Computational Social Networks

RESEARCH Open Access

A privacy-preserving framework for ranked retrieval model

Tong Yan[1*†], Yunpeng Gao[1†] and Nan Zhang[2]

*Correspondence:
tongyan@gwmail.gwu.edu
†Tong Yan and Yunpeng Gao contributed equally to this research
[1] School of Engineering and Applied Science, George Washington University, 2121 I St NW, Washington, DC 20052, USA
Full list of author information is available at the end of the article

Abstract

In this paper, we address privacy issues related to ranked retrieval model in web databases, each of which takes private attributes as part of input in the ranking function. Many web databases keep private attributes invisible to public and believe that the adversary is unable to reveal the private attribute values from query results. However, prior research (Rahman et al. in Proc VLDB Endow 8:1106–17, 2015) studied the problem of rank-based inference of private attributes over web databases. They found that one can infer the value of private attributes of a victim tuple by issuing well-designed queries through a top-*k* query interface. To address the privacy issue, in this paper, we propose a novel privacy-preserving framework. Our framework protects private attributes' privacy not only under inference attacks but also under arbitrary attack methods. In particular, we classify adversaries into two widely existing categories: domain-ignorant and domain-expert adversaries. Then, we develop equivalent set with virtual tuples (ESVT) for domain-ignorant adversaries and equivalent set with true tuples (ESTT) for domain-expert adversaries. The ESVT and the ESTT are the primary parts of our privacy-preserving framework. To evaluate the performance, we define a measurement of privacy guarantee for private attributes and measurements for utility loss. We prove that both ESVT and ESTT achieve the privacy guarantee. We also develop heuristic algorithms for ESVT and ESTT, respectively, under the consideration of minimizing utility loss. We demonstrate the effectiveness of our techniques through theoretical analysis and extensive experiments over real-world dataset.

Keywords: Rank inference, Privacy and security, Database management

Introduction

In this paper, we address privacy issues related to ranked retrieval model in web databases, each of which takes private attributes as part of input in the ranking function. Many web databases have both public and private attributes which serve different purposes. Websites, which are the owners of web databases, show the public attributes but keep private attributes invisible to the public. For example, social network websites provide privacy settings which allow users to control the visibility of user profiles by hiding certain attribute values from public view. In order to maximize the protection effect, these websites also hide private attributes in query results so that the public can only access attributes that are set to public by users. Many websites believe that the adversary is unable to reveal the private attribute values from query results though private attributes have been taken as part of input in the ranking function. They declare that the private attributes are well protected. Intuitively, users indeed cannot view others' private attribute values and their own private attribute values are hidden from public view. Users trust these websites because they believe that "what you see is what you get," and are persuaded to input sensitive personal information as private attributes to databases. However, the investigation in [1] proved that though the values of private attributes could be hidden from public view, they still can be inferred from the ranked results.

Motivation

According to the study by Rahman et al., one can infer the values of private attributes of a victim tuple by issuing well-designed queries through a top-k query interface. Rahman et al. discovered that under the premise that the ranking function satisfies both monotonicity condition and additively condition [1], the problem of excluding a value θ of a private attribute B_1 in its domain can be reduced to finding a pair of differential queries q_θ and q'_θ which satisfy the following properties: (1) they hold the same predicate on all attributes but B_1, (2) $q_\theta[B_1] = \theta$ while $q'_\theta[B_1] \neq \theta$, and (3) q_θ returns the victim tuple v while q'_θ does not. An adversary therefore is able to infer the value of a private attribute by excluding all but one value in the domain if the domain is discrete and finite. Most research done to date has focused on the development and evaluation of effective ranking functions of the ranked retrieval model. Thus, the discovery of this unprecedented privacy risk attracted our interest because it had not been adequately addressed by any existing defense techniques so far.

Before introducing our technical results, we would like to first review existing privacy-preserving techniques. The most straightforward method for privacy preserving is completely removing all private attributes from ranking functions. In this case, the query results do not contain any information of private attributes. However, in practical, the benefit of adding such private attributes as *inputs* is obvious: higher effectiveness (e.g., dating sites may take sensitive private attribute race and religion into consideration during matching their users). Therefore, in this paper, we focus on preserving private attributes as well as maintaining the utility of ranking functions.

