $\mathcal{A}_n$. For instance, in Fig. 1, $\Gamma_{s+}(Profession = Entrepreneur) = \{u_1, u_2\}$. The degree of u_i can be denoted as $|\Gamma_+(u_i)|$ where

Table 1 Notations used in this paper

Notation	Description
G	A graph of an online social network
D	A data set
D_{tr} and D_{ts}	A training and test data set, respectively
N	Number of records in D
C	Set of sensitive attributes in D
c	Number of sensitive attributes in D i.e. $c = \|C\|$
C_o	oth sensitive attribute where $C_o \in C$
D_{tr,C_o}	A training data set where C_o is selected as a class attribute
D_{ts,C_o}	A test data set where C_o is selected as a class attribute
u	The *3LPEx* user
u_i	ith user in G
$\mathcal{A}$	Set of non-class attributes in D
$\mathcal{A}^r$ and $\mathcal{A}^l$	A set of *regular* and *link* attributes, respectively where $\mathcal{A}^r, \mathcal{A}^l \in \mathcal{A}$
$\mathcal{A}_n$	nth attribute
v	Value of $\mathcal{A}_n$ attribute
$\mathcal{Z}$	An attacker or a malicious data miner
b	Number of non-class attributes, i.e. $b = \|\mathcal{A}\|$
d	The domain size of an attribute
$\mathcal{L}$	Value of C_o
$\mathcal{R}$	Set of generated rules
$\mathcal{R}_a$	ath rule where $\mathcal{R}_a \in \mathcal{R}$
$N_{\mathcal{R}_a}$	Number of records that trigger the ath rule
$N^+_{\mathcal{R}_a}$	Number of records correctly classified by the ath rule
S_a	Sensitivity value of the ath rule
θ	User defined sensitivity threshold
R^u	Set of sensitive rules for user u
r	Number of sensitive rules i.e. $r = \|R^u\|$
t	A friend of ith user u_i
q	The degree of both number of friends and attribute values of each friend of u
$\mathcal{H}$	A $d \times b$ matrix
$\Gamma_{s+}(u_i)$	A set of user nodes in G connected to u_i
$\Gamma_{\mathcal{A}+}(u_i)$	A set of neighbour attribute-value nodes connected to u_i
$\Gamma_{s+}(\mathcal{A}_n = v)$	A set of users in G who assign v as a value for the attribute $\mathcal{A}_n$
$\|\Gamma_+(u_i)\|$	The degree of u_i
$P(C_o)$	The probability of class attribute C_o

$|\Gamma_+(u_i)| = |\Gamma_{s+}(u_i) \cup \Gamma_{\mathcal{A}+}(u_i)|$. For example, $|\Gamma_+(u_1)| = 6$ and $|\Gamma_+(u_2)| = 5$ in Fig. 1.

User t is a friend of u_i who has an attribute-value $\mathcal{A}_n = v$ and $t \in \Gamma_{s+}(u_i) \cap \Gamma_{s+}(\mathcal{A}_n = v)$. The weight $w(t)$ of the link between user u_i and user t is applicable when there is reason to believe some connections have a stronger meaning than another, for the purposes of this study, we model all link weights in G equally, setting $w(t) = 1$, for all t. The higher the metric value $m(u_i, \mathcal{A}_n = v)$, the higher the impact of v on u_i. The attacker $\mathcal{Z}$ can calculate $m(u_i, \mathcal{A}_n = v)$ as per Eq. (1) to incorporate the influence of friendship information into the data set D_{tr} (see Table 2).

After preparing D_{tr}, $\mathcal{Z}$ may select "*Political View*" as a class attribute and the rest of the attributes as non-class attributes. While selecting a user to include in the data set D_{tr}, $\mathcal{Z}$ follows a strategy of collecting users who have disclosed directly or indirectly their "*Political View*". For this example, we assume there are only two possible values: they are "*Liberal*" and "*Labour*". Our sample data set is illustrative of what could be prepared in a real-world scenario with an even larger set of attributes.

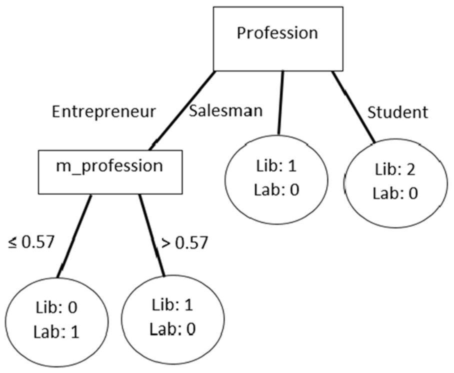
Table 2 A sample of training data set with friendship information

User	Relationship status	m_relation	Profession	m_profession	m_political	Class attribute
u_1	Married	0.72	Entrepreneur	0.62	0.72	Liberal
u_2	Widowed	0	Entrepreneur	0.56	0	Labour
u_3	Single	0	Student	0	0	Labour
u_4	Married	0.56	Salesman	0	0.56	Liberal
u_5	Single	0.56	Student	0.56	0.72	Labour

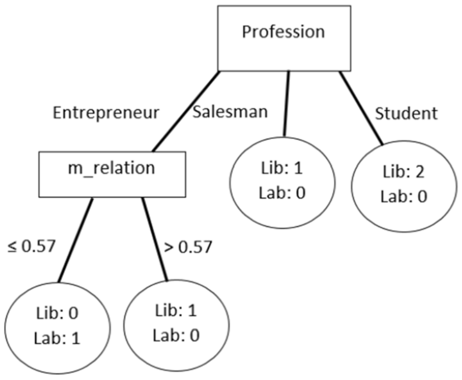
Table 2 presents such sample data set D_{tr} where rows are users' records and columns hold attributes. We name the attributes directly related to the users' information (such as "*Relationship Status*" and "*Profession*") as *regular attributes* and the attributes derived from metrics of friendship links (such as, *m_relation*, *m_profession*, and *m_political*) as *link attributes*. *Link* attributes are computed using Eq. (1).

$\mathcal{Z}$ can use a classifier (for example, a decision forest) to extract the rules from the data set D_{tr} and apply the learned classifier on the data of user u_6, (refer to Table 3), to breach his/her privacy. Figure 2 shows a sample of a decision forest generated from Table 2. In Fig. 2, the leaves appear as ovals while rectangular boxes are used for internal nodes. The path between a root node (the node at the top) and a leaf represents the logic rule for the leaf. For example, the logic rule for Leaf 1 (see Fig. 2a) states that "if the value of the attribute "*Profession*" is "*Entrepreneur*" and the value of the attribute "*m_profession*" is less than or equal to 0.57, then the class value is "*Labour*", and there is 1 record from D_{tr} that the model classifies correctly as "*Labour*" in this leaf. Here, the attributes "*Profession*" and "*m_profession*" are in the antecedent of the rule and the attribute "*Political View*" is in the consequent. *m_profession* of u_6 is 0.72 (see Table 3) which is the link attribute, for u_6 and Profession = Salesman, computed by Eq. (1).

By applying these learned rules on public data, on the public information about u_6, (refer to Table 3), $\mathcal{Z}$ can predict the "*Political View*" of user u_6. The possibility of this attack is experimentally described in previous studies as well [25, 26].



(a) Tree-1



(b) Tree-2

Fig. 2 Decision trees built on the sample data set given in Table 2. Here, "Lib" indicates "Liberal" and "Lab" indicates "Labour"

1.1.1 Privacy definition

For an OSN user u_i with an attribute value $\mathcal{L}$ of a sensitive attribute C_o, attribute inference may occur with a confidence γ when $P(C_o = \mathcal{L}) \geq \gamma$.

Table 3 A sample of test data set with friendship information

User	Relationship status	m_relation	Profession	m_profession	m_political	Class attribute
u_6	Widowed	0.72	Salesman	0.72	1.34	?

1.2 Related work

To try to eliminate the possibility of this privacy breach, a commercially available privacy preserving technique *NOYB* [27] uses a non-machine learning technique to guide what to obfuscate. *NOYB* takes a random suppression approach to select and suppress the *regular* attribute values, in the testing data set, that could be predictors of the class label. In an extreme scenario, *NOYB* may obfuscate all the *regular* attribute values for the sake of providing privacy.

Another technique, *TOTAL_COUNT*, and its variant *CUM_SENSITIVITY* [28], are based on calculating heuristics rank of the possibility of a *regular* attribute to be a strong predictor. The heuristics behind *TOTAL_COUNT* and *CUM_SENSITIVITY* are derived from logic rules obtained from the *SysFor* decision forest algorithm [29]. Based on the rules, it then ranks the attributes that appear most often in the rules with sensitivity larger than 1.006 and iteratively suggest the OSN user to suppress its value.

The *Sensitivity* of each rule is a function of the rule's *Support* and *Confidence* calculated as

$$S_j = \left(\alpha \times \frac{N_{\mathcal{R}_a}}{N}\right) + \left(\beta \times \frac{N^{+}_{\mathcal{R}_a}}{N_{\mathcal{R}_a}}\right). \tag{2}$$

Here, $N_{\mathcal{R}_a}$ refers to the number of records that trigger the ath rule, and $N^{+}_{\mathcal{R}_a}$ indicates the number of records correctly classified by the ath rule. The constants α and β are parameters of the method which the authors recommend to be set to 1 [28], which shows that the accuracy of the rule, relative to the number of users, is the main contributor to the sensitivity.

In our running example, the sensitivity of Rule 1 (refer to Fig. 2a) is calculated as: $S_1 = \left(1 \times \frac{1}{5}\right) + \left(1 \times \frac{1}{1}\right) = 1.2.$

Considering both *regular* and *link* attributes in D_{tr} suggests that the users shall add and hide friends from the friend list [30, 31] to reduce the possibility of a privacy breach. However, how to minimise the number of friends to hide or add or how to achieve an effective balance relative to the privacy level is left open.

A privacy protection technique based on Naïve Bayes classifier (*PrivNB* for short [32]) can provide privacy by suppressing attribute values and deleting friendship links. None of the experiments supporting the earlier techniques evaluates the effect of the method on the data utility of the protected data set. This would require a model of how each individual user interacts with the technique to achieve their satisfactory level of privacy, and the resulting data set global utility should be assessed after each of the interventions of each user.

Our earlier technique (*3LP*) [33] can provide privacy by suggesting three layers of action to protect the users' privacy. In Layer 1, it suggests to the user to suppress the values of *regular* attributes as ranked by the heuristics. However, if the user chooses not to use regular attributes or there are not any more regular attributes to achieve a satisfactory privacy level, in Layer 2, *3LP* suggests to the user to hide some friends from the friend list. If Layer 2 still does not offer a satisfactory level of privacy, in Layer 3, *3LP* suggests to the OSN users to add a few OSN users wisely in their friend list.

The *3LP* algorithm can protect the privacy of a sensitive attribute. But it would require re-application of *3LP* if there were more than one sensitive attribute.

We improved *3LP* to *3LP+* [34] to adopt a coordinated approach to protect the privacy of multiple sensitive attributes. This method uses a matrix to store the history of any modification of friendship information during a run to avoid a conflicting suggestion in a subsequent run.

The data utility is also an important element while protecting privacy. A user can protect their privacy by hiding all of their information, but it is an unwanted situation as the user wants to share some of their information with their friends. Therefore, the goal for a privacy-preserving technique should be a trade-off between privacy and utility of users' data.

The rest of this paper is organised as follows. Section 2 presents our proposed technique to protect privacy against the inference attack. We discuss our experimental set up in Sect. 3 and the results in Sect. 4. Finally, Sect. 5 gives the concluding remarks.

2 Our technique

2.1 Our contributions

3LP appeared in an Information and Security Conference [33] and *3LP+* in an Information Systems Security and Privacy Conference [34].

The multiple application of *3LP* would not use information about several attributes being targeted as sensitive. Therefore, such re-applications could be counterproductive by loosing a lot more information than is required to be lost. More importantly, the re-application can produce contradictory suggestions that if acted upon could re-introduce the risk of disclosure. For example, to protect the privacy of "*Political View*" *3LP* might suggest a user to hide a friend from his/her friend list while a subsequent run (say to protect the privacy of "*Religious View*") might suggest the user to disclose the same friendship information resulting in the loss of protection of "*Political View*".

This is mainly because multiple runs of *3LP* do not have proper co-ordination among the runs.

On the other hand, *3LP* and *3LP+* [34] can both suffer from a high *attack success rate*: if the attacker uses a different classifier (than the classifier to identify high predictor attributes in *3LP* or *3LP+*) to launch the attack. The term *attack success rate* refers to the probability of inferring users' sensitive attribute values correctly; the lower the *attack success rate* value, the higher the privacy.

In this paper, we present *3LPEx* that generates the sensitive rule set by considering all possible combinations of the attribute values in a data set. Therefore, even if an attacker uses a different classifier (i.e. not used by the protection technique) to generate logic rules, *3LPEx* should be able to provide protection against those rules from linking sensitive information. In addition, *3LPEx* can protect users' multiple sensitive information in one run while maintaining data utility. We present other four major contributions of this paper in Table 4 with their corresponding section numbers. Any of these contributions has never been published before.

2.2 Basic concept

A decision forest built from a training data set by an algorithm (e.g. Random Forest [35]) can be different from another decision forest produced by another algorithm (e.g. Bagging [36]) as different algorithms take different approaches to create the forests. So the logic rules discovered by a decision forest algorithm can be different from the logic rules discovered by another decision forest algorithm. If we provide privacy based on the logic rules discovered by a decision forest algorithm, we may not be able to secure a user if the attacker builds the decision forest using a different algorithm. Therefore, we need to first build a set of logic rules that covers/matches the logic rules obtained by any algorithm and then provide protection against this set of logic rules.

In this section, we experimentally demonstrate the dissimilarity of different logic rules by exploiting a set of decision forest algorithms such as *Random Forest* [35], *Random Subspace* [37], *SysFor* [29], *AdaBoost* [38], *Bagging* [36], *J48* [39], and *ForestPA* [40] that the attacker may utilise to compromise privacy. We apply these algorithms separately on an OSN data set [41], denoted as D_1, to extract a set of logic rules *TR*, and then follow the procedure described in Sect. 1.1 to prepare a set of sensitive rules *SR*. In the data set, we chose "*Political View*" as the class attribute (the details of the data set are presented in Sect. 3.1).

For each algorithm Al_g, we evaluate how good Al_g is to match the sensitive rules generated by another algorithm Al_k. Table 5 offers a row for each choice of algorithm Al_g and a column for each choice of algorithm Al_k. The number of rules generated by Al_g can be seen in the diagonal values of the table since the set of rules generated by an algorithm always matches with itself. However, the other values in a row are informative as follows. Consider the row for Al_g=*Random Forest* (RF), where 627 rules are generated by *Random Forest* as we can see in the 2^{nd} column of the 2^{nd} row in Table 5. We can also see that *Random Subspace* (RS) generates 12 rules (see the 4th column of Row 1). There is only one rule (out of 12 rules generated by *Random Subspace*) that is matched by one or more rules of the set of 627 rules generated by *Random Forest*. Similarly, there are only two rules (out of 402 rules generated by *AdaBoost*) that are matched by the set of 627 rules generated by *Random Forest*. Obviously, the more rules (generated by an algorithm) that are matched by the rules generated by *Random Forest*, the better privacy can be provided by using the rules generated by *Random Forest*.

A rule $\mathcal{R}_1$ is considered to be matched by a rule $\mathcal{R}_2$ if one of the following three conditions is met.

1. The antecedents of the rules completely match each other. By complete matching, we mean all the splitting attributes and their splitting points in the antecedent of $\mathcal{R}_1$ are exactly the same as the splitting attributes and splitting

Table 4 Major contributions of this study

Contribution number	Description of the contributions	Presented in section
1	We experimentally scrutinise the level of privacy protection of some existing privacy providing techniques if $\mathcal{Z}$ launches the attack using a different decision forest classifier rather than the one used by the technique	2.2
2	We propose a new algorithm *3LPEx* that provides privacy for multiple sensitive attributes	2.3, 2.4
3	A synthetic data generator is presented to generate OSN data sets for the experimental purposes	3.1
4	We empirically demonstrate the superiority of *3LPEx* over some existing techniques, even when $\mathcal{Z}$ uses a set of different classifiers to breach privacy	4.2
5	We also evaluate the data utility of *3LPEx* and compare with some existing privacy preserving techniques	4.3

Table 5 Sensitive rules matching

Row	Algorithms: SR (TR)	RF SR = 627	RS SR = 12	AdaBoost SR = 402	Bagging SR = 141	SysFor SR = 58	J48 SR = 2	FPA SR = 721
1	RF: SR = 627 (TR = 23,393)	627	1	2	7	18	0	17
2	RS: SR = 12 (TR = 2910)	5	12	0	7	0	0	8
3	AdaBoost: SR = 402 (TR = 20,562)	8	0	402	7	0	0	7
4	Bagging: SR = 141 (TR = 8914)	12	2	0	141	0	0	3
5	SysFor: SR = 58 (TR = 2666)	2	2	0	12	58	2	1
6	J48 - 1 tree: SR = 2 (TR = 298)	0	0	0	1	11	2	0
7	FPA: SR = 721 (TR = 38,453)	23	3	5	7	1	1	721
8	RuleBank - exclude the rules generated by the selected algorithm: SR = 1963 (TR = 97,196)	62	10	7	56	27	2	37
9	Exhaustive: SR = 65,218 (TR = 563,830)	**593**	**10**	**344**	**89**	**56**	**2**	**285**

* *TR* = total number of rules, *SR* = number of sensitive rules, *FPA* = forest PA, *RF* = random forest, *RS* = random subspace

points in the antecedent of $\mathcal{R}_2$ (even if the attributes are tested in different order in the tree, they represent the same antecedent as the logical conjunction and conjunction is a connective that commutes). For example, Rule 3 in Fig. 2a and Rule 7 in Fig. 2b are considered to be matched.

2. The antecedent of $\mathcal{R}_2$ is an refinement of the antecedent of $\mathcal{R}_1$. By the refinement of logic rules, we mean the antecedent of $\mathcal{R}_2$ has all the splitting attributes and points exactly the same as the antecedent of $\mathcal{R}_1$ plus some additional splitting attributes and points (the refinement will apply to a subset of the cases that satisfy the original rule; that is the refinement is more precise). For example, let us assume $\mathcal{R}_1$ is: *if Profession = Salesman* $\xrightarrow{then}$ *Lib: 1* and $\mathcal{R}_2$ is: *if Profession = Salesman & m_political* $> 0.31 \xrightarrow{then}$ *Lib: 1*. In this example, $\mathcal{R}_2$ is an refinement of $\mathcal{R}_1$. $\mathcal{R}_2$ represents records those are a subset of the records represented by $\mathcal{R}_1$. Therefore, if the system were to suggest the obfuscation of attributes in the antecedent of $\mathcal{R}_1$, records that are classified by $\mathcal{R}_2$ are also protected. This is why we consider $\mathcal{R}_2$ is matched by $\mathcal{R}_1$ for our purpose (in a sense this is analogous to the anti-monotone property of support in the analysis of frequent item sets by the apriori algorithm).

3. Either of the above two conditions is met except that the splitting point of a numerical splitting attribute varies within 10% of the domain range of the attribute. For example, if the domain of the attribute m_profession = [0,1], $\mathcal{R}_1$ is: *if Profession = Entrepreneur & m_profession* $> 0.57 \xrightarrow{then}$ *Lib: 1* and $\mathcal{R}_2$ is: *if Profession = Entrepreneur & m_profession* $> 0.54 \xrightarrow{then}$ *Lib: 1* then we consider $\mathcal{R}_2$ is matched by $\mathcal{R}_1$.

Because of the difference of what is found by one method among the rules produced by another, we can take the following approach. We name *RuleBank* (Al_k) the union of all the rules generated by all algorithms except Al_k, and we check how well all other algorithms do in terms of covering Al_k. The results of this "all others versus one left out" appear as the row for *RuleBank*. Column 3 of Row 8 suggests that the *RuleBank* combining all algorithms except Random Forest can only match 62 rules (out of 627 rules) generated by Random Forest. We see that even if we use all other algorithms, we only match a small fraction of the rules generated by a single algorithm as long as that one is not included in the *RuleBank*. This lack of power of an assemble of algorithms emphasises that for maximising privacy protection one algorithm to choose sensitive rules is clearly insufficient.

We also consider 5% and 15% difference of the domain range of the attribute but an insignificant difference (from the 10% of the domain range results) has been observed (see "Appendix 1").

Algorithm 1 GetExhaustiveRuleset $(D,\ D_{tr},\ \mathcal{A},\ C_o,\ u)$

```
Input   : $D$, $D_{tr}$, $\mathcal{A}$, $C_o$ and $u$.
Output  : a set of sensitive rules $R^u$ for $u$.
Step 1: Store the attribute values in $\mathcal{H}$.
    Initially $\mathcal{H}$ is set to null.
    foreach $\mathcal{A}_n \in \mathcal{A}$ do
        A set $T$ is set to empty.
        if $\mathcal{A}_n$ is categorical then
            Extract all the domain values of $\mathcal{A}_n$ from $D$ and store them in $T$.
            for $j = 1$ to $|T|$ do
                $\mathcal{H}_{j,n} = T_j$ /* Store the $j^{th}$ value of $T$ in $\mathcal{H}$ */
        if $\mathcal{A}_n$ is numerical then
            Calculate the gain ratio for each split point using $D$ and store the top 5 gain ratio in a set $g$ and their corresponding split points in $T$.
            for $j$=1 to $|T|$ do
                $\mathcal{H}_{j,n} = T_j$ /* Store the $j^{th}$ split point in $\mathcal{H}$ */
Step 2: Generate a set of sensitive rules for the 3LPEX user.
    Initially $\mathcal{R}$ and $R^u$ are set to null.
    for $i$=1 to $(|\mathcal{A}| - 2)$ do
        for $x$=1 to $|\mathcal{A}_i|$ do
            for $j$=$i$+1 to $(|\mathcal{A}| - 1)$ do
                for $y$=1 to $|\mathcal{A}_j|$ do
                    for $k$=$j$+1 to $|\mathcal{A}|$ do
                        for $z$=1 to $|\mathcal{A}_k|$ do
                            $\mathcal{R}_a = BuildRule(\mathcal{H}_{x,i}, \mathcal{H}_{y,j}, \mathcal{H}_{z,k},\ D_{tr}, C_o)$
                            $\mathcal{R} = \mathcal{R} \cup \mathcal{R}_a$
    foreach $\mathcal{R}_a \in \mathcal{R}$ do
        Calculate sensitivity, $S_a$, of $\mathcal{R}_a$ using Equation 2.
        if $\mathcal{R}_a$ is applicable on $u$ and $S_a \geq \theta$ then
            $R^u = R^u \cup \mathcal{R}_a$
            $\mathcal{R} = \mathcal{R} \setminus \mathcal{R}_a$
    return $R^u$;
```

2.3 Exhaustive approach

In this *Exhaustive* approach, a set of logic rules is created by considering all possible combinations of the non-class attribute values (see Algorithm 1). The entire approach is completed in two steps. In Step 1, the attribute values of each non-class attribute (obtained from the training data set D_{tr}) are stored in a matrix $\mathcal{H}$. In Step 2, a set of sensitive rules is generated by utilising the $\mathcal{H}$. We describe each step as follows.

Step 1: Store the attribute values in $\mathcal{H}$.

As an input, the *Exhaustive* approach (see Step 1 of Algorithm 1) first takes the main data set D, the training data set D_{tr}, the set of non-class attributes $\mathcal{A}$, the class attribute C_o, and the user u. A non-class attribute $\mathcal{A}_n$ can be *categorical* or *numerical*. If $\mathcal{A}_n$ is *categorical* then it stores all the domain values of $\mathcal{A}_n$ in a matrix $\mathcal{H}$ (shown in Step 1 of Algorithm 1).

On the other hand, if $\mathcal{A}_n$ is *numerical*, then it calculates the gain ratios for the split points and selects the top five gain ratios to store their corresponding splitting points in the matrix $\mathcal{H}$. At the end of Step 1, matrix $\mathcal{H}$ is prepared and made ready for use by Step 2.

Step 2: Generate a set of sensitive rules for the 3LPEx ***user.***

In Step 2, a set of exhaustive rules is first generated by considering all possible combinations of the attribute values stored in $\mathcal{H}$. It is theoretically impossible to protect all inferences when data on users are available. One cannot release any data, but then OSN would not be interesting to users. It is also impossible to anticipate what machine learning tools would the attacker apply attempting to compromise the privacy of OSN users. Thus, the first alternative is what has been attempted before (see Sect. 1.2): to anticipate what are the attributes that have high predictive power by observing how frequently they appear in the rules generated by some machine learning algorithms. Here, for the first time, we inaugurate the approach of being exhaustive and aim at inspecting all rules regardless of how could they be generated (and in fact most of these rules would have no predictive power). The approach here surpasses previous approaches by being agnostic to the identification of predictive attributes (while all other machine learning algorithms introduce some learning bias as they apply some induction principle).

Once the exhaustive rules are generated, sensitivity of each rule is calculated by using Eq. (2) and the rules having sensitivity value over a threshold are considered as sensitive rules. Our earlier study [42] demonstrated experimentally that a sensitivity threshold lower than 1.006 can provide better privacy by suppressing more *regular* attribute values. Therefore, following earlier studies [28, 33, 34], in this study we also consider the sensitivity threshold to be 1.006. At the end of Step 2, a set of sensitive rules is generated for the use by Algorithm 2.

While generating the set of exhaustive rules, we consider the number of attributes tested in the antecedent of an exhaustive rule to be bound at 3 (see Algorithm 1). That is, we exhaustively generate all classification rules whose antecedents have at most three terms/conditions in an antecedent. Our algorithm does not require this number to be 3 (i.e. our algorithm is not restricted to using 3 as a bound on the size of the antecedent). However, as we will see later, the algorithms complexity will be exponential on this parameter. One can use a larger number of attributes in the antecedent of a rule, provide more privacy protection, but would consume more computational resources. As we explained before, guaranteeing privacy protection to all attacks would require either complete obfuscation of all data (which would destroy all utility of the OSN) or exhaustive generation of all rules of all types (not just decision tree type with logic rules but oblique and regression rules such as in CART and all forms and mappings as hyperspace mappings by support vector machines, etc.).

We will show with our experiments that the value 3 seems a good choice to highlight and made recommendations to users to obfuscate the least number of nonsensitive attributes and minimise the need to fabricate friendships or obfuscate genuine ones. Because of the anti-monotone property mentioned earlier, by recommending obfuscating attributes in rules with antecedent with capped size 3, any refinement that uses any of these attributes becomes ineffective to disclose confidential attributes. However, the possibility exists for larger antecedents which results in extremely peculiar and precise rules that still jeopardise the privacy of some users. Our experiments will also show that at bound 3, we do not achieve total privacy protection. But certainly much more protection than any previous method.

We now give an example of exhaustive rules with two attributes in the antecedents and a class value. If two non-class attributes being tested in the antecedents are $\mathcal{A}_1$ and $\mathcal{A}_2$ with domain values $\mathcal{A}_1 = \{v_{11}, v_{12}\}$ and $\mathcal{A}_2 = \{v_{21}, v_{22}\}$, and the class attribute is C_o with domain values $C_o = \{C_{o1}, C_{o2}\}$, then the set of exhaustive rules is as follows.

Rule 1: If $\mathcal{A}_1 = v_{11}$ and $\mathcal{A}_2 = v_{21} \xrightarrow{then} C_o$=either C_{o1} or C_{o2}.

Note that the actual value of C_o will be determined by the majority class value of the records satisfying the antecedent of the rule.

Rule 2: If $\mathcal{A}_1 = v_{11}$ and $\mathcal{A}_2 = v_{22} \xrightarrow{then} C_o$=either C_{o1} or C_{o2}.

Rule 3: If $\mathcal{A}_1 = v_{12}$ and $\mathcal{A}_2 = v_{21} \xrightarrow{then} C_o$=either C_{o1} or C_{o2}.

Rule 4: If $\mathcal{A}_1 = a_{12}$ and $\mathcal{A}_2 = v_{22} \xrightarrow{then} C_o$=either C_{o1} or C_{o2}.

The attributes tested in the antecedent of an exhaustive rule can be either *categorical* or *numerical* or a combination of both *categorical* and *numerical*. If an attribute is *numerical*, then the split points stored in $\mathcal{H}$ are used one by one in an exhaustive fashion. The effectiveness of the sensitive rules obtained through the *Exhaustive* approach in preserving privacy is also demonstrated in Row 9 of Table 5. We can see that sensitive rules obtained this way match with a high number of rules obtained by all other algorithms. Therefore, if we protect the privacy of users based on those exhaustive rules that highlight which attributes are high predictors, then we can ensure higher privacy than the privacy provided based on the sensitive rules obtained by any other approach that focuses on a single algorithm (including *RuleBank*). This is illustrated across Row 1 to Row 8 of Table 5.

2.4 Main steps of the *3LPEx*

We now introduce our proposed privacy protection algorithm, namely *3LPEx*, Three Layers of Protection using an Exhaustive set of rules. In the first layer, *3LPEx* suggests its users to hide some non-sensitive *regular* attribute values that can be high predictor of a sensitive attribute value. After the first layer, if any sensitive rule remains that can reveal the users' sensitive attribute value, then *3LPEx* suggests its users to hide some friendship links in the second layer so that the metric value of a *link* attribute becomes lower than mentioned in the sensitive rules. After the first and second layer, if there are still some sensitive rules that may disclose the users' sensitive information, only then *3LPEx* uses third layer. In the third layer, *3LPEx* suggests its users to add some new friends on their profiles so that the metric value of a *link* attribute becomes greater than mentioned in the sensitive rules. We now discuss the main steps of *3LPEx* by assuming a *target user u* who uses the *3LPEx* to protect his/her sensitive attributes.

3LPEx takes, as an input, the main data set D, the friendship network G, and a set of non-class attributes $\mathcal{A}$. Here $\mathcal{A}$ contains both *regular* attributes $\mathcal{A}^r$ and *link* attributes $\mathcal{A}^l$. *3LPEx* also takes a set C of attributes, that a user u considers to be sensitive. For instance, if u considers "*Political View*" and "*Religious View*" to be sensitive attributes, then $C = \{$*Political View*, *Religious View*$\}$.

3LPEx randomly chooses a sensitive attribute in C, and one by one, *3LPEx* aims to provide the privacy of the sensitive attribute by using its 3 layers/steps of privacy protection. In Layer 1 (i.e. Step 1), it suggests u to suppress some attribute values, in Layer 2 it suggests to hide some friendship links and in Layer 3 it suggests to add some new friends. Note that the layers are applied sequentially, and a later layer (e.g. Layer 2) is used, only if the privacy of the sensitive attribute is not protected by using an earlier layer. Once the privacy of the current sensitive attribute is protected (at the end of an iteration of *3LPEx*), the sensitive attribute is removed from C. Once the privacy of all sensitive attributes is protected, the set C becomes empty and *3LPEx* prompts a user to provide more sensitive attributes, if any (see Algorithm 2).

In each iteration, *3LPEx* protects the privacy for a sensitive attribute. Hence, in this example where $C = \{$ *Political View*, *Religious View*$\}$, *3LPEx* iterates twice. The first iteration protects the attribute-value pair "*Political View*" and later, the pair "*Religious View*".

An iteration begins when a user u agrees to protect his/her sensitive attribute by giving a flag information (i.e. $Continue = True$) and the entire process will not stop until u wishes to do so (i.e. $Continue = False$).

Step 1: Compute Sensitivity of Each Attribute for a User and Suggest the User to Suppress Regular Attribute Values as Necessary.(Layer 1)

In Step 1 (see Algorithm 2), the *3LPEx* first selects a sensitive attribute C_o (where $C_o \in C$) as a class attribute and prepares a training data set D_{tr}. In this example, C_o can be either "*Political View*" or "*Religious View*".

3LPEx then takes the *Exhaustive* approach (by applying Algorithm 1) to generate a set of sensitive logic rules, denoted by R^u, for u. We represent the sensitive rules generating process by using a function *GetExhaustiveRuleset* $(D, D_{tr}, \mathcal{A}, C_o, u)$.

In R^u, a non-class attribute $\mathcal{A}_n$ can appear multiple times. Therefore, *3LPEx* counts the total number of appearances of $\mathcal{A}_n$ in R^u. After counting the number of appearances of $\mathcal{A}_n$, it stores the number in the nth index of a set *Appear*. Similarly, *3LPEx* counts the number of appearances of all other non-class attributes (in D_{tr}) and stores the information in *Appear*.

Once the counting is completed, *3LPEx* selects the *regular* attribute $\mathcal{A}_n$ (i.e. $\mathcal{A}_n \in \mathcal{A}^r$) with the highest number of appearances in *Appear* and recommends u to suppress the value of $\mathcal{A}_n$.

If u accepts the recommendation of suppressing the value of $\mathcal{A}_n$, the set of sensitive rules in R^u containing $\mathcal{A}_n$ in their antecedents are first added into a set T and then removed from R^u. T is removed from R^u because T will no longer be useful in breaching the privacy (i.e. predicting the value) of the sensitive attribute due to the suppression of $\mathcal{A}_n$. At this stage, *3LPEx* sets $Appear_n$ (i.e. the nth of the set, *Appear*) to zero.

After the removal of T, if R^u is still not empty, then *3LPEx* identifies another *regular* attribute $\mathcal{A}_n$ (with a different value for n) with the next highest number of appearances. This suppression process continues as long as R^u is not empty and R^u contains at least a rule containing a *regular* attribute in its antecedent.