There has been extensive work on privacy preserving in databases using data perturbation in which tuples in the databases are modified to protect sensitive information. Multiplicative noise [2] masks continuous data by adding variance to the original data. Micro-aggregation [3] groups individual records into small aggregations and replaces the values of each record with the average value of each group. However, variance or averages cannot be computed for categorical data, especially non-numeric data. Therefore, multiplicative noise and micro-aggregation cannot solve the problem where datasets contain non-numeric categorical data. Rule hiding [4] proposed a strategy that reduces the confidence of rules that specify how significant they are, given the inference rules known. Categorical data perturbation [5] proposed a guarantee of privacy by limiting the posterior probability of perturbed dataset, which suffers from high utility loss. Data swapping [6] swaps the values of sensitive records with non-sensitive records in order to protect privacy of the former while maintain summary statistic of the dataset. However, in our setting, our goal is to preserve privacy of private attributes of all records. Data shuffling [7] preserves privacy of numeric attributes by swapping their values with each other. This method, however, is limited by the distribution of datasets and subsequent utility loss.

Condensation is another approach of privacy-preservation data mining. Aggarwal and Philip [3] treat tuples as data points in a multidimensional space and try to cluster all data points in a database. However, this method requires a distance function for tuples in a database. Therefore, attributes have to be numerical. Furthermore, we need prior knowledge to scale and standardize different attributes on different axes.

Suppression and generalization on databases have a number of studies [8–10]. This paper [8] analyzed the risk of releasing data without the consideration of re-identification by linking. Sweeney [8] provided the k-anonymity protection model. Sweeney et al. [11–13] also gave real-world systems which adopted the k-anonymity protection. Even the following research of [9, 10] proposed l-diversity and t-closeness, we cannot ignore the fact that in a ranked retrieval model, namely, we take into account the private attribute values but we do not return the entire dataset regarding a single query. Nevertheless, it is possible to directly apply the suppression and generalization on the dataset, but none of the approaches considered the optimization on ranked retrieval models.

Another widely studied approach in the academic area is the differential privacy [14]. It protected the privacy of individual when releasing statistical information of databases by adding noise. Dwork et al. [15] then studied the algorithmic foundations of differential privacy. Moreover, Wasserman and Zhou [16] introduced a statistical framework for differential privacy. However, in practice, the differential privacy sometimes is not the first choice for the data owner because of its recondite.

An alternate approach is query auditing [17]. Query auditing is the process of examining past actions to check whether they were in conformance with official policies. In a specific online database system, query auditing is the process of examining user's queries answered in the past and denying queries from the same user that could potentially cause a breach of privacy. However, it is inadequate to assume that an adversary is limited to only one account. In fact, many online databases impose no or loose restrictions on the number of accounts a user can create and the number of queries an account can issue every day. A recent study [1] shows that privacy can be compromised without breaking query auditing. As mentioned by [1], an adversary is able to infer private attribute values of any user in eHarmony[1] with freely created accounts. In fact, the adversary conducts only two kinds of operations: creating accounts and issuing a number of queries, whose accesses are open to the public.

The limitation of any defense techniques for specific attacking methods (e.g., [18]) is that the defense technique eventually loses its effectiveness if the adversary changes the way of attack. In this paper, we remove the constraints on the methods of attack adopted by the adversary. In our framework, we allow a complete adversary that is able to compromise private attribute values through the top-k results in arbitrary attacking methods. We also allow a harmless user do the same querying operations to acquire top-k results as usual. Since attacking techniques can no longer be used to differentiate adversaries, we categorize adversaries by their prior knowledge—the domain of private attributes they are going to infer.

[1] eHarmony is an online dating website. It was launched on August 22, 2000, and is based in Los Angeles, California.