Step 2: Suggest the User to Hide Friendship Links as Necessary if they are not fabricated previously. (Layer 2)

After Step 1, if a rule remains in R^u that predicts the sensitive attribute value, then such rule can only contain *link* attributes from the set $\mathcal{A}^l$ in its antecedent. Unlike the *regular* attributes, the *link* attributes' values cannot be suppressed (as they can be computed based on Eq. (1)) and can only be modified by hiding or adding some OSN users in the friend list of u.

If a user u satisfies the conditions of a rule R^u_j the antecedent of which requires the value of a link attribute A^l to be $\geq$ a splitting point *SplitPoint*, then *3LPEx* aims to change the value of A^l for u to be $<$*SplitPoint* by hiding some suitable friends in order to get u not satisfying the R^u_j so that an adversary cannot use R^u_j to breach the privacy of the sensitive attribute for u. In this step (i.e. Layer 2), *3LPEx* only hides suitable friends to remove u from as many rules in R^u as possible.

Algorithm 2 The Main Steps of *3LPEx*

```
Input     : user $u$, main data set $D$, friendship network $G$, non-class attributes $\mathcal{A}$ where $\mathcal{A}^r, \mathcal{A}^l \subset \mathcal{A}$, and a set of sensitive attributes $C$.
Output    : Recommend $u$ to suppress some regular attribute values and/or modify some friendship links.
Variables : $F = 1 \times N$ matrix stores Flag information for $u$ /*Initially $F$ is set to $False$ */; $Appear$ = A set $Appear$ stores the number
            of appearances of each attribute in $R^u$.

Initialise $Continue = True$, $G' = G$, and $F' = F$.
while $Continue = True$ do
    if $|C| = \phi$ then
        Prompt and wait until $u$ provides the list of sensitive attributes.
    else
        Step 1: Compute Sensitivity of Each Attribute for a User and Suggest the User to Suppress Regular Attribute Values
        as Necessary. (Layer 1)
            Select a sensitive attribute $C_o$ (where $C_o \in C$) randomly as class attribute and prepare $D_{tr}$;
            $R^u = GetExhaustiveRuleset\ (D, D_{tr}, \mathcal{A}, C_o, u)$;
            $Appear = \phi$ /* $Appear$ is set to empty */
            for $j = 1$ to $|R^u|$ do
                for $n = 1$ to $|\mathcal{A}|$ do
                    if $\mathcal{A}_n$ is in the antecedent of $R^u_j$ then
                        $Appear_n = Appear_n + 1$
            while $R^u \neq \phi$ AND $R^u$ contains at least a rule that has regular attribute/s in its antecedent do
                $T = \phi$ /* A set $T$ is set to empty */
                $n \leftarrow maxarg(Appear)$ /* Returns the index of the attribute that appears the most */
                for $j = 1$ to $|R^u|$ do
                    if $\mathcal{A}_n \in \mathcal{A}^r$ AND $\mathcal{A}_n$ is an antecedent of $R^u_j$ then
                        if The value of $\mathcal{A}_n$ is not suppressed before then
                            Recommend $u$ to suppress the value of $\mathcal{A}_n$
                        $T = T \cup R^u_j$
                $Appear_n = 0$ and $R^u \leftarrow R^u \setminus T$ /* $T$ is removed from $R^u$ */
        Step 2: Hide Friendship Links as Necessary if they are not fabricated previously. (Layer 2)
            while $R^u \neq \phi$ AND $R^u$ contains at least a rule that has link attribute/s (containing $\geq$) in its antecedent do
                $T = \phi$
                $n \leftarrow maxarg(Appear)$ and $v \leftarrow CalculateValue(\mathcal{A}'_n, u, D, G')$ /* using Equation (1) and $\mathcal{A}'_n$ =corresponding regular attribute
                of $\mathcal{A}_n$.*/
                for $j = 1$ to $|R^u|$ do
                    if $\mathcal{A}_n$ is an antecedent of $R^u_j$ AND $R^u_j$ contains $\geq SplitPoint(R^u_j, \mathcal{A}_n)$ then
                        while $v \geq SplitPoint(R^u_j, \mathcal{A}_n)$ do
                            $t_i \leftarrow FriendWithLeastDegree(u, G', F', \mathcal{A}_n)$ /* A user $t_i$ is in $u$'s friend list where $i$ indicates the user index */
                            if $t_i$ is not appeared in $F'$ then
                                Recommend $u$ to Hide $t_i$
                                $G' \leftarrow HideLink(G', u, t_i)$; $F' \leftarrow Flag(F', t_i)$; $v \leftarrow CalculateValue(\mathcal{A}'_n, u, D, G')$
                            Ignore $t_i$ for calculating next $FriendWithLeastDegree(u, G', F', \mathcal{A}_n)$.
                        $T = T \cup R^u_j$
                $Appear_n = 0$ and $R^u \leftarrow R^u \setminus T$
        Step 3: Add Friendship Links as Necessary if they are not fabricated previously. (Layer 3)
            while $R^u \neq \phi$ AND $R^u$ contains at least a rule that has link attribute/s (containing $<$) in its antecedent do
                $T = \phi$
                $n \leftarrow maxarg(Appear)$ and $v \leftarrow CalculateValue(\mathcal{A}'_n, u, D, G')$
                for $j = 1$ to $|R^u|$ do
                    if $\mathcal{A}_n$ is an antecedent of $R^u_j$ AND $R^u_j$ contains $< SplitPoint(R^u_j, \mathcal{A}_n)$ then
                        while $v < SplitPoint(R^u_j, \mathcal{A}_n)$ do
                            $t_i \leftarrow UserWithLeastDegree(u, G', F', \mathcal{A}_n)$
                            if $t_i$ is not appeared in $F'$ then
                                Recommend $u$ to Add $t_i$ in the friend list
                                $G' \leftarrow AddLink(G', u, t_i)$; $F' \leftarrow Flag(F', t_i)$; and $v \leftarrow CalculateValue(\mathcal{A}'_n, u, D, G')$
                            Ignore $t_i$ for calculating next $UserWithLeastDegree(u, G', F', \mathcal{A}_n)$.
                        $T = T \cup R^u_j$
                $Appear_n = 0$ and $R^u \leftarrow R^u \setminus T$
        $C \leftarrow C \setminus C_o$
    $Continue$ = "Ask $u$ to input $False$ to discontinue protecting privacy for rest of the sensitive attributes"
    if $Continue = False$ then
        break;
```

Similar to Step 1, *3LPEx* first identifies the *link* attribute $\mathcal{A}_n$ (where $\mathcal{A}_n \in A^l$) which is most frequent in R^u and computes $\mathcal{A}_n$'s value, denoted as v, using Eq. (1). If v is higher than the split point mentioned in R^u_j, then *3LPEx* suggests u to hide a friendship link to reduce the value below the split point. While choosing a friendship link, *3LPEx* selects a friend t_i (where i is a user index) of u who has the smallest degree. If t_i has not appeared in a friendship matrix F' (i.e. the friendship link between u and

t_i was not previously modified by *3LPEx*), *3LPEx* recommends u to hide t_i so that it can reduce v the most. Here, F' is a $1 \times N$ matrix which stores the friendship *Flag* information for u. If u follows the recommendation, *3LPEx* puts a *Flag* up in the ith column of the Friendship matrix F' and this user will no longer be recommended for further hiding or adding. The *3LPEx* then updates G', F', and recomputes v (see Step 2).

This process continues until v is lower than the *SplitPoint* mentioned in R^u_j. Once v is lower than the split point, the process of hiding friends stops. *3LPEx* then removes R^u_j and other rules (that require the value of $\mathcal{A}_n$ to be $\geq$ than the *SplitPoint*) from R^u. At the end of Step 2, if R^u is not empty, then *3LPEx* moves to Step 3, i.e. *Layer* 3.

Step 3: Suggest the User to Add Friendship Links as Necessary if they are not fabricated previously. (Layer 3)

After Step 1 and Step 2 if R^u is still not empty, the remaining rules contain only *link* attributes in their antecedents. If u satisfies the condition of a rule R^u_j which requires the value of a *link* attribute v to be less than a splitting point *SplitPoint*, then *3LPEx* recommends u to add new friends wisely so that the value of v increases, with the aim to make the value of v eventually greater than the *SplitPoint* appearing in the set R^u_j of rules.

3LPEx recommends u to add a user t_i in u's friend list if two conditions are fulfilled: 1. the flag in the ith column of F' matrix has previously been set to *False* and 2. The user t_i is chosen as having the smallest degree (i.e. the smallest $\Gamma_+(t)$ value). If u follows the recommendation, *3LPEx* then sets *True* in the ith column of the matrix F' so that user t_i will not be recommended for further hiding or adding. *3LPEx* also updates the friendship graph G' and increases the value of the *link* attribute value.

This process of adding friends continues as long as v is smaller than *SplitPoint*. Once v is greater than *SplitPoint*, R^u_j becomes ineffective for u to predict u's sensitive attribute value correctly and *3LPEx* then removes R^u_j and other rules (that require the value of $\mathcal{A}_n$ to be $<$ than the *SplitPoint*) from R^u. Adding an OSN user in the friend list would be inconvenient as it depends on the other users accepting the friendship invitation [43]. Therefore, *3LPEx* keeps *Layer* 3 as a last option for providing privacy.

At the end of *Layer* 3, R^u becomes empty, *3LPEx* removes the class attribute C_o from C and asks u whether to continue protecting privacy for the next sensitive attribute or not. If u agrees to continue (i.e. *Continue* = *True*) and the list of sensitive attribute $|C|$ is not empty, then the entire process (i.e. Step 1 to Step 3) iterates over the next sensitive attribute.

2.5 Complexity analysis

We now apply standard analysis of algorithms under the uniform-cost measurement [44] (that is, every machine operation, regardless of the size of the numbers involved has constant cost) to obtain the time-complexity expressions of our algorithms. Clearly, if we were to generate all rules with antecedents having k attributes (out of a data set with d attribute values), the complexity of the algorithm would be dominated by a term of the form $\binom{d}{k}$. Since typically k is much less than d, the complexity would be dominated by

$$\frac{(d/k - 0.5)^k e^k}{\sqrt{2\pi k}}$$

(by Stirling's approximation). However, as we have indicated, full privacy is only achievable with the unpalatable option of no data release. In what follows, we present a detailed analysis when the practical choice of exhaustive exploration of all rules with at most $k = 3$ attributes is adopted.

The overall complexity of *3LPEx* is the sum of the complexity of Algorithm 1 and Algorithm 2. First, we analyse the complexity of Algorithm 1 and then the complexity of Algorithm 2.

Step 1 of Algorithm 1 stores the domain values of each non-class attribute. Hence, the complexity of storing the domain values of an attribute is $O(N^2 + N + d)$. Therefore, the complexity of storing the domain values of b number of attributes is $O(bN^2 + bN + bd)$. Step 2 of Algorithm 1 generates a set of exhaustive rules and in each rule the number of antecedents of each rule is considered to be 3. Therefore, in our case, the number of attributes, b, in a data set should not be less than 3 i.e. $b \geq 3$.

We assume a data set containing three attributes (e.g. $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$), and each of the attribute's domain value is d. If the rules are generated using one attribute in their antecedent, then the complexity of generating such rules is $O(d)$. Therefore, the complexity of generating the rules with first two attributes is $O(d + d^2)$, and the complexity with the first three attributes is $O(d + d^2 + d^3)$. As mentioned earlier, the number of antecedents of each rule is considered to be 3 in our study, so the total complexity of generating rules is dominated by the third attribute i.e. $O(d^3)$. It is important to mention here that the number of appearance of $\mathcal{A}_3$ attribute in the third antecedent position of the generated rules is 1 as the matrix $\mathcal{H}$ propagates from left to right while building a rule and attributes $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot be in the third antecedent position of the generated rules (see Step 2 of Algorithm 1).

We now assume a data set containing four attributes and the attributes are $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ and $\mathcal{A}_4$. Here, the number of appearance of $\mathcal{A}_3$ attribute in the third antecedent position of the generated rules is 1 (the combination is $\mathcal{A}_1$-$\mathcal{A}_2$-$\mathcal{A}_3$). On the other hand, the number of appearance of $\mathcal{A}_4$ attribute in the third antecedent position is 3 ($\mathcal{A}_1$-$\mathcal{A}_2$-$\mathcal{A}_4$, $\mathcal{A}_1$-$\mathcal{A}_3$-$\mathcal{A}_4$, and $\mathcal{A}_2$-$\mathcal{A}_3$-$\mathcal{A}_4$. Therefore, the complexity of generating the exhaustive rules for four attributes is dominated by the fourth attribute. We can calculate as $O((4-2)d^3 + (4-3)d^3)$ or $O(3d^3)$. Similarly, for a data set with five attributes (e.g. $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$, $\mathcal{A}_5$), the complexity of generating exhaustive rules is $O((5-2)d^3 + (5-3)d^3 + (5-4)d^3))$ or $O(6d^3)$.

Therefore, for the b number of attributes, We can write the series, s, as:

$$s = [\{(b-2) + (b-3) + (b-4) + (b-5) + \cdots + 1\} \times d^3] \tag{3}$$

By writing Eq. (3) backward, we get:

$$s = [\{1 + 2 + 3 + \cdots + (b-4) + (b-3) + (b-2)\} \times d^3] \tag{4}$$

By adding Eqs. (3) and (4), we get

$$2s = [\{(b-1) + (b-1) + \cdots + (b-1)\} \times d^3] \tag{5}$$

The sum of this series is $\frac{(b-2)(b-1)}{2} \times d^3$.

Therefore, the complexity of Task 1 of Step 2 is $O\left(\frac{(b-2)(b-1)}{2} \times d^3\right)$ which can be simplified as $O(b^2d^3)$. To calculate the sensitivity of a rule, we compare the antecedents of a rule with b number of attributes of each record in D and hence the complexity is $O(bN|\mathcal{R}|)$. The overall complexity of Algorithm 1 is $O(bN^2 + b^2d^3 + bN|\mathcal{R}|)$.

We now analyse the complexity of Algorithm 2. At first, Step 1 of Algorithm 2 takes Algorithm 1 as an input (Task 1). Therefore, the complexity of Algorithm 1 is considered as the complexity of Task 1 i.e. $O(bN^2 + b^2d^3 + bN|\mathcal{R}|)$. In Task 2, the number of appearance of each attribute in the sensitive rule set is counted. The complexity of counting an attribute in the sensitive rule set is $O(r)$. Therefore, the complexity of counting b number of attributes in the sensitive rule set is $O(br)$. In Task 3, we suppress the *regular* attribute values that appear most in the sensitive rule set. If the total number of attributes is b and the total number of sensitive rules is r, then the complexity of Task 3 is $O(br)$. Therefore, the overall complexity of Step 1 of Algorithm 2 is $O(bN^2 + b^2d^3 + bN|\mathcal{R}| + 2br)$.

In Step 2 of Algorithm 2, the appearance of each link attribute in the sensitive rule set is first counted (Task 1). Therefore, the complexity of counting b number of attributes in the sensitive rule set is $O(br)$. In Task 2, to get rid of a sensitive rule, we calculate the metric value of the most appeared link attribute for u by using Eq. (1). If u has f number of friends and each friend having a degree of q, then complexity is $O(f^2q)$. The complexity to get rid of all the sensitive rules r is $O(f^2qr)$. So, the overall complexity of Step 2 of Algorithm 2 is $O(bs + f^2qr)$.

The complexity of Step 3 of Algorithm 2 is almost similar to the complexity of Step 2 of Algorithm 2 and that is $O(br + N^2qs)$ as u needs to search the N number of users in the data set for adding friends.

The overall complexity of Algorithm 2 is $O(bN^2 + b^2d^3 + bN|\mathcal{R}| + 3br + f^2qr + N^2qr)$ for a user to protect a sensitive attribute from attribute inference attack. The total complexity of Algorithm 2 to protect the privacy for c number of sensitive attributes is $O(cbN^2 + cb^2d^3 + cbN|\mathcal{R}| + cb^2r + cf^2qr + cN^2qr)$. For low-dimensional data sets (such as those used in this study), the complexity of *3LPEx* can be simplified to $O(N^2 + d^3)$.

3 Experiments

3.1 Data sets and notation

We use two OSN data sets in this study for experimental purposes and denoted as D_1 and D_2. Both data sets contain users' personal and friendship information that an attacker may utilise to launch the attack. The details of the data sets are given in Table 6. The first data set D_1 [41] was used in our earlier studies [33, 34]. We synthetically generate the second data set D_2 in this study. The synthetic data set D_2 and the data generator is available here: https://drive.-google.com/drive/folders/1My3V7h959X-UlEVe45F6f1y7rHlEidKV?usp=sharing. In this section, we first describe the data set D_2 generation process and then the distribution of records in the training and test data sets for both D_1 and D_2.

In order to generate records in D_2, we first select ten *regular* attributes, namely *Age range*, *Relationship status*, *City of residence*, *Number of friends on OSN*, *Number of pages followed on OSN*, *Number of uploa-ded photos per week*, *Number of comments made on contents per week*, *Political view*, *Religion*, and *Profes-sion*. Table 11 shows that what attribute values users may upload on their profiles. From the users preference, our method selects attributes randomly and categorises the attributes in three groups: "users' activities on OSN", "users' personal information" and "users' sensitive information".

In order to get meaningful rules from the data set, similar to previous studies [29, 45], we then generate the

Table 6 Data sets at a glance

Data set	Number of links	Number of records	Number of nonclass attributes	Class attributes	Class values
D_1	50,397	1000	21	Political view	Far left, left, centre left, centre, centre right, right, far right
				Religion	Christian, Muslim, Jewish, Hindu, Buddhist, Sikh, No-religion, Other-religion
				Sexual orientation	Absent sexual information, bisexual, heterosexual, homosexual
D_2	12,567,829	10,000	19	Political view	Liberal, labour, green party
				Religion	Christian, Muslim, Hindu, No-religion, Other-religion
				Profession	Govt.-employee, salesman, entrepreneur, student, retired-person

records based on some predefined logic rules. For example: If *Age range = 18-27* and *Number of friends on OSN = High* $\xrightarrow{then}$ *Profession = Student*. We take advantage of the Australian Bureau of Statistics website [46] to prepare the logic rule set and based on the rules, we then generate records in D_2. The logic rules are provided in "Appendix 2".

In order to generate the friendship links among the users, we set the probability of a link between two users by calculating record-to-record distance (or $R2RD \in [0, 1]$) as the Hamming distance divided by the number of total attributes (i.e. 10). Users having similar attribute values (i.e. low Hamming distance) are likely to have common interests and thus are likely to have friendship links between them. In D_2, if the value of the *R2RD* distance of any two records is 0.3 or less, we consider them as friends and place a link between the two records. Thus, we generate 12,567,829 friendship links among the 10,000 records and then calculate the *link* attribute values to insert into the data set.

Table 6 shows our assumption: users in the data sets consider up to three attributes as sensitive. For simplicity, we denote them as C_1, C_2, and C_3. We therefore prepare three versions of each data set; in each version we select an attribute as a class attribute and denote them accordingly. For example, if we select C_1 as the class attribute, then we denote the training data sets as D_{tr,C_1} and the testing data set as D_{ts,C_1}.

On the other hand, when we consider a particular attribute as a class attribute, then the rest of the sensitive attributes are selected to be non-class attributes. For instance, in D_{tr,C_1}, C_2 and C_3 are considered as non-class attributes.

3.1.1 Distribution of records in the data sets

For the validation of experimental results, we follow tenfold cross-validation methods throughout our experiments. V-fold cross-validation is standard technique in evaluation of accuracy in machine learning and V=10 offers low variance of the estimate, reliable estimation of the accuracy and not excessive validation cost. Other values of V do not provide much more precision on the estimate of a classifier accuracy, but consume more time to perform. With tenfold cross validation, the data are partitioned in ten random parts. For each of the parts, the remaining data (containing 90% of the total records) perform the role of training set and the part (containing 10% of the records) become the test data set.

In a real-world scenario, OSN users may have a diverse range of information that they consider to be sensitive. Therefore, we assume that different records (i.e. OSN users) in a test data set consider the set of sensitive attributes differently. While some users may consider C_1 as a sensitive attribute, some others may consider C_1 as a non-sensitive attribute. Moreover, while some users may consider only one attribute to be sensitive, some other users may consider two or three attributes to be sensitive. Hence, we labelled the records of the test data set into three groups (named as Group 1, Group 2, and Group 3) based on the number of sensitive attributes considered by the test data set users. Group 1 contains 6% of users who consider any one (either "C_1" or "C_2" or "C_3") attribute as sensitive, Group 2 contains 3% of users who consider any two attributes as sensitive, and finally Group 3 contains 1% of users who consider all the three attributes as sensitive.

Group 1 is again divided into three subgroups, they are subgroup 11: contains 2% of users who consider "C_1" as sensitive only, subgroup 12: contains 2% of users who consider "C_2" as sensitive only, subgroup 13: contains 2% of users who consider "C_3" as sensitive only. Similarly, Group 2 is again divided into three subgroups, they are subgroup 21: contains 1% of users who consider "C_1" and "C_2" as sensitive only, subgroup 22: contains 1% of users who consider "C_1" and "C_3" as sensitive only, subgroup 23: contains 1% of users who consider "C_2" and "C_3" as sensitive only.

While preparing a test data set (say D_{ts,C_1}), we select the records who consider "C_1" as sensitive and leave all other records in the training data set D_{tr,C_1}. For example, D_{ts,C_1} contains 5% of the total records, i.e. subgroup11 (2%), subgroup 21 (1%), subgroup 22 (1%), and Group 3 (1%) records and D_{tr,C_1} contains 95% of the total records.

3.2 Experimental set-up

In this experimental set up, we consider that there are at most three attributes that can be considered to be sensitive. We present the experimental set-up here in three phases. In each phase, the privacy techniques secure a sensitive attribute. For example, privacy of sensitive attribute C_1 is protected in Phase I, C_2 in Phase II, and C_3 in Phase III.

3.2.1 Phase I

Phase I is comprised of five steps.

Step 1: Preparation of training and test data sets.

At the beginning of Phase 1, shown in Fig. 3, the privacy preserving techniques randomly select a sensitive attribute C_1 (from the set of sensitive attributes) as a class attribute and prepare training data set D_{tr,C_1} and testing data set D_{ts,C_1} from the main data set D.

Step 2: Application of existing classifiers before applying any privacy techniques on the test data sets.

After preparing the training and test data sets, we investigate the *attack success rate* of an attacker $\mathcal{Z}$ to infer private information about the users in the test data set before applying any privacy preserving technique. To do so, we train a set of existing classifiers (that $\mathcal{Z}$ might use to breach privacy) on D_{tr,C_1} and we denote the classifiers as "*Classifiers 1*" as shown in Fig. 3. We then apply *Classifiers 1* on D_{ts,C_1} to identify the *attack success rate* in absence of any privacy technique. The results are analysed and compared with the results that will be achieved later in *Step 4*. We use the same set of classifiers (in *Classifiers 1*, *Classifiers 2*, ...*Classifiers 6*) throughout our experiments to test the *attack success rate*.

Step 3: Application of privacy techniques to ensure privacy of attributes users consider confidential.

In this step, we apply the privacy techniques, namely *3LP+*, *PrivNB*, and our proposed algorithm *3LPEx*, to provide privacy to the users in the test data set (i.e. D_{ts,C_1}). Please note that due to the modification of friendship links in test data sets (for the sake of providing privacy), there may be a change in *link* attribute values in the training data set as there are friendship links among the users in the training and test data set. We use (from this point forward) two different superscripts " $'$" and " $*$" to denote the modified data sets. The data sets modified by our proposed technique *3LPEx* are denoted by " $'$ ". On the other hand, the data sets modified by any other privacy preserving technique (e.g. *3LP+* and *PrivNB*) are denoted by " $*$ ". For example, the resultant secure test data sets D'_{ts,C_1} is secured by *3LPEx* and D^*_{ts,C_1} is secured by another privacy technique. Note that in these experiments we only secure records in test data sets.

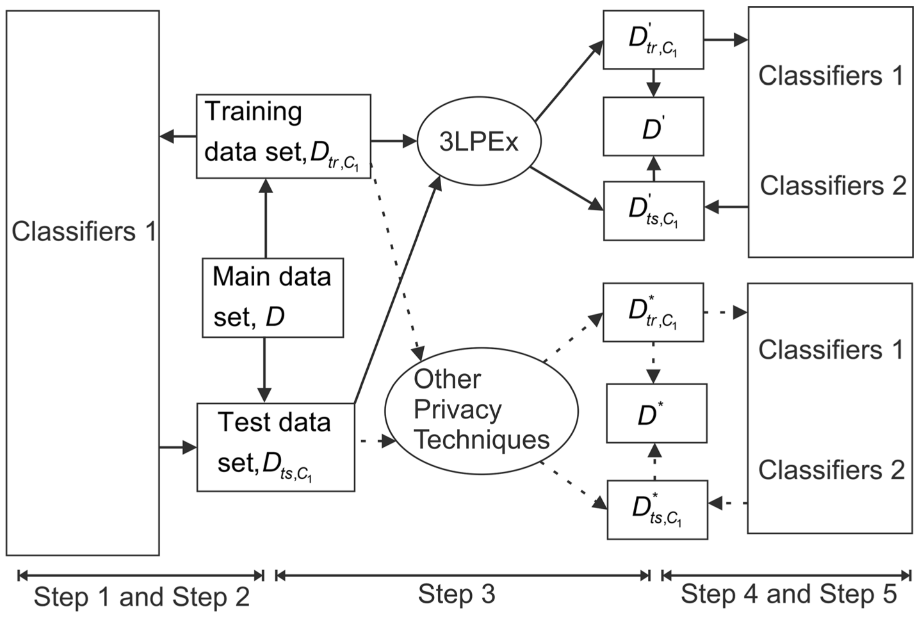


Fig. 3 Phase I of the experiments

Step 4: Application of existing classifiers after applying privacy techniques on the test data sets.

In this step, the robustness of secured test data sets against an attack is analysed and compared with the results achieved in *Step 2*. We first train the classifiers (denoted as *Classifiers 2*) with a modified training data set D'_{tr,C_1} or D^*_{tr,C_1}. Please note that *Classifiers 2* obtained from D'_{tr,C_1} (see the top right rectangle of Fig. 3) are different from the *Classifiers 2* obtained from D^*_{tr,C_1} (see the bottom right rectangle of Fig. 3), in the sense that they are built from different training data sets. We then apply *Classifiers 2* on secure test data sets D'_{ts,C_1} and D^*_{ts,C_1}, respectively. We also apply *Classifiers 1* on D'_{ts,C_1} and D^*_{ts,C_1} to evaluate the *attack success rate* with and without the privacy protection techniques. *Attack success rate* refers to the number of users in test data sets whose class values can be correctly inferred by the classifiers. Smaller *accuracy* indicates better privacy protection. *Step 4* is divided in two substeps: *Step 4a* and *Step 4b*.

In *Step 4a*, we analyse the accuracy of *Classifiers 1* on secure test data sets D'_{ts,C_1} and D^*_{ts,C_1}. In *Step 4b*, we analyse the accuracy of Classifiers 2 (trained by modified training data set) on secure test data sets D'_{ts,C_1} and D^*_{ts,C_1}.

Step 5: Measurement of data utility and inclusion of the records from secured test data set into the main data set.

We calculate data utility of the secure test data sets, i.e. D'_{ts,C_1} and D^*_{ts,C_1}, in this step. The data utility is measured in terms of the number of suppressed attribute values where the less number of suppression indicates higher utility. At the end of *Step 5*, we return all the records (from training and testing data sets) to D and the original data set D is now modified to D' and D^*.

Please note that irrespective of the classification algorithms used by the privacy techniques (i.e. *3LPEx* uses an exhaustive set of axis -parallel rules while *3LP+* uses *SysFor*). In our experiments, to evaluate the potential penetration by an adversary, we use a number of classification algorithms such as *Random Forest*, Support Vector Machine (*SVM*), and *Logistic Regression*. For the full list of classification algorithms used in the experiments, please see Fig. 9. *Classifiers 1*, *Classifiers 2*, ... *Classifiers 6* (see Figs. 3, 4, 5) use all of these classification algorithms, one by one, in our experiments. The goal of these experiments is to evaluate the *attack success rate* of each of these classification algorithms, to understand the impact of the use of these algorithms on the privacy of the users. In the experiment where we use a particular classification algorithm (say *Logistic Regression*) as *Classifiers 1*, we continue to use the same classification algorithm for all other classifiers, i.e. *Classifiers 2*, *Classifiers 3* etc.

It is also important to clarify that *Classifiers 2* in the top rectangle and bottom rectangle on the right side of Fig. 3 are not the same classifiers. While *Classifiers 2* on the top rectangle are built from D'_{tr,C_1} and the *Classifiers 2* on the bottom rectangle are built from D^*_{tr,C_1}. However, *Classifiers 1* are built from D_{tr,C_1} as shown in the left side rectangle in Fig. 3.

Classifiers 3 in the top left side and top right-side rectangles of Fig. 4 are built from D'_{tr,C_2}. *Classifiers 3* in the bottom left side and bottom right-side rectangles of Fig. 4 are built from D^*_{tr,C_2}. However, *Classifiers 4* in the top right side rectangle are once built from $D^{2'}_{tr,C_2}$ and another time from $D^{2'}_{tr,C_1}$ and then tested on $D^{2'}_{ts,C_2}$ and $D^{2'}_{ts,C_1}$, respectively. Similarly, *Classifiers 4* in the bottom right side rectangle are once built from $D^{2^*}_{tr,C_2}$ and another time from $D^{2^*}_{tr,C_1}$ and then applied on $D^{2^*}_{ts,C_2}$ and $D^{2^*}_{ts,C_1}$, respectively.

3.2.2 Phase II

Similar to Phase I, Phase II is also completed in five steps: Step 6 to Step 10 as shown in Fig. 4.

Step 6: Preparation of the training and test data sets.

At the beginning of Phase II, *3LPEx* randomly selects a sensitive attribute C_2 (from the set of sensitive attributes) as a class attribute and then prepares the training data set D'_{tr,C_2} and test data sets D'_{ts,C_2} from D'. Similarly, D^*_{tr,C_2} and D^*_{ts,C_2} are prepared from D^* by the other privacy techniques (see Fig. 4).

Step 7: Application of existing classifiers before applying the privacy techniques on the test data sets.

In *Step 7*, before applying any privacy technique, a set of classifiers, namely *Classifiers 3*, are first trained from the training data sets D'_{tr,C_2} and D^*_{tr,C_2}. It is important to note that *Classifiers 3* trained by D'_{tr,C_2} are different from *Classifiers 3* trained by D^*_{tr,C_2}. For simplicity, we denote the both classifier models here as *Classifiers 3* and shown in the left-side rectangle in Fig. 4. Once the classifier models are built, they are applied on the test data sets, i.e. D'_{ts,C_2} and D^*_{ts,C_2}. The *Classifiers 3* results are kept for analysing and comparing with the *Classifiers 4* results that will be achieved in *Step 9*.

Step 8: Application of privacy techniques to secure users in test data sets.