Overview of technical results

In this paper, we propose a formal definition of adversaries. We divide adversaries into two categories:

1. The first category of adversaries are those who have no prior knowledge of attributes. Thus, adversaries cannot validate the authenticity of any tuple. We refer to this class of adversaries as *domain-ignorant adversaries*.
2. The second category of adversaries are those who have prior knowledge of non-trivial attributes. This category of adversaries is able to validate the authenticity of an attribute value. For example, this category of adversaries can at least exclude a value of a target's private attribute if the value cannot coexist with target's public attributes according to the prior knowledge. We refer to this category of adversaries as *domain-expert adversaries*.

In this paper, we will focus on both categories of adversaries. We believe that classifying adversaries based on the extent of prior knowledge rather than their techniques of attacking is better for the following reasons: first, it is relatively feasible to determine whether the adversary has prior knowledge. Second, in terms of data owner, in practice it is unrealistic to predict the techniques that may be used by the adversaries. Moreover, we cannot deny the fact that adversaries will launch multiple attacks at the same time in order to maximize the effect of their attacks.

In this study, we focus on the rank inference problem, in which the adversary infers the values of private attributes by issuing *k*NN queries through a top-*k* query interface. The rank inference problem is privacy leakage discussed in [1]. Rahman el al. proved that a ranking function designed without taking privacy issues into consideration may lead to privacy leakage of private attributes, even though private attribute values are not exposed to the public. Our goal in this paper is not to design solutions for an individual rank inference problem. Instead, we are using the rank inference problem as an example to illustrate our methodology to deal with adversaries without technique restrictions.

For *domain-ignorant adversaries*, as we will show in this paper, we focus on preserving privacy by constructing virtual tuples. Because the ranked results will be modified by ESVT, a trade-off between privacy protection and utility loss will be fully discussed. As such, we propose a framework in "Framework with virtual tuples" section, which provides our privacy guarantee while maximizing data utility. We prove that framework strongly protects privacy of victim tuples against all *domain-ignorant adversaries*. We further prove that the optimal algorithm implementing our framework is a NP-complete problem. To evaluate our framework, we develop a heuristic algorithm in "Framework with virtual tuples" section, which meets our requirement of privacy with the trade-off of a little utility loss.

For *domain-expert adversaries*, in this paper we consider the construction of a robust framework that defends against arbitrary attacks by *domain-expert adversaries* while minimizing the utility loss. As such, we propose *ESTT* algorithm of constructing privacy-preserving framework. Then we evaluate its privacy guarantee, and prove that the privacy is preserved. Then we consider an optimal solution for constructing the framework with the minimal number of *Data Obfuscations*. We prove that the optimal solution is an NP-complete problem even under a relatively simple structure. We thereby propose a heuristic implementation. The experiment results show that we successfully well protected the privacy of private attributes in ranked retrieval model. Our work is unprecedented because our privacy-preserving framework significantly increases the privacy preservation under any the attacking methods.

The rest of the paper is organized as follows. In "Framework" section, we introduce the framework. In "Adversary model" section, we introduce our adversary model. In "Framework with virtual tuples" section, we present an implementation of the Framework with virtual tuples, along with privacy and utility loss analysis. In "Framework with true tuples" section, we present an implementation of the framework with true tuples, along with privacy and utility loss analysis. We present experiment results in "Experimental results" section, followed by final remarks in "Final remarks" section.

Framework

Ranked retrieval model

As discussed in the introduction, many web databases store both public and private attributes of users. In recent years, a large number of databases have been adopting the ranked retrieval model. Within proper ranking function, the system returns tuples that best satisfy the query (e.g., returns top-k-tuples). Consider an n-tuple (i.e., n-user) database D with a total of $m + m'$ attributes, including m public attributes $A_1, \ldots, A_m$ and m' private attributes $B_1, \ldots, B_{m'}$. Let V_i^A and V_j^B be the attribute domain (i.e., set of all attribute values) for A_i and B_j, respectively. We use $t[Ai]$ (resp. $t[Bj]$) to denote the value of a tuple $t \in D$ on attributes A_i (resp. B_j). For the purpose of this paper, we assume there is no duplicate tuple in the database.