In this step, we apply *3LPEx* and other two privacy preserving techniques (i.e. *3LP+* and *PrivNB*) on D'_{ts,C_2} and D^*_{ts,C_2}, respectively, to secure the privacy of the users. It is important to mention that both of these test data sets were secured previously in Phase I for C_1. In Phase II, we implement the privacy techniques to secure them for C_2. Due to the implementation of the privacy techniques, the

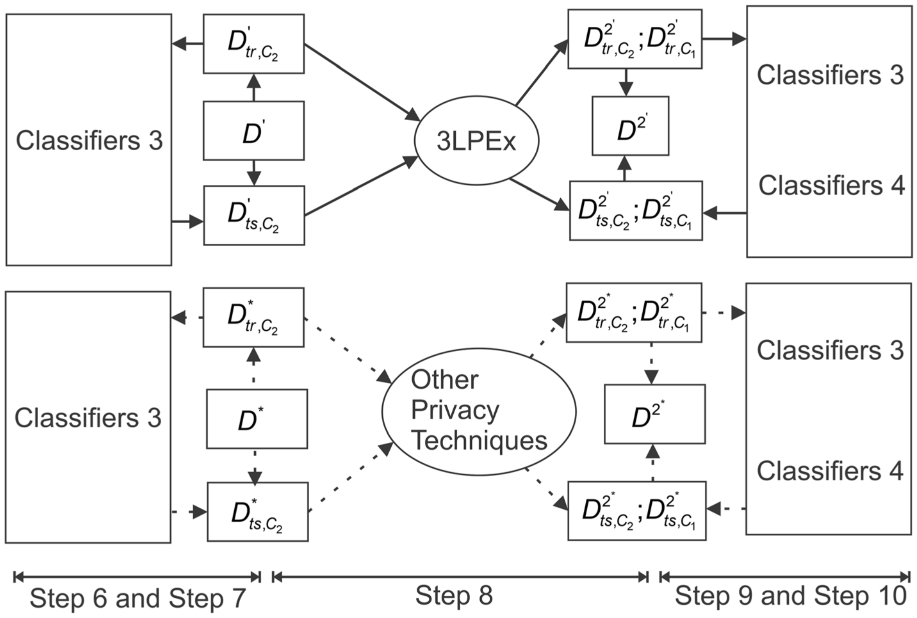


Fig. 4 Phase II of the experiments

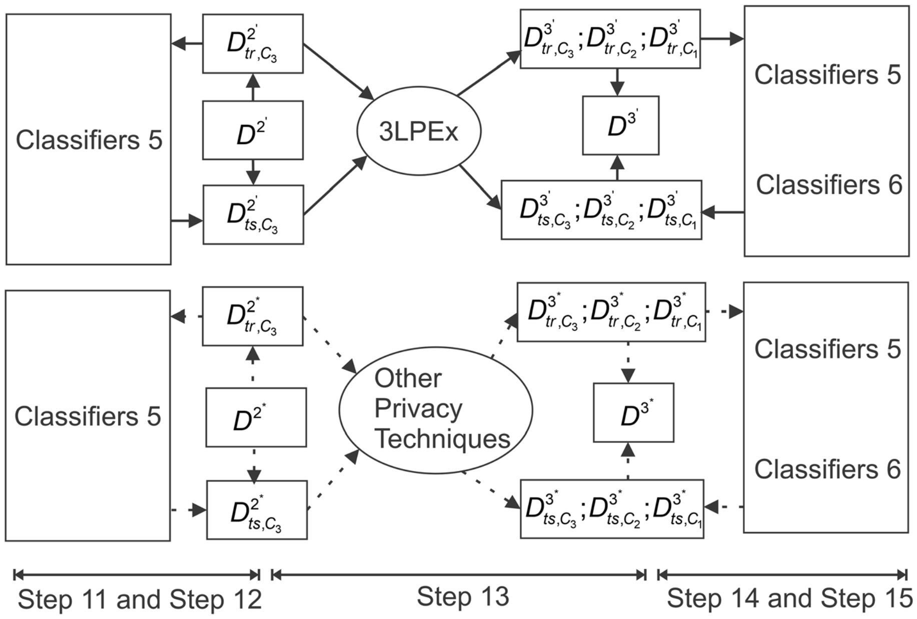


Fig. 5 Phase III of the experiments

training and testing data are changed. Here, we denote the modified test and training data sets by $D^{2'}_{ts,C_2}$ and $D^{2'}_{tr,C_2}$ for *3LPEx*. Similarly we denote $D^{2^*}_{ts,C_2}$ and $D^{2^*}_{tr,C_2}$ to represent the test and training data sets modified by the other privacy preserving techniques.

Step 9: Application of existing classifiers after applying the privacy techniques on the test data sets.

In this step, a classifier model, i.e. *Classifiers 4*, is trained from the $D^{2'}_{tr,C_2}$ and $D^{2^*}_{tr,C_2}$ separately (as shown in the right side in Fig. 4) and then applied on test data sets $D^{2'}_{ts,C_2}$ and $D^{2^*}_{ts,C_2}$, respectively. We also apply *Classifiers 3* on $D^{2'}_{ts,C_2}$ and $D^{2^*}_{ts,C_2}$ for analysing and comparing the number of insecure users, and prediction accuracy of different conventional classifiers (as mentioned in *Step 7*). Therefore, *Step 9* is divided in two substeps: *Step 9a* and *Step 9b*.

In *Step 9a*, we analyse the accuracy of *Classifiers 3* on secure test data sets $D^{2'}_{ts,C_2}$ and $D^{2^*}_{ts,C_2}$. In *Step 9b*, we analyse the accuracy of *Classifiers 4* (trained by modified training data set) on secure test data sets $D^{2'}_{ts,C_2}$ and $D^{2^*}_{ts,C_2}$.

Step 10: Measurement of data utility and return the secure test data set records into the main data set.

In this step, we measure data utility for each test data set $D^{2'}_{ts,C_2}$ and $D^{2^*}_{ts,C_2}$. After that, we return all the records to the main data set. The main data set is now modified to $D^{2'}$ and D^{2^*} for *3LPEx* and other privacy preserving algorithms, respectively.

For the sake of providing privacy to the users (who consider C_2 as sensitive attribute), there can be a privacy breach for the other users who consider C_1 as sensitive. Therefore, we also investigate the safety of users (who consider C_1 as sensitive) in $D^{2'}_{ts,C_1}$ and $D^{2^*}_{ts,C_1}$ in this step.

3.2.3 Phase III

The privacy techniques provide privacy for the third sensitive attribute C_3 in Phase III. Similar to Phase I and Phase II, Phase III is also completed in five steps: Step 11 to Step 15 as shown in Fig. 5.

Step 11: Preparation of the training and test data sets.

In this step, *3LPEx* selects the remaining sensitive attribute, i.e. C_3, (from the set of sensitive attributes) as a class attribute and then prepares training data set $D^{2'}_{tr,C_3}$ and testing data sets $D^{2'}_{ts,C_3}$ from $D^{2'}$. Other privacy preserving techniques also prepare training data set $D^{2^*}_{tr,C_3}$ and test data set $D^{2^*}_{ts,C_3}$ from D^{2^*} in the similar manner.

Step 12: Application of Classifiers' performance to invade privacy.

In this step, a set of classifiers, namely *Classifiers 5*, are first trained from the training data sets $D^{2'}_{tr,C_3}$ and $D^{2^*}_{tr,C_3}$ (before applying any privacy preserving techniques) as shown in the left side in Fig. 5. *Classifiers 5* is then applied on the test data sets $D^{2'}_{ts,C_3}$ and $D^{2^*}_{ts,C_3}$, respectively. The results are kept for analysis and comparing purposes with the results which will be achieved in *Step 14*.

Step 13: Application of privacy techniques to secure users.

In this step, *3LPEx* and other privacy preserving techniques are on $D^{2'}_{ts,C_3}$ and $D^{2^*}_{ts,C_3}$. It is important to mention that both $D^{2'}_{ts,C_3}$ and $D^{2^*}_{ts,C_3}$ were secured previously in Phase I for C_1 and in Phase II for C_2. In this phase, we apply the privacy techniques to secure the users who consider C_3 as sensitive attribute. After application of *3LPEx*, the training and testing data are modified and denoted as $D^{3'}_{ts,C_3}$ and $D^{3'}_{tr,C_3}$ for *3LPEx*. Similarly, we use $D^{3^*}_{tr,C_3}$ and $D^{3^*}_{ts,C_3}$ to denote the modified training and test data sets by the other privacy preserving techniques.

Step 14: Application of existing classifiers after applying the privacy techniques on the test data sets.

In this step, classifier model, i.e. *Classifiers 6*, is trained from $D^{3'}_{tr,C_3}$ and $D^{3^*}_{tr,C_3}$ separately and then applied on test data sets $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$, respectively. On the other hand, *Classifiers 5* is also applied on $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$ for analysing and comparing with the *attack success rate* achieved in *Step 12*. Therefore, *Step 14* is divided in two substeps: *Step 14a* and *Step 14b*.

In *Step 14a*, we analyse the accuracy of *Classifiers 5* on privacy-protected test data sets $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$. In *Step 14b*, we analyse the accuracy of *Classifiers 6* (trained by modified training data set) on secure test data sets $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$.

Step 15: Measurement of data utility and inclusion of the privacy-protected test data set records into the main data set.

After securing the test data sets, similar to Phase II, In *Step 15a* and *Step 15b*, respectively, we again analyse the *attack success rate* in $D^{3'}_{ts,C_1}$, $D^{3^*}_{ts,C_1}$, and $D^{3'}_{ts,C_2}$ $D^{3^*}_{ts,C_2}$ (who consider C_1 and C_2 as sensitive attribute). The privacy-protected test data sets (including $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$) are also analysed for the utility.

Therefore, in our experiments, *Step 15* is divided in two substeps: *Step 15a* and *Step 15b*. In *Step 15a*, we analyse the classifiers' accuracy of breaching users' (who consider C_1 as a sensitive attribute) privacy in $D^{3'}_{ts,C_1}$ and $D^{3^*}_{ts,C_1}$ after protecting users' privacy for C_3. In *Step 15b*, we analyse the classifiers' accuracy of breaching users' (who consider C_2 as a sensitive attribute) privacy in $D^{3'}_{ts,C_2}$ and $D^{3^*}_{ts,C_2}$ after protecting users' privacy for C_3. After securing the privacy for C_3, we return all the records to the main data set which becomes $D^{3'}$ and D^{3^*} for *3LPEx* and other privacy techniques, respectively.

The descriptions of the main experimental steps (described in Phase I to Phase III and shown in Figs. 3, 4 and 5) are summarised in Table 7.

4 Experimental results and discussion

We present the experimental results here in four subsections. In Sect. 4.1, we present the experimental results (i.e. the degree of privacy) achieved by *3LPEx* against an attacker who uses the *exhaustive* approach (i.e. uses the exhaustive approach to build the classifiers) to launch the attack. In Sect. 4.2, we present the experimental results of *3LPEx*, if an attacker uses a well-known existing classifier to launch the attack. Finally, in Sect. 4.3, we analyse and compare the data utility of the proposed technique in terms of the number of suppressed *regular* attribute values. In Sect. 4.4, we analyse the inference risk of a sensitive information after protected by the *3LPEx* algorithm.

Table 7 Summary of the main experimental steps

Steps	Description of the steps
2	The initial stage before applying any privacy preserving techniques on the test data sets
4*a*	The classifiers' (trained by unmodified training data set) accuracy on protected test data set (for the first sensitive attribute) after applying privacy preserving techniques
4*b*	The classifiers' (trained by modified training data set) accuracy on protected test data set (for the first sensitive attribute) after applying privacy preserving techniques
6	The initial stage (for the second sensitive attribute) before applying any privacy preserving techniques on the test data sets
9*a*	The classifiers' (trained by unmodified training data set) accuracy on protected test data set (for the second sensitive attribute) after applying privacy preserving techniques
9*b*	The classifiers' (trained by modified training data set) accuracy on protected test data set (for the second sensitive attribute) after applying privacy preserving techniques
10	The classifiers' (trained by modified training data set) accuracy on protected test data set (for the first sensitive attribute) after applying privacy preserving techniques
11	The initial stage (for the third sensitive attribute) before applying any privacy preserving techniques on the test data sets
14*a*	We analyse the accuracy of Classifiers 5 (trained by unmodified training data set) on secure test data sets $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$
14*b*	We analyse the accuracy of Classifiers 6 (trained by modified training data set) on secure test data sets $D^{3'}_{ts,C_3}$ and $D^{3^*}_{ts,C_3}$
15*a*	We analyse the classifiers' accuracy of breaching users' privacy (who consider C_1 as sensitive attribute) in $D^{3'}_{ts,C_1}$ and $D^{3^*}_{ts,C_1}$ after protecting users' privacy for C_3
15*b*	We analyse the classifiers' accuracy of breaching users' privacy (who consider C_2 as sensitive attribute) in $D^{3'}_{ts,C_2}$ and $D^{3^*}_{ts,C_2}$ after protecting users' privacy for C_3

The experimental results, presented in Sects. 4.1 and 4.2, are shown in terms of *attack success rate* percentage (y-axis) observed at each step (x-axis).

4.1 Protection against the exhaustive approach

In Fig. 6, we present the number of insecure users whose class values can still be inferred if an attacker applies the same classifier (i.e. *exhaustive* approach) to breach the privacy. We first provide privacy by *3LPEx* and the two other privacy preserving techniques (i.e. *3LP+* and *PrivNB*) separately as described in Sect. 3.2. Then, we apply an *exhaustive* approach to build a classifier for classifying the records in the test data set.

Similar to our previous study [34], in this study, we also utilise *attack success rate* as a scale to determine the percentage of users whose privacy is compromised. We observe in Fig. 6a that the *attack success rate* for *3LP+* and *PrivNB* is much higher than *3LPEx* except in Step 2. In Fig. 6b, we can see a similar pattern of *attack success rate* for D_2. In both data sets, *3LPEx* outperforms *3LP+* and *PrivNB*.

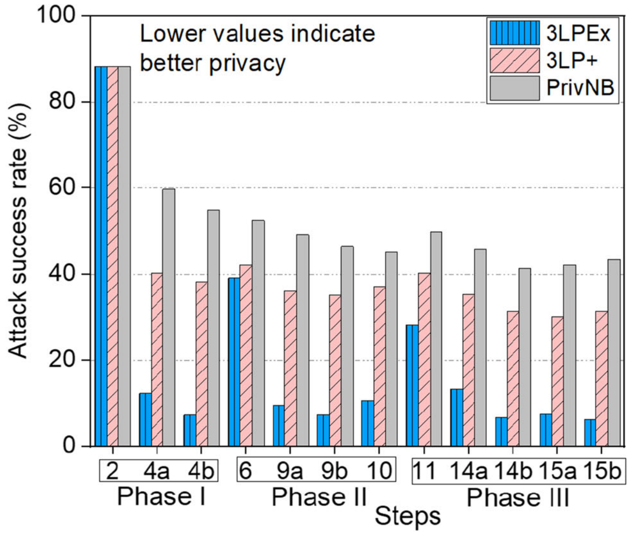
Compared to other steps, we observe a high *attack success rate* in Step 2, Step 6, and Step 11 of Fig. 6a and b as these steps are the initial steps of Phase I, Phase II, and Phase III, respectively. Please see the first, fourth, and eighth rows of Table 7 for the description of Step 2, Step 6, and Step 11, respectively.

In our experiments, Step 2 indicates the initial stage where privacy protection techniques are yet to be implemented on the test data sets. Therefore, the *attack success rate*, in Step 2, is the same for all the three protection techniques. In Step 6 and in Step 11, the second and third sensitive attributes are selected, respectively, as class attributes of a data set and the privacy preserving algorithms are yet to be applied on the data set.

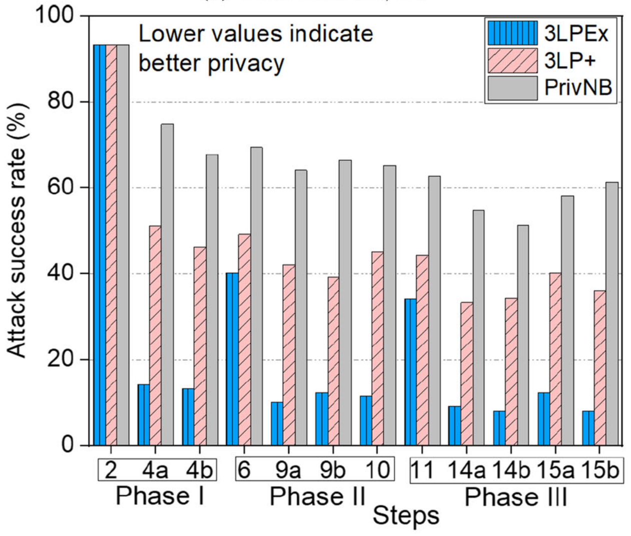
4.2 Protection against the existing classifiers

An attacker can utilise any classifier rather than the one used by the privacy preserving technique to infer the target users' hidden information. Therefore, we test the performance of our proposed technique against different classifiers and present the results in terms of *attack success rate* observed in each step.

In order to conduct the experiments of this subsection, we utilise the classifier packages from WEKA [47]. In a real-world scenario, an attacker could use any machine learning algorithm; in particular, the WEKA package offers a diversity of classifier learning algorithms whose implementations is acknowledged as robust by the community as it won the 2005 SIGKDD Data Mining and Knowledge Discovery Service Award. We select a set of existing classifiers, namely *Naïve Bayes* (NB) [48], *SVM* [49, 50],

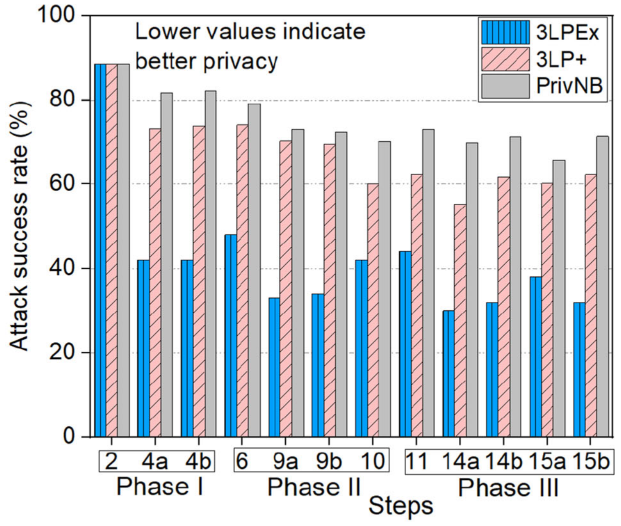


(a) First data set, D_1

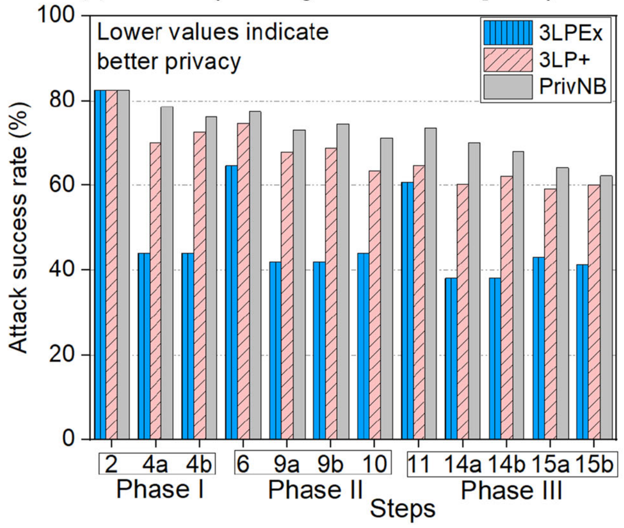


(b) Second data set, D_2

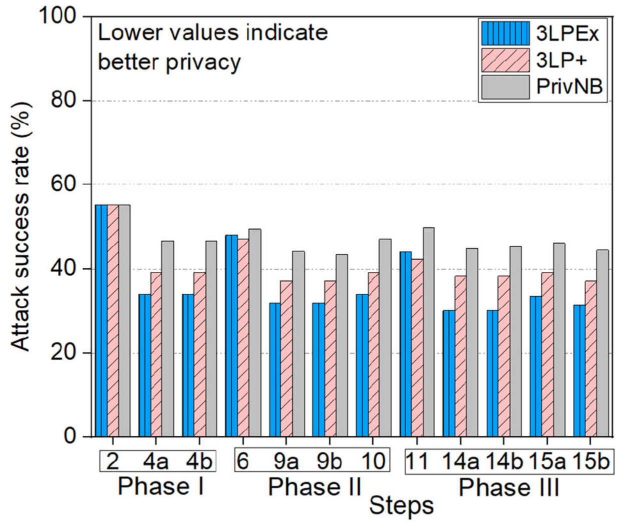
Fig. 6 *Attack success rate* accuracy of *exhaustive* approach to invade privacy on two data sets



(a) Naïve Bayes being used to breach privacy



(b) Support Vector Machine being used to breach privacy



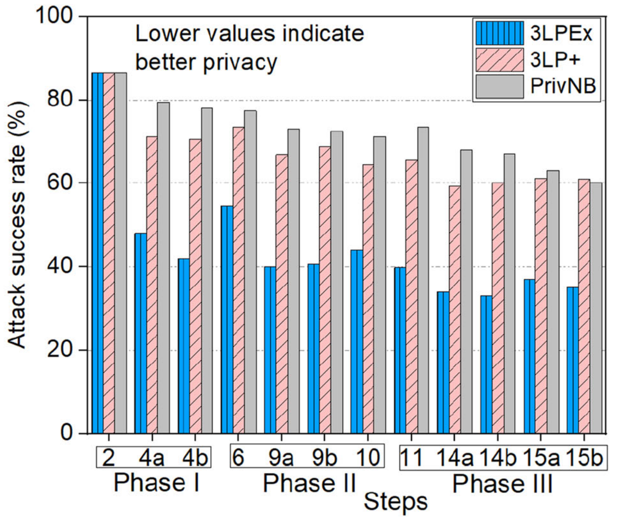
(c) Average of all the classifiers being used to breach privacy

Fig. 7 *Attack success rate* accuracy of different classifiers to invade privacy on first data set D_1

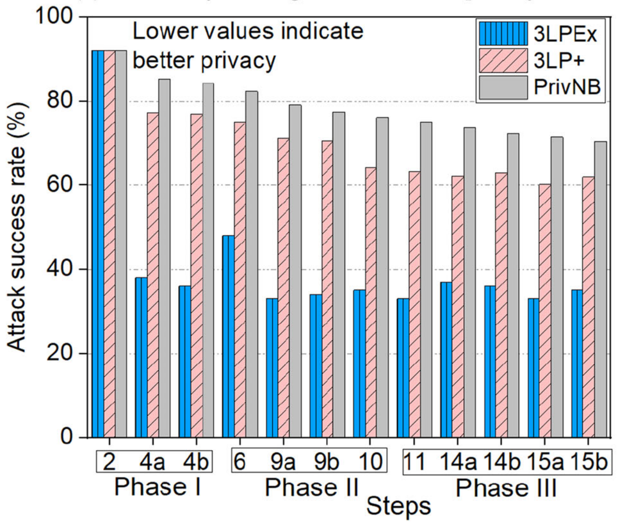
Logistic Regression [51], and *J48* [52]. In addition, we consider some decision forest algorithms, namely *Random Forest* (RF) [35], *AdaBoost* [38], *Bagging* [36], *SysFor* [29], *ForestPA* [40], and *Random Subspace* [37], that an attacker may utilise for inferring private information from D_1 and D_2.

We present the results of D_1 in Fig. 7 and D_2 in Fig. 8. Figure 7a shows the *attack success rate* when an attacker uses *Naïve Bayes* (denoted as *NB*) to breach privacy of records protected by the privacy preserving techniques. We can see from Fig. 7a that *3LPEx* provides better privacy than the other privacy techniques in each step.

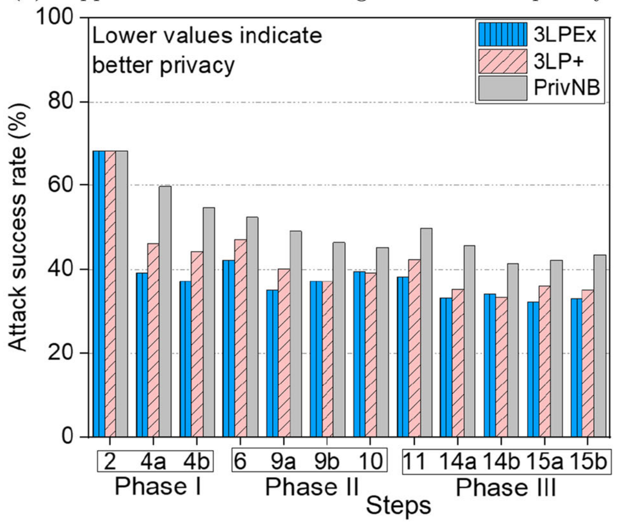
In Step 2, the *attack success rate* is the same for all the privacy techniques but it drops significantly (more than 40 percent) when we apply *3LPEx*. Similar trend is observed



(a) Naïve Bayes being used to breach privacy



(b) Support Vector Machine being used to breach privacy



(c) Average of all the classifiers being used to breach privacy

Fig. 8 *Attack success rate* accuracy of different classifiers to invade privacy on second data set D_2

throughout the experimental steps, and it is obvious that our technique clearly outperforms the existing privacy techniques. We can see a similar results for the *Support Vector Machine* classifier shown in Fig. 7b. Similar to the *Naïve Bayes* and *SVM* results, *3LPEx* outperforms the previous privacy preserving techniques for all other classifiers.

We present the results of D_2 in Fig. 8 where we can observe a similar trend.

The two classifiers against which *3LPEx* can reduce the *attack success rate* value the most, in D_2 data set, are also *NB* and *SVM* as shown in Fig. 8a and b, respectively. The average of all the classifiers' *attack success rate* percentage, shown in Fig. 8c, indicates the superiority of *3LPEx* over the existing privacy techniques.

Although *3LPEx* provides higher privacy than the existing privacy preserving techniques, we can see from Figs. 6, 7, and 8 that the *attack success rate* is still not zero. That is, often an attacker can be successful in inferring the sensitive information of a user. However, *3LPEx* significantly reduces the *attack success rate* reducing the confidence level of an attacker. according to the privacy definition provided in Sect. 1.1.1 the reduced confidence will support privacy protection by making $Pr(C_o = \mathcal{L}) < \gamma$.

Moreover *3LPEx* can provide higher privacy by lowering the threshold for sensitive rules. Hence, we conduct an experiment, on data set D_1, by reducing the sensitive threshold values to observe the change in *attack success rate*. The result is presented in Fig. 9. In Fig. 9, the *x*-axis represents different sensitivity threshold levels and the *y*-axis represents the *attack success rate*. Figure 9 shows that, by reducing the sensitivity threshold values, we can reduce the *attack success rate*.

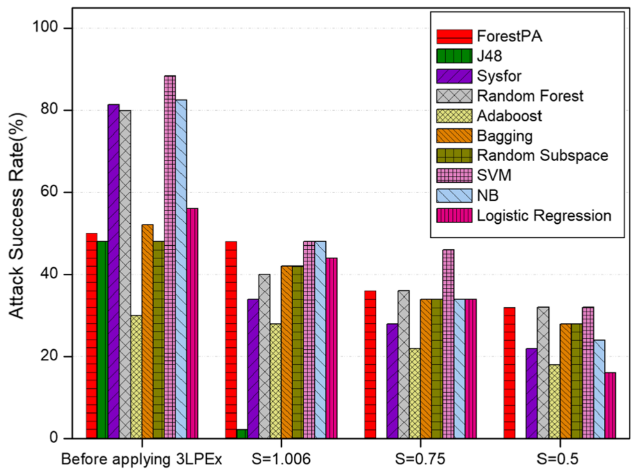
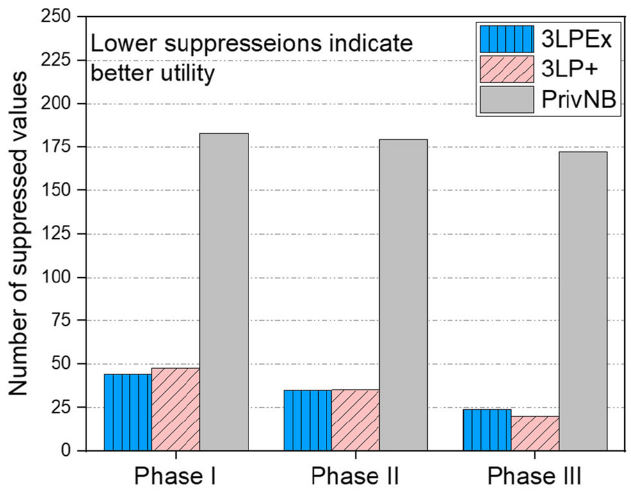


Fig. 9 Privacy levels achieved by *3LPEx* when different classifiers used to breach the privacy. We use different sensitivity thresholds to see their impact on privacy protection

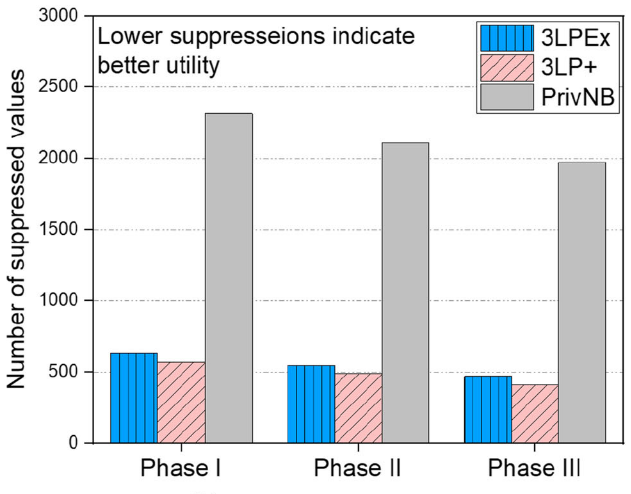
At lower sensitivity threshold, *3LPEx* considers more logic rules as sensitive than at higher sensitivity threshold values. More sensitive rules suggest the user suppresses more *regular* attribute values.

4.3 Data utility

We explore the data utility of *3LPEx* and compare the results with *3LP+* and *PrivNB* as both of these techniques suggest their users to suppress necessary *regular* attribute values in order to eliminate this privacy attack. Therefore, we compare the number of suppressed attribute values suggested by each privacy preserving technique. As mentioned earlier, the data utility is high when the number of suppressed attribute values is low. The results are presented in Fig. 10 where the x-axis represents the three phases of the experiments and the y-axis represents the number of suppressed values.



(a) First data set, D_1



(b) Second data set, D_2

Fig. 10 Comparison of the number of suppressed values

As mentioned in Sect. 3.1, each test data set contains 5% of the total records and each record contains at most ten nonclass *regular* attributes in D_1 and nine nonclass *regular* attributes in D_2. Therefore, in D_1, each test data set contains 500 *regular* attribute values (50 records multiplied with 10 *regular* attribute values) before applying any privacy techniques.

As shown in Sect. 3.1.1, some of the records may consider more than one attribute as sensitive and hide that attribute value. Therefore, the number of attribute values varied in different test data sets. As an example, for D_1, in the test data set D_{ts,C_1} exactly 20 users consider a single attribute as sensitive, 20 more users consider two attributes as sensitive, and ten other users consider three attributes as sensitive. Therefore, the maximum number of attribute values in D_{ts,C_1} is 460 (i.e. 20 users $\times$ 10 *regular* attributes +20 users $\times$9 *regular* attributes+10 users $\times$ 8 *regular* attributes=460). For D_2 data set, the available *regular* attribute values in D_{ts,C_1} is 4100.

However, this number is not the same for the rest of the two phases' test data sets i.e. D_{ts,C_2}, and D_{ts,C_3}. After applying the privacy preserving techniques on D_{ts,C_1}, the available number of *regular* attribute values will be different on D_{ts,C_2}, and D_{ts,C_3}, but less than D_{ts,C_1}.

It is obvious from both Fig. 10a and b that both *3LPEx* and *3LP+* outperform *PrivNB* in maintaining the data utility. For D_1, the average number of attribute value suppressions in each phase by *PrivNB* was 170, whereas for *3LPEx* and *3LP+* the number of suppressed values is less than 50. On the other hand, the number of suppressed attribute values by *3LPEx* is less than *3LP+* in the first two phases. However, in Phase III, *3LP+* outperforms our proposed algorithm *3LPEx*. In the case of data set D_2, shown in Fig. 10b, *3LP+* can provide slightly better utility than *3LPEx*.

While generating the sensitive rules, *3LPEx* takes an exhaustive approach by considering all attributes in a data set. Therefore, both *regular* and *link* attributes get the equal opportunity to be considered as antecedents in sensitive rules. However, *3LP+* uses *SysFor* algorithm that takes a greedy approach to generate rules where the attributes are selected as antecedents based on their gain ratio. In our experiments, the sensitive rules those generated by *3LP+* contain more *link* attributes than the rules generated by *3LPEx*. Therefore, less suppressions are required for *3LP+* than *3LPEx* to protect users' privacy. Although the suppression number is higher for *3LPEx*, we observe that *3LPEx* offers better privacy than *3LP+* in all the three phases (see Figs. 6, 7, 8).

4.4 Inference risk analysis

We, throughout this study, consider that an attacker launches the attack by utilising the full network information. For example, we consider that the same number of records used both in the protection techniques and the inference attack. Therefore, the calculated *link* attribute values used in the data sets are the same for the protection techniques and the inference attack. However, in a real-world scenario, the attacker may have limited access to an OSN and thus the privacy protected by the proposed protection technique using full information of the OSN can be vulnerable against the *attribute inference attack* that launched by using partial network information. Therefore, in this section, we analyse the inference risk of a sensitive information after protected by *3LPEx*. Here, "inference risk" defines the probability of a sensitive information being inferred by an attacker.

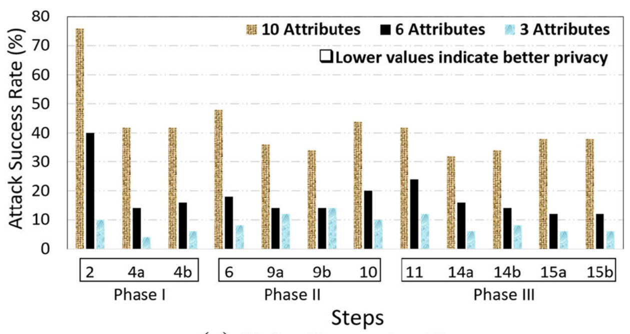
Consider a target user u considers C_o attribute to be sensitive where C_o has two values C_{o1} and C_{o2}. If $\mathcal{A}$ is the set of non-class attributes that u disclosed their values, then the posterior probability of class value C_{o1} can be calculated by using *Naïve Bayes* [48] equation as follows:

$$P(C_{o1}|\mathcal{A}) = (P(\mathcal{A}_1|C_{o1}) \times P(\mathcal{A}_2|C_{o1}) \times \ldots \times P(\mathcal{A}_n|C_{o1})) \times P(C_{o1}) \tag{6}$$

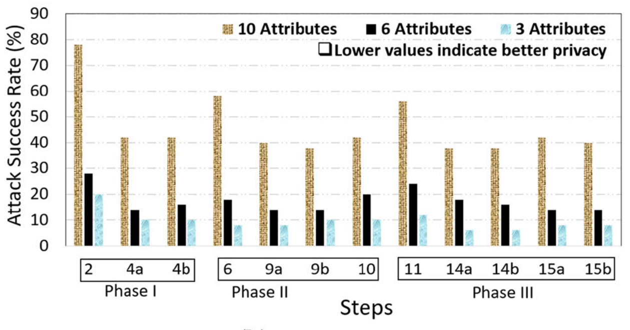
Here, $P(C_{o1}|\mathcal{A})$ is the posterior probability of class value C_{o1} for $\mathcal{A}$, $P(C_{o1})$ is the probability of class value C_{o1}, $P(\mathcal{A}_n|C_{o1})$ is the likelihood of $P(\mathcal{A}_n)$ for the C_{o1} class value. Similarly, we can calculate the posterior probability of C_{o2}. A high value of posterior probability indicates the high chance for the attacker to infer u's sensitive attribute value (i.e. either C_{o1} or C_{o2} in this running example). On the other hand, the posterior probability, presented in Eq. (6), can be reduced by hiding the non-class attribute values. We now experimentally demonstrate the inference risk (in terms of *attack success rate*) can be reduced significantly if the number of disclosed attribute values are reduced in the training data set.