Our ranked retrieval model is formalized as follows. Given a top-k query q, the model is able to compute a score $s(t|q)$ based on a predetermined ranking function for each tuple $t \in D$, and returns the k-tuples with the highest $s(t|q)$. In this paper, we consider a linear ranking function. The linear ranking function can be defined as

$$s(t|q) = \sum_{i=1}^{m} \cdot w_i^A \cdot \rho(q[A_i], t[A_i]) + \sum_{j=1}^{m'} \cdot w_j^B \cdot \rho(q[B_j], t[B_j]), \tag{1}$$

where w_i^A, $w_j^B \in (0, 1]$ are the ranking weights for attributes A_i and B_j, respectively. In this paper, we consider the case that attributes A_i and B_i are categorical, and $\rho(q[A_i], t[A_i]) = 1$ if $q[A_i] = t[A_i]$ ($q[B_j] = t[B_j]$, respectively), or 0 if $q[A_i] \neq t[A_i]$ ($q[B_j] \neq t[B_j]$, respectively). ρ can be easily extended to numerical attribute cases in which $\rho(q[A_i],\ t[A_i]) = |q[A_i] - t[A_i]|$ ($|q[B_j] - t[B_j]|$, respectively). We note again that the ranking function follows the monotonicity and additivity properties.

Problem statement

Ideally, we want to keep adversaries from retrieving the private attributes' values of victim tuple v from ranking results. In practice, adversaries may retrieve the possible private attribute' values through arbitrary attacks. However, if adversaries cannot 100% determine the values of v's private attributes, we can conclude that privacy of private attributes is well preserved. We define the *privacy* of $v[B_j]$ as $P_{v[B_j]}$, which is the possibility that an arbitrary adversary fails to infer the value of $v[B_j]$. Therefore, $P_{v[B_j]} = 0$ represents the worst case where privacy of $v[B_j]$ is compromised, while $P_{v[B_j]} = 1$ represents an ideal case where an adversary is unable to infer the authentic value of $v[B_j]$.

In this paper, the objective of privacy preserving is to protect all private attributes of any tuple t. Therefore, we define the privacy-preserving problem as

$$\forall t \in D, j \in \{1, \ldots, m'\}, \quad P_{t[B_j]} \geq \epsilon, \tag{2}$$

where ϵ is a constant that we present as a privacy guarantee.

Privacy-preserving framework

In the framework, we want to keep the adversary from identifying a victim tuple v from ranked results. We find that it can be achieved by finding at least another tuple t (where $t \neq v$) which has the same ranking score as v for all queries. In the query results, the rank of v is equal to the rank of t for an arbitrary $q \in Q$, where Q is the set of all possible queries:

$$\forall q \in Q, \mathrm{Rank}(v|q) = \mathrm{Rank}(t|q).$$

We now prove that v and t are indistinguishable if v and t are *equivalent*. Given two tuples v and t, where $v[A_i] = t[A_i]$, and two databases D_1 and D_2, where $D_2 = (D_1 - \{v\} \cup \{t\})$. Consider v and t are equivalent. By simply issuing queries and examining the ranked results, the adversary cannot distinguish D_1 from D_2. Therefore, v and t are indistinguishable.

As such, we introduce the construction of an *Equivalent Set*, which is to put the victim tuple t into a set in which all tuples are *equivalent* in any ranked results. In this paper, we name this set as the *Equivalent Set* and denote it as E_t. We define *Equivalent Set* as follows:

Definition 1 In a set $E_v = \{v, t_1, \ldots, t_k\}$, where $v[A_i] = t_1[A_1] = \cdots = t_k[A_i]$, and $v, t_1, \ldots, t_k$ are equivalent, we call the set E_v as the *Equivalent Set*, and the domain of B_j as C_j^B.

To achieve *privacy* of $v[B_j]$, since any technique based on ranked results cannot distinguish $v, t_1, \ldots, t_k$ in E_v, the privacy guarantee of $v[B_j]$ is

$$P_{v[B_j]} = \left(1 - \frac{1}{|C_j^B|}\right) * 100\%. \tag{3}$$

To achieve our guarantee of privacy preservation defined in (2), we have to make sure that (a) for each $t \in D$, t is included by one and only one equivalent set E_t and (b) the privacy guarantee defined in (3) is valid for any $j \in \{1, \ldots, m'\}$. Note that for the condition *a*, if t is not included by any E_t, then obviously t is not protected. Our privacy guarantee cannot be achieved. Also, if t appears in more than one E_t, e.g., $t \in E_t$ and $t \in E_t'$, then according to Definition 1, all elements in E_t and E_t' are equivalent and, thus, E_t and E_t' should be merged into a new E_t''.