We utilise D_1 data set for this experiment where we first consider a training data set containing full friendship network and all the attributes (as given in D_1). The *3LPEx* algorithm takes the training data set as an input to generate sensitive rule set and then applied on the test data sets to secure users' privacy. We follow the same experimental set-up as described in Sect. 3.2.

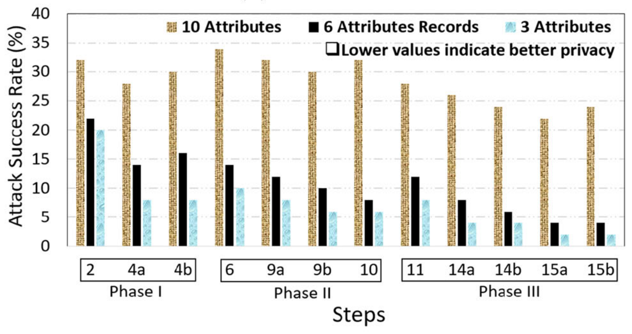
Once the protection has been given to the test data set users, we then launch the *attribute inference attack* by reducing the number of attributes in the training data sets. To do so, we first randomly select three nonclass attributes and then six non-class attributes. That is, we first build a classifier model by using a training data set that containing



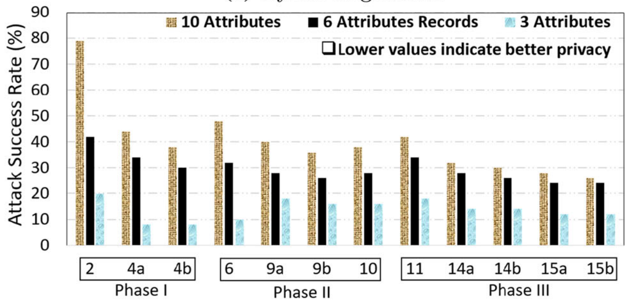
(a) Naïve Bayes classifier



(b) SVM classifier



(c) Sysfor Algorithm



(d) Random Forest Algorithm

Fig. 11 Different machine learning algorithms being used to breach privacy in the test data set

three non-class attributes and using the classifier model, we launch the attack to infer the sensitive attributes. Similarly, we repeat this procedure for six attributes.

We present the experimental results in Fig. 11 where x axis represents the steps and y axis represents *attack success rate*. We use different classifiers such as *NB*, *SVM*, *SysFor*, and *RF* that the attacker may use to infer sensitive information. We follow the same steps in our experiments as mentioned in Table 7.

The experimental results, presented in Fig. 11, indicate that as we reduce the number of nonclass attribute from 10 to 3, the *attack success rate* reduces significantly at all steps. Figure 11a presents the *attack success rate* when an attacker uses *NB* classifier to breach privacy of records protected by *3LPEx*. For *NB* classifier, presented in Fig. 11a, we can see a drop of around 30% in *attack success rate* at all steps (except Step 2). At Step 2, we can observe an *attack success rate* drop of about 65%. We can see a similar trend for *SVM*, *SysFor*, and *RF* as presented in Fig. 11b–d.

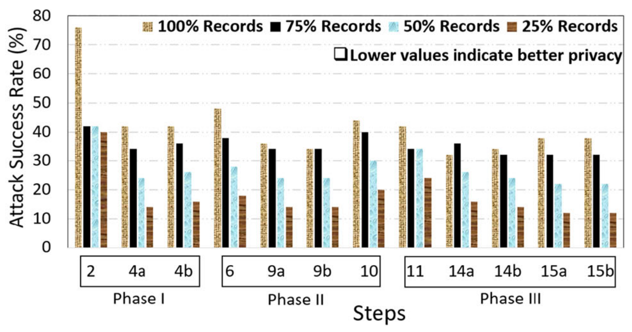
We also experimentally demonstrate the inference risk analysis by varying the number of records in the training data sets. We follow a distribution of 25%, 50%, and 75% randomly selected records from the full OSN network. That is, we first consider a sub-OSN containing 25% randomly selected records (out of the total records) and a friendship network containing those 25% user nodes only. From this sub-OSN network, we calculate the link attribute values and insert them in the training data set. We name this training data set as "Training Data Set containing 25% of Total records".

We build a classifier model by using the classifiers (such as *NB*, *SVM*, *SysFor*, and *RF*) on "Training Data Set containing 25% of Total records". We then apply classifier model on the secure test data set to infer the sensitive attributes. We follow this similar procedure for 50% and 75% distribution of the total records.

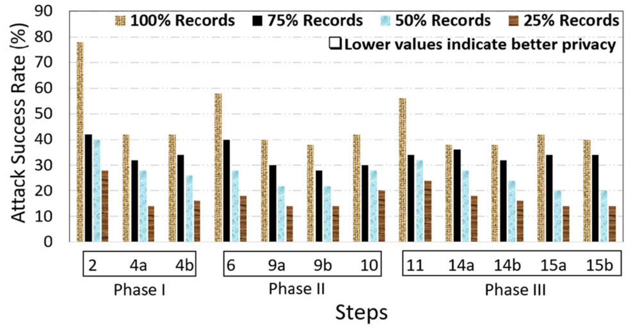
The experimental results are presented in Fig. 12 where the steps are the same as mentioned in Table 7. We can see from Fig. 12a that when an attacker uses the entire network (indicated as 100% records), then the *attack success rate* is higher at all steps. As we reduce the number of training data set records from 100 to 75%, this *attack success rate* drops from 76 to 42% at Step 2. This reduction of *attack success rate* goes further when 50% and 25% of the total records are used in the training data set (presented in Fig. 12). The above results confirm that the inference risk of a sensitive attribute value, after providing privacy by *3LPEx* using a full OSN information, can be reduced significantly if the attacker has a limited access to the OSN.

4.5 Time complexity analysis

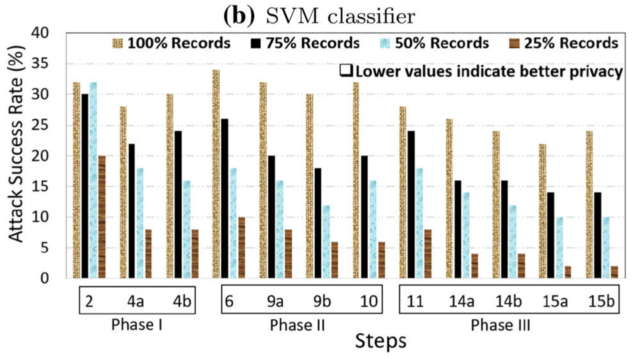
Throughout this study, we set the number of antecedents in sensitive rules those generated by *3LPEx* to 3 and sensitivity threshold to 1.006. We now analyse the time complexity of the *3LPEx* algorithm if these values are changed. We use D_1 data set for this experiment where *3LPEx* first generates a set of sensitive rules (from the training data set) where each rule containing two antecedents. *3LPEx* then



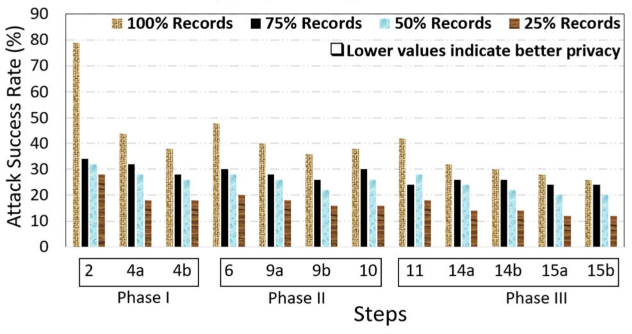
(a) Naïve Bayes classifier



(b) SVM classifier



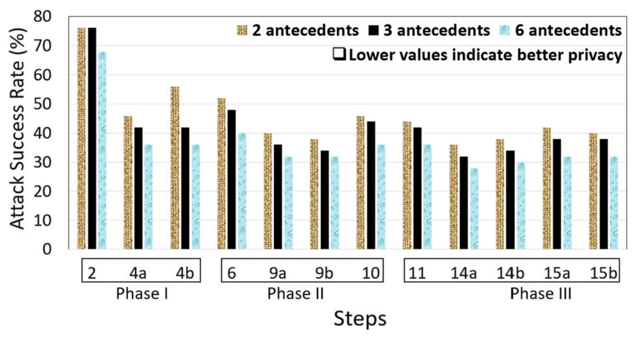
(c) Sysfor Algorithm



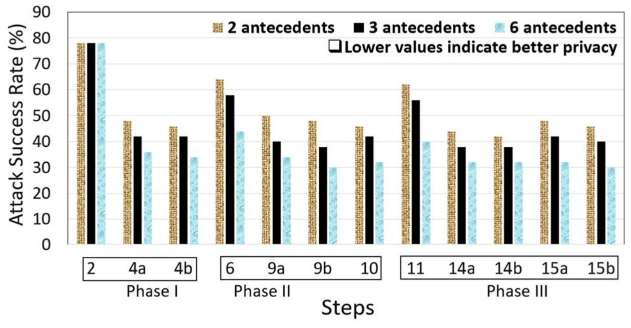
(d) Random Forest Algorithm

Fig. 12 Different machine learning algorithms being used to breach privacy in the test data set

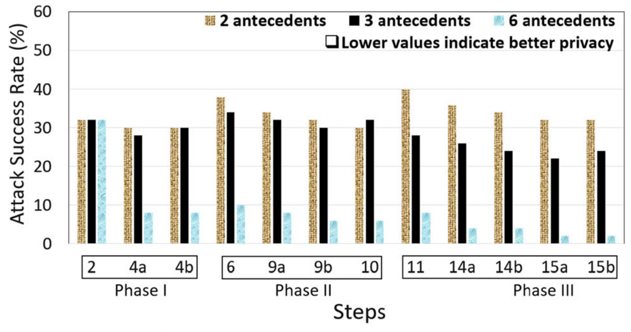
uses the generated sensitive rules to secure users' privacy on the test data set. We follow the same experimental setup as described in Sect. 3.2. Once the test data set is secured by *3LPEx*, we then apply *NB*, *SVM*, and *RF* on the test data sets to analyse the *attack success rate* and the required time to secure the users. We also analyse the data



(a) Naïve Bayes classifier



(b) SVM classifier

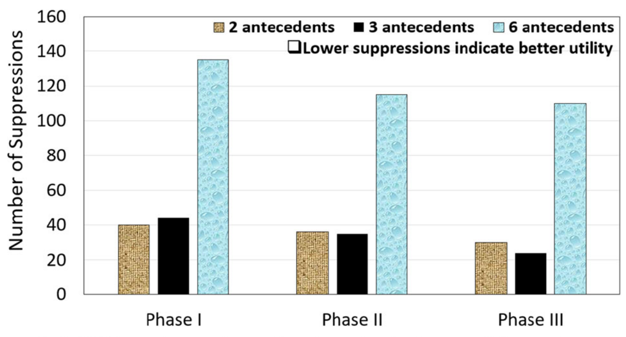


(c) Random Forest Algorithm

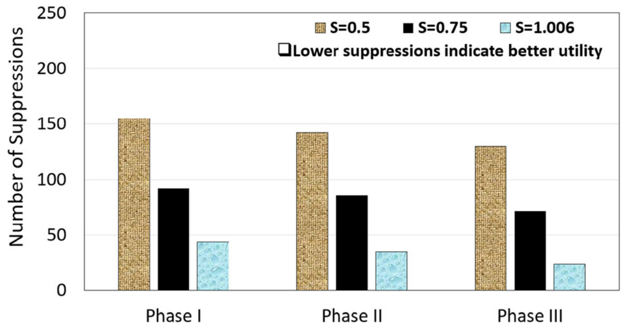
Fig. 13 Different machine learning algorithms being used to breach privacy in the test data set. We vary the number of antecedents in the sensitive rules generated by the *3LPEx* algorithm while sensitivity threshold is set to 1.006

utility of the secured test data sets. Similarly, we conduct experiments for the sensitive rules containing three antecedents, six antecedents and then compare the results with the two antecedents' results. While varying the number of antecedents in the sensitive rules, we fixed the sensitivity threshold to 1.006. We present the results in Figure 13.

From Fig. 13, we can see that the sensitive rules containing six antecedents provide better privacy by minimising the *attack success rate* for all the three classifiers. However, in terms of time consumption, it takes longer time to secure the test data set than two antecedents and three antecedents based sensitive rules as shown in Table 8. We also observe that the sensitive rules containing six antecedents reduce the data utility in the test data set more than the sensitive rules containing two antecedents and three antecedents as shown in Fig. 14a. On the other hand, *3LPEx* can reduce the *attack success rate* more if it uses the sensitive rules containing three antecedents rather than the sensitive rules containing two antecedents. Data utility is also high if *3LPEx* uses the sensitive rules containing 3 antecedents.



(a) Different number of antecedents in sensitive rules



(b) Different sensitivity thresholds considered while generating sensitive rules

Fig. 14 Comparison of the number of suppressed values for different number of antecedents in the sensitive rules and different sensitivity thresholds

We also conduct experiments by varying the sensitivity thresholds where the number of antecedents in sensitive rules is fixed to 3. We vary the sensitivity threshold from 1.006 to 0.75 and then 0.5. The results are presented in Figure 9 where we can see the lower sensitivity threshold, i.e. 0.5 can provide better privacy. This is because, in lower

Table 8 Summary of the required time for the experiments

	Number of antecedents			Sensitivity thresholds		
	2	3	6	0.5	0.75	1.006
Required time (in mins)	11	13	95	112	79	13

sensitivity threshold, *3LPEx* considers more rules as sensitive and provide privacy accordingly. However, at 0.5 sensitivity threshold, it takes longer time compared to the sensitivity threshold of 0.75 and 1.006 as shown in Table 8. On the other hand, at higher sensitivity threshold (i.e. 1.006) the secure test data set's data utility is higher than the other two sensitivity thresholds as shown in Fig. 14b. Therefore, these results justify the use of 1.006 sensitivity threshold and three antecedents in sensitive rules in our experiments.

5 Conclusion and future work

We addressed the *attribute inference attack* on online social networks (OSNs). We showed that an attacker can successfully infer users' private information with high probability by applying various data mining algorithms on the non-sensitive information disclosed by the users. We use a parameter, namely *attack success rate*, to measure the effectiveness of the privacy preserving techniques. We propose a new privacy preserving technique, namely *3LPEx*, that can protect users' multiple sensitive information (that the users consider to be sensitive). Our experimental results indicate that *3LPEx* outperforms the existing privacy preserving techniques by reducing the *attack success rate* even if the attacker applies different existing classifiers (rather than the one used by the privacy technique). Our experimental results also show that *3LPEx* can maintain a high data utility compared to the existing techniques by suppressing less attribute values while preserving privacy.

In this study, we have considered that only the user or a few others in the user's network are consumers of the protection techniques. If all friends in a user's friend list continually used and adopted the recommendations of the *3LPEx* algorithm, the calculation would be different and dynamic. Therefore, a future research could demonstrate the analysis of that calculation. Another future research effort should focus on generalisation of attribute values (rather than the suppression) suggested by the *3LPEx* algorithm.

Appendix 1: sensitive rules matching

The results for 5% and 15% difference of the domain range are presented in Tables 9 and 10, respectively.