In the privacy-preserving framework, except the victim tuple, the other tuples in the equivalent set could be either virtual tuples or true tuples. We will give details of two implementations which construct equivalent set with virtual tuples and with true tuples.

Utility loss measurement

To quantify utility loss provided by a method, we use a measurement based on the distance of ranked results before and after applying our privacy-preserving frameworks, respectively. Given a database D and a set of all possible queries Q, we define utility loss as follows:

$$U = \sum_{t \in D} \sum_{q \in Q} |\text{Rank}(t|q) - \text{Rank}'(t|q)|, \tag{4}$$

where $\text{Rank}(t|q)$, $\text{Rank}'(t|q)$ refer to the ranks of tuple t given query q before and after applying our privacy-preserving frameworks, respectively.

Adversary model

We mentioned that the adversary wants to compromise private attribute values of a victim tuple v through arbitrary attacks. Without loss of generality, we make the assumption that an adversary has knowledge about the ranking function and all the public attributes. Furthermore, we assume that the adversary is able to issue queries and insert tuples with specified values to the database.

As we discussed in "Motivation" section, prior knowledge of attributes will definitely help the adversary to perform a more effective attack.

For example, when $\theta(V_j^B)$ is a uniform distribution, the prior knowledge of adversaries can be ignored. When $\theta(V_j^B)$ is a non-trivial prior distribution for V_j^B, the prior knowledge of adversaries possess can potentially be used to launch more effective attacks, e.g., for a victim tuple t, adversaries prefer to choose some $t[B_j]$ based on the distribution.

Thus, we partition the adversary into two categories: (1) The adversary has no prior knowledge of attributes; (2) the adversary has prior knowledge of attributes.

We assume that the objective of the adversary is to maximize the following g_A value

$$g_A = \Pr(v[B_j] = a|\theta(V_j^B)), \tag{5}$$

where $\theta(V_j^B)$ stands for the adversary's prior knowledge of V_j^B. The prior knowledge may have diverse forms, e.g., correlations between $v[A_i]$ and $v[B_j]$. In this case, the adversary can infer $v[B_j]$ by knowing $v[A_i]$, where $v[A_i]$ is known in public. Here we assume that the adversary is able to validate the authenticity of $v[B_j]$ based on $\theta(V_j^B)$. This model can describe various kinds of adversaries because the adversary can possess the $|V_j^B|$, and thus can assume arbitrary value in $|V_j^B|$ be the true $v[B_j]$. The adversary can validate this arbitrary value by $\theta(V_j^B)$.

Definition 2 An adversary is domain-ignorant if the adversary has no prior knowledge of attributes. An adversary is domain-expert if the adversary has prior knowledge of non-trivial attributes.

As we discussed in "Privacy-preserving framework" section, a tuple in equivalent set could be either virtual or true tuple. Based on the classification of adversaries, we propose two implementations of privacy-preserving framework in this paper:

- The implementation with virtual tuples. Except v, other tuples in E_v are virtual. We demonstrate details in "Framework with virtual tuples" section.
- The implementation with true tuples. Include v, every tuple in E_v is true. We demonstrate details in "Framework with true tuples" section.

Framework with virtual tuples

Design

In this section, we introduce the construction of ESVT, i.e., equivalent sets constructed with virtual tuples which are generated by our framework instead of collecting from real data. In order to ensure the effectiveness of the ranked retrieval model, we do not show virtual tuples in ranked results.

As we proved in "Privacy-preserving framework" section, an adversary cannot distinguish tuples in an equivalent set by issuing queries and inserting tuples. We observe that an equivalent set does not have to be formed by true tuples if the adversary has no prior knowledge of the authenticity of the tuples. Consider the following situation. Given a database D, a tuple v ($v \in D$) and a ranking function $s(t|q)$, imagine we construct a virtual tuple t_1 such that t_1 is same with v over all public attributes and different from v over all private attributes. And when generating the rank scores of v and t_1 given an arbitrary query q, we use $s'(v|q)$ and $s'(t_1|q)$ ($s'(v|q) = s'(t_1|q) = \frac{s(v|q)+s(t_1|q)}{2}$) instead of $s(v|q)$ and $s(t_1|q)$, respectively. Though t_1 is a virtual tuple that does not exist in D, v and t_1 still form an equivalent set $\{v, t_1\}$ since $s'(v|q) = s'(t_1|q)$ for any queries. As a result, $|C_j^B| = 2$ and $P_{v[B_j]} = 50\%$. For a domain-ignorant adversary, we can achieve the privacy guarantee with virtual tuples.

An intuitive algorithm to generate an equivalent set of size $e+1$ for tuple v is to generate e virtual tuples $t_1, \ldots, t_e$ such that $t_i[B_j] \neq v[B_j]$, for all $i \in \{1 \ldots e\}$ and $j \in \{1 \ldots m'\}$. Specifically, let the initial $E_v = \{v\}$. Then we can generate a virtual tuple t_1 by assigning each $t_1[A_i]$ with $v[A_i]$ and each $t_1[B_j]$ with a value randomly picked from $V_j^B \setminus \{v[B_j]\}$ for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, m'\}$. In order to achieve the privacy guarantee $P_{v[B_j]} \geq \epsilon$, we can further generate more virtual tuples $t_2, \ldots, t_e$ to enlarge each C_j^B until $1 - \frac{1}{|C_j^B|} \geq \epsilon$ for $j \in \{1, \ldots, m'\}$. We name $\min\{|C_j^B|\}, \forall j \in \{1, \ldots, m'\}$ as the *cardinality* of E_v.

Privacy guarantee: For database D where every $v \in D$ is included by one and only one equivalent set whose *cardinality* is at least l, a privacy level of $\epsilon = 1 - \frac{1}{l}$ is achieved. Let us consider an arbitrary $v \in D$. Since v is included by an equivalent set of at least l cardinality, there are at least $l-1$ other tuples in E_v that have different values in B_j compared with v. For a domain-ignorant adversary, it is impossible to distinguish v with the $l-1$ tuples in E_v. Therefore, we have $P_{v[B_j]} \geq 1 - \frac{1}{l}$ for $\forall v \in D$ and $\forall j \in \{1, \ldots, m'\}$. According to (2), a privacy level of $1 - \frac{1}{l}$ can be achieved.

Utility optimization

In this section we consider the utility optimization of ESVT. In (4) we defined a general metric of utility loss based on the sum of rank differences of all tuples for all possible queries. In order to practically measure the utility loss, we introduce *query workload*, which is a set of queries, to our framework. Consider that we are given a query workload W and are required to optimize the utility loss on W. Equation (4) can be rewritten as

$$U_W = \sum_{q \in W} \sum_{v \in D} |\text{Rank}(v|q) - \text{Rank}'(v|q)|, \tag{6}$$

where W is the query workload.

Without loss of generality, we consider constructing equivalent sets of cardinality 2, i.e., to generate a virtual tuple t_1 for each $v \in D$. In order to achieve privacy, in a E_v of cardinality 2, $v[B_j] \neq t_1[B_j]$ for $\forall j \in \{1, \ldots, m'\}$. Now the problem is, given a query workload W, how can we find the assignments of virtual tuples' private attributes that minimize U_W. We prove that this problem is NP-Complete (see Appendix 4).

Heuristic algorithm

Since 2-Equivalent Problem is proved to be NP-Complete, we develop a heuristic algorithm which approximates the criteria of U_W that can be solved in polynomial time. To minimize utility loss U_W defined by (6), a heuristic approach is proposed to minimize the difference between $s(v|q)$ and $s(t_1|q)$ for all $v \in D$ and $q \in Q$. This new measure of utility, denoted as U_S, is an approximation of U_W. We define U_S as

$$U_S = \sum_{q \in Q} \sum_{v \in D} |s(v|q) - s'(v|q)|, \tag{7}$$

where $s(v|q)$, $s'(v|q)$ refer to the scores of tuple v given query q before and after the modification, respectively, i.e., $s'(v|q) = \frac{s(v|q)+s(t_1|q)}{2}$.