Table 9 Sensitive rules matching (when maximum 5% numerical value mismatch in between two rules is considered as the same rules)

Row	Algorithms: SR (TR)	RF SR = 627	RS SR = 12	AdaBoost SR = 402	Bagging SR = 141	SysFor SR = 58	J48 SR = 2	FPA SR = 721
1	RF: SR = 627 (TR = 23,393)	627	1	2	7	18	0	17
2	RS: SR = 12 (TR = 2910)	5	12	0	7	0	0	8
3	AdaBoost: SR = 397 (TR = 20,562)	8	0	402	7	0	0	7
4	Bagging: SR= 140 (TR=8914)	12	2	0	141	0	0	3
5	SysFor: SR= 58 (TR = 2666)	2	2	0	12	58	2	1
6	J48 - 1 tree: SR = 2 (TR = 298)	0	0	0	1	8	2	0
7	FPA: SR = 718 (TR = 38,453)	23	3	5	7	1	1	721
8	RuleBank - exclude the rules generated by the selected algorithm: SR = 1963 (TR = 97,196)	60	10	7	56	24	2	37
9	Exhaustive: SR = 65,218 (TR = 5,63,830)	593	10	344	89	56	2	285

* *TR*= total number of rules, *SR* = number of sensitive rules, *FPA* = forest PA, *RF* = random forest, *RS* = random subspace

Table 10 Sensitive rules matching (when maximum 15% numerical value mismatch in between two rules is considered as the same rules)

Row	Algorithms: SR (TR)	RF SR = 627	RS SR = 12	AdaBoost SR = 402	Bagging SR = 141	SysFor SR = 58	J48 SR = 2	FPA SR = 721
1	RF: SR = 627 (TR = 23,393)	627	1	2	7	18	0	17
2	RS: SR = 12 (TR = 2910)	5	12	0	7	0	0	8
3	AdaBoost: SR = 402 (TR = 20,562)	8	0	402	7	0	0	7
4	Bagging: SR = 141 (TR = 8914)	12	2	0	141	0	0	3
5	SysFor: SR = 58 (TR = 2666)	2	2	0	12	58	2	1
6	J48 - 1 tree: SR = 2 (TR = 298)	0	0	0	1	11	2	0
7	FPA: SR = 721 (TR = 38,453)	23	3	5	7	1	1	721
8	RuleBank - exclude the rules generated by the selected algorithm: SR = 1963 (TR = 97,196)	62	10	7	56	27	2	37
9	Exhaustive: SR = 65,218 (TR = 5,63,830)	**595**	**10**	**347**	**90**	**56**	**2**	**286**

* *TR* = total number of rules, *SR* = number of sensitive rules, *FPA* = forest PA, *RF* = random forest, *RS* = random subspace

Appendix 2: the properties of the synthetic data set D_2

In Table 11, we present the list of attributes and their corresponding attribute values used to generate the synthetic data set D_2. Each record of the data set D_2 has been created using a logic rule, and they are as follows:

```
  Residence = A value is generated using
the
  following probability distribution:
  30% probability for each of Brisbane,
Sydney,
  Melbourne and 10% for Bathurst;
  if(Residence = Bathurst) {
    Age range = A value is generated using
  the following probability distribution:
  40% probability for the age 50+, 20%
  probability for each of the age range
48-57
  and 38-47, 10% probability for each of
the
  age range 28-37 and 18-27;
    Friends on OSN = A value is generated
  using the following probability
distribution:
  60% probability for low, 25% probability
for
  medium and 15% probability for high;
      if(Age range = 18-27) {
  Relationship status = 45% probability
for
```

Table 11 List of attributes and their corresponding values utilised in synthetic data set $\mathcal{D}_2$

Types of information	Attribute name	Attribute values
Activities on OSN	Number of friends on OSN	Low (1–100), medium (101–500), high (501 and above)
	Number of pages followed on OSN	Low (0–9), medium (10–19), high (20 and above)
	Number of uploaded photos per week	Low (0–5), medium (6–10), high (11 and above)
	Number of comments made on other users' contents per week	Low (0–9), medium (10–19), high (20 and above)
Personal information	Age range	18–27, 28–37, 38–47, 48–57, 58 and above
	Current city of residence	Sydney, Melbourne, Brisbane, Bathurst
	Relationship status	Single, in-a-relationship, married, not mentioned
Sensitive information	Political view	Labour, Liberal, Green
	Religion	Christianity, Islam, Others, Buddhism, no religion
	Profession	Government employee, entrepreneur, salesman, retired person, student

```
  single, 30% probability for in-a-relationship,
  23% probability for not-mentioned and 2%
  probability for married; }
      else if(Age range = 28-37 or 38-47){
  Relationship status = A value is generated
  using the following probability distribution:
  45% probability for married, 30% probability
  for in-a-relationship, 15% probability for
  single and 10% probability for not-mentioned;
  }
      else {
  Relationship status = 55% probability for
  married, 35% probability for in-a-relatio-
  nship, 5% probability for single and 5%
  probability for not-mentioned;}
      if(Number of Friends on OSN = low) {
  Number of uploaded photos or comments or
  pages followed = A value is generated using
  the following probability distribution:
  65% probability for low, 25% probability
for medium and 10% probability for high.}
      if(Number of Friends on OSN = medium)
  { Number of uploaded photos or comments or
  pages followed = A value is generated
using the following probability distribution:
  30% probability for low, 45% probability
  for medium and 25% probability for high.}
      else {
  Number of uploaded photos or comments or
  pages followed = A value is generated using
  the following probability distribution: 55%
  probability for high, 40% probability for
  medium and 5% probability for low.}
  } // End of if(Residence = Bathurst)
  else(Residence = Sydney or Melbourne or
  Brisbane) {
    Age range = A value is generated using
  the following probability distribution: 42%
  probability for the age range 18-27, 25%
probability for the age range 28-37,
  15% probability for each of the age range 38-47,
  48-57 and 5% probability for the age 58
  and above;
    Friends on OSN = A value is generated
  using the following probability distribution:
  50% probability for high, 35% probability
  for medium and 15% probability for low;
      if(Age range = 18-27) {
  Relationship status = A value is generated
  using the following probability distribution:
  55% probability for single, 25% probability
  for in-a-relationship, 18% probability
  for not-mentioned and 2%
  probability for married;
      Number of uploaded photos or comments
  or pages followed = A value is generated
  using the following probability distribution:
  85% probability for high, 10% probability
  for medium and 5% probability for low.}
      else if(Age range = 28-37 or 38-47 or
  48-57) {
  Relationship status = A value is generated
  using the following probability distribution:
  35% probability for in-a-relationship, 30%
  probability for married, 15% probability
  for single and 10% probability for not-
  mentioned;
      Number of uploaded photos or comments
  or pages followed = A value is generated
  using the following probability distribution:
  40% probability for medium, 30% probability
  for each low and high.}
      else {
      Relationship status = A value is
  generated using the following probability
  distribution: 55% probability for married,
  30% probability for in-a-relationship, 10%
  probability for single and 5% probability
  for not-mentioned;
      Number of uploaded photos or comments
  or pages followed = A value is generated
  using the following probability distribution:
  70% probability for low, 28% probability for
  medium and 2% probability for high.}
  } // End of if(Residence = Sydney or
  Melbourne or Brisbane)
  if(Age range = 48-57 or 58 and over; and
  Relationship Status = Married)
  Religion = Christian;
  else if (Age range = 48-57 or 58 and over;
  and Relationship Status = Single or
  in-a-relationship or Not Mentioned)
  Religion = No religion;
  else if (Age range = 18-27 and
  Relationship Status = married)
  Religion = 25% Probability for each of
  Christian, Islam,
  Buddhism and others;
  else
  Religion = 44% Probability of being
  No-religion, 50% probability of being Chri-
  stian, 2% probability for each of Islam,
  Buddhism, Others;
  if(Age range = 18-27 and Relationship
  Status = single and Number of friends and
  uploaded photos or comments or pages follow-
  ed = medium or high) Profession = student;
  if(Age range = 28-37 or 38-47 and Relation-
  ship Status = in-a-relationship and Number of
  friends and uploaded photos or comments or
  pages followed = medium or low) Profession
  = salesman;
  else if (Age range = 58 and over; and
  Relationship Status = Married or
  in-a-relationship) Profession = Retired_person;
  else if (Age range = 28-37 or 38-47 or
  48-57 and Relationship Status = Married or
  in-a-relationship)
  Profession = Government employee;
  else
  Profession = entrepreneur;
  if(Number of uploaded photos or comments or
  pages followed = medium or High and Relation-
  ship Status = single or in-a-relation-ship or
  not-mentioned)
  Political view = Green;
  if(Number of uploaded photos or comments or
  pages followed = low and Relationship Status =
  married or in-a-relationship and Number of
  friends = medium or low
  Political view = Labour;
  else
  Political view = Liberal;
```

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Gong NZ, Talwalkar A, Mackey L, Huang L, Shin ECR, Stefanov E, Shi ER, Song D (2014) Joint link prediction and attribute inference using a social-attribute network. ACM Trans Intell Syst Technol 5(2):27:1-27:20. https://doi.org/10.1145/2594455
2. Chen W, Cai R, Hao Z, Yuan C, Xie F (2020) Mining hidden non-redundant causal relationships in online social networks. Neural Comput Appl 32:6913–6923
3. Wang D, Chen Y (2019) A neural computing approach to the construction of information credibility assessments for online social networks. Neural Comput App 31(1):259–275
4. Mulders D, De Bodt C, Bjelland J, Pentland A, Verleysen M, de Montjoye YA (2020) Inference of node attributes from social network assortativity. Neural Comput Appl 32:18023–18043
5. Heussner K (2009) 'gaydar' n facebook: can your friends reveal sexual orientation? ABC News. http://abcnews.go.comTechnologygaydar-facebook-friendsstor. Accessed 5 Apr 2021
6. Johnson C (2009) Project gaydar. The Boston Globe 20
7. Kosinski M, Stillwell D, Graepel T (2013) Private traits and attributes are predictable from digital records of human behavior. Proc Natl Acad Sci 110(15):5802–5805. https://doi.org/10.1073/pnas.1218772110
8. Ryu E, Rong Y, Li J, Machanavajjhala A (2013) Curso: protect yourself from curse of attribute inference: a social network

privacy-analyzer. In: Proceedings of the ACM SIGMOD workshop on databases and social networks. ACM, pp 13–18

9. Mislove A, Viswanath B, Gummadi KP, Druschel P (2010) You are who you know: inferring user profiles in online social networks. In: Proceedings of the third ACM international conference on web search and data mining. ACM, pp 251–260
10. Wu X, Kumar V, Ross Quinlan J, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou ZH, Steinbach M, Hand DJ, Steinberg D (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
11. Hong T, Wang Z, Luo X, Zhang W (2020) State-of-the-art on research and applications of machine learning in the building life cycle. Energy Build 212:109831
12. Fotovatikhah F, Herrera M, Shamshirband S, Chau KW, Faizollahzadeh Ardabili S, Piran MJ (2018) Survey of computational intelligence as basis to big flood management: challenges, research directions and future work. Eng Appl Comput Fluid Mech 12(1):411–437
13. Chau KW (2017) Use of meta-heuristic techniques in rainfall-runoff modelling. Water 9(3):186. https://doi.org/10.3390/w9030186
14. Shamshirband S, Rabczuk T, Chau KW (2019) A survey of deep learning techniques: application in wind and solar energy resources. IEEE Access 7:164650–164666
15. Najafi B, Faizollahzadeh Ardabili S, Shamshirband S, Chau KW, Rabczuk T (2018) Application of anns, anfis and rsm to estimating and optimizing the parameters that affect the yield and cost of biodiesel production. Eng Appl Comput Fluid Mech 12(1):611–624
16. Faizollahzadeh Ardabili S, Najafi B, Shamshirband S, Minaei Bidgoli B, Deo RC, Chau KW (2018) Computational intelligence approach for modeling hydrogen production: a review. Eng Appl Comput Fluid Mech 12(1):438–458
17. Moazenzadeh R, Mohammadi B, Shamshirband S, Chau KW (2018) Coupling a firefly algorithm with support vector regression to predict evaporation in northern Iran. Eng Appl Comput Fluid Mech 12(1):584–597
18. Shekhar H, Seal S, Kedia S, Guha A (2020) Survey on applications of machine learning in the field of computer vision. In: Mandal JK, Bhattacharya D (eds) Emerging technology in modelling and graphics. Springer Singapore, Singapore, pp 667–678
19. Cui L, Yang S, Chen F, Ming Z, Lu N, Qin J (2018) A survey on application of machine learning for internet of things. Int J Mach Learn Cybern 9(8):1399–1417. https://doi.org/10.1007/s13042-018-0834-5
20. La Fond T, Neville J (2010) Randomization tests for distinguishing social influence and homophily effects. In: Proceedings of the 19th international conference on World Wide Web. WWW '10, ACM, New York, pp 601–610
21. Kossinets G, Watts D (2006) Empirical analysis of an evolving social network. Science 311(5757):88–90
22. Kumar R, Novak J, Raghavan P, Tomkins A (2004) Structure and evolution of blogspace. Commun ACM 47(12):35–39
23. Kim M, Leskovec J (2011) Modeling social networks with node attributes using the multiplicative attribute graph model. In: Proceedings of the twenty-seventh conference on uncertainty in artificial intelligence. UAI'11, AUAI Press, Arlington, pp 400–409. http://dl.acm.org/citation.cfm?id=3020548.3020595. Accessed 5 Apr 2021
24. Adamic LA, Adar E (2003) Friends and neighbors on the web. Soc Netw 25(3):211–230
25. Al-Saggaf Y, Islam MZ (2012) Privacy in social network sites (sns): the threats from data mining. Ethical Space Int J Commun 9(4):32–40
26. Al-Saggaf Y, Islam MZ (2015) Data mining and privacy of social network sites' users: implications of the data mining problem. Sci Eng Ethics 21(4):941–966
27. Guha S, Tang K, Francis P (2008) NOYB: Privacy in online social networks. In: Proceedings of the first workshop on online social networks. ACM, pp 49–54
28. Estivill-Castro V, Hough P, Islam MZ (2014) Empowering users of social networks to assess their privacy risks. In: 2014 IEEE international conference on big data (big data). IEEE, pp 644–649
29. Islam Z, Giggins H (2011) Knowledge discovery through SysFor: a systematically developed forest of multiple decision trees. In: Proceedings of the ninth Australasian data mining conference, vol 121. Australian Computer Society, Inc, pp 195–204
30. Estivill-Castro V, Nettleton DF (2015) Can on-line social network users trust that what they designated as confidential data remains so? In: TrustcomBigDataSEISPA, 2015 IEEE. vol 1. IEEE, pp 966–973
31. Estivill-Castro V, Nettleton DF (2015) Privacy tips: would it be ever possible to empower online social-network users to control the confidentiality of their data? In: Proceedings of the 2015 IEEEACM international conference on advances in social networks analysis and mining 2015. ACM, pp 1449–1456
32. Heatherly R, Kantarcioglu M, Thuraisingham B (2013) Preventing private information inference attacks on social networks. IEEE Trans Knowl Data Eng 25(8):1849–1862
33. Reza KJ, Islam MZ, Estivill-Castro V (2017) 3lp: three layers of protection for individual privacy in facebook. In: IFIP international conference on ICT systems security and privacy protection. Springer, pp 108–123
34. Reza KJ, Islam MZ, Estivill-Castro V (2019) Privacy preservation of social network users against attribute inference attacks via malicious data mining. In: 5th international conference on information systems security and privacy: ICISSP 2019. Scitepress, pp 412–420
35. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
36. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
37. Barandiaran I (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):1–22
38. Freund Y, Schapire RE et al (1996) Experiments with a new boosting algorithm. In: ICML, vol 96. Citeseer, pp 148–156
39. Quinlan JR (2014) C4. 5: programs for machine learning. Elsevier, Amsterdam
40. Adnan MN, Islam MZ (2017) Forest pa: constructing a decision forest by penalizing attributes used in previous trees. Expert Syst Appl 89:389–403
41. Nettleton DF (2015) Generating synthetic online social network graph data and topologies. In: 3rd workshop on graph-based technologies and applications (graph-TA), UPC, Barcelona, Spain
42. Reza KJ, Islam MZ, Estivill-Castro V (2017) Social media users' privacy against malicious data miners. In: 12th international conference on intelligent systems and knowledge engineering (ISKE), 2017. IEEE, pp 1–8
43. Gürses G, Berendt B (2010) The social web and privacy: practices, reciprocity and conflict detection in social networks. In: Privacy-aware knowledge discovery, novel applications and new techniques. CRC Press, pp 395–429
44. Aho A, Hopcroft J, Ullman J (1974) The design and analysis of computer algorithms. Addison-Wesley Publishing Co., Reading
45. Islam MZ (2007) Privacy preservation in data mining through noise addition. University of Newcastle, Brisbane
46. Australia YB (2008) Australian bureau of statistics. Canberra, Australia, p 161

47. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor 11(1):10–18
48. John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the eleventh conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc, pp 338–345
49. Platt J (1998) Sequential minimal optimization: a fast algorithm for training support vector machines. https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/
50. Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK (2001) Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput 13(3):637–649
51. le Cessie S, van Houwelingen J (1992) Ridge estimators in logistic regression. Appl Stat 41(1):191–201
52. Quinlan JR, et al (1996) Bagging, boosting, and C4. 5. In: AAAI/IAAI, vol 1. pp 725–730