We start with Algorithm ESVT constructor which aims at constructing equivalent sets of size 2 for each $v \in D$. Then we show that constructing equivalent sets can be achieved by constructing virtual tuples successively. Finally we analyze the cost of the algorithm.

```
Algorithm 1: ESVT Constructor
  Input: $k$, $D$, $m$, $m'$, $W$
  Output: $ESet$
  $ESet = \emptyset$;
  for $v$ in $D$ do
      $t_1 = (v[A_1], \ldots, v[A_m], 0, \ldots, 0)$;
      for $l = 1$ to $m'$ do
          for $i = 1$ to $|W|$ do
              for $j = 1$ to $|W|$ do
                  if $q_i[B_l] == q_j[B_l]$ then
                      $p_i^{B_l}[j] = 1$;
                  else
                      $p_i^{B_l}[j] = 0$;
                  end
              end
          end
          $p_{|W|+1}^{B_l} = (0, \ldots, 0)$;
      end
      $H_{m'*(|W|+1),|W|} = \begin{bmatrix} p_1^{B_1}, \cdots, p_{|W|+1}^{B_1}, p_1^{B_2}, \cdots, \\ p_{|W|+1}^{B_2}, \cdots, p_1^{B_{m'}}, \cdots, p_{|W|+1}^{B_{m'}} \end{bmatrix}^T$;
      $G = (s(v|q_1), \ldots, s(v|q_{|W|}))$;
      solve MIQP problem with independent variable $\hat{x}$:
      $min(F(x)) = \hat{x}^T H_2 \hat{x} - f\hat{x} + G^2$ where $H_2 = HH^T$ and $f = 2 * GH^T$ subject to: $\hat{x}[i] = 0$
        or 1 for all $i \in 1, \ldots, m' * (|W| + 1)$ and
        $x[i * (|W| + 1) + 1] + x[i * (|W| + 1) + 2] + \cdots + [i * (|W| + 1) + |W| + 1] = 1$ for all
        $i \in \{1, \ldots, m'\}$;
      for $i$ = 1 to $m'$ do
          for $j$ = 1 to $|W| + 1$ do
              if $\hat{x}[(i-1) * (|W| + 1) + j] == 1$ then
                  if $j == |W| + 1$ then
                      $t'[i] = qb, qb \in DS_{B_i}$;
                  else
                      $t'[i] = q_j[B_i]$;
                  end
              end
          end
      end
      $E_v = \{v, t_1\}$;
      $ESet = ESet \cup E_v$;
  end
```

The pseudo code of this algorithm is shown in Algorithm 1. Given inputs tuple v(private attributes $B_1, \ldots, B_{m'}$) and query workload W, ESVT constructor constructs a virtual tuple t_1 for v and minimize $\sum_{q \in W} |s(v|q) - s(t_1|q)|$. We achieve this goal by creating a tuple t_1 which shares the same values with v on all public attributes and then assigning $t_1[B_1], \ldots, t_1[B_{m'}]$ with proper values. Without loss of generality, we assume that the value of all public attributes of v is *null*. Thus the initial $s(v|q)$ is 0. Our algorithm is expected to decrease the value of $S_{\text{remain}} = \sum_{q \in W}(s(v|q) - s(t_1|q))^2$ to minimize the difference between $s(v|q)$ and $s(t_1|q)$ for every $q \in W$.

We start with the first private attribute B_1. We define QVS_{B_1} as $\cup_{q \in W} q[B_1]$, i.e., the domain of B_1 in W. For B_1, there are $|V_1^B|$ candidate values for us to choose from. However, values in $V_1^B - QVS_{B_1}$ are equivalent and interchangeable: let set $DS_{B_1} = V_1^B - QVS_{B_1}$. For an arbitrary value $d_1 \in DS_{B_1}$, assignment $t_1[B_1] = d_1$ will not change the value of $s(t_1|q)$, $q \in W$ because $d_1 \neq q[B_1], q \in W$. Thus all the values in DS_{B_1} have the same effect on B_1 and we can use a single value d_{B_1} to represent them. Now we have $|W| + 1$ candidate values for attribute B_1: $q_1[B_1], \ldots, q_{|W|}[B_1], d_{B_1}$. For each candidate value qb, we use a 0–1 vector of length $|Q|$ to describe the effect of assignment $t_1[B_1] = qb$ on $s(t_1|q_1), \ldots, s(t_1|q_{|W|})$ and denote it as $\hat{p}_i^{B_1}$. The value of $\hat{p}_i^{B_1}[j]$ refers to the effect of assignment $t_1[B_1] = q_i[B_1]$ on $s(t_1|q_j)$. For instance, if $q_i[B_1] = q_j[B_1]$, then the value of $s(t_1|q_j)$ will increase by 1 and thus we let $\hat{p}_i^{B_1}[j] = 1$. If $q_i[B_1] \neq q_j[B_1]$, then the value of $s(t_1|q_j)$ remains the same and we should let $\hat{p}_i^{B_1}[j] = 0$. Note that $\hat{p}_{|W|+1}^{B_1}$, the vector describing the effect of $t_1[B_1] = d_{B_1}$, is a zero vector. Also note that if $q_i[B_1] = v[B_1]$ then we let $\hat{p}_i^{B_1} = \{\infty, \ldots, \infty\}$, as we do not want to assign $t_1[B_1]$ the same value of $v[B_1]$.

Now we have constructed $|W| + 1$ vectors $(\hat{p}_1^{B_1}, \ldots, \hat{p}_{|W|+1}^{B_1})$ for B_1. In the same way we can also construct $|W| + 1$ vectors for each attributes $B_2, \ldots, B_{m'}j$. Note that the assignment of B_1 is not independent from the assignment of B_2 and all the other private attributes. Therefore we set up a goal vector $G = \{s(v|q_1), \ldots, s(v|q_{|W|})\}$. Now the problem of minimizing $\sum_{q\in W} |s(v|q) - s(t_1|q)|^2$ is transformed into problem:

$$\min\left|\sum_{i=1}^{m} p^{B_i} - G\right|^2, \quad \text{where each } p^{B_i} \text{ is chosen from } \{p_1^{B_i}, \ldots, p_{|W|+1}^{B_i}\}. \tag{8}$$

The solution of (8) can be described by a column vector $\hat{x}$ of length $m * (|W| + 1)$ such that the $i * (|W| + 1) + j$th element of $\hat{x}$ is equal to one if and only if $p^{B_i} = p_j^{B_i}$ and is equal to zero in other conditions. To solve problem (8), we construct matrix H as

$$H_{m'*(|W|+1),|W|} = \left[p_1^{B_1}, \ldots, p_{|W|+1}^{B_1}, p_1^{B_2}, \ldots, p_{|W|+1}^{B_2}, \ldots, p_1^{B_{m'}}, \ldots, p_{|W|+1}^{B_{m'}}\right]^T.$$

Note that $\sum_{i=1}^{m} p^{B_i}$ in (8) is equal to $\hat{x}^T * H$. Thus (8) can be further formalized to problem:

$$\min F(x) = \hat{x}^T H_2 \hat{x} - f\hat{x} + G^2 \quad \text{where } H_2 = HH^T \text{and } f = 2 * GH^T \tag{9}$$

subject to $\hat{x}[i] = 0$ *or* $1$for all $i \in 1, \ldots, m' * (|W| + 1)$ *and* $x[i*(|W|+1)+1]+x[i*(|W|+1)+2] + \cdots + [i * (|W| + 1) + |W| + 1] = 1$ *for all* $i \in 1, \ldots, m'$.

Problem (9) is a mixed integer quadratic programming problem (MIQP). The parameter H_2 in (9) is a positive definite matrix. Therefore, we can solve the MIQP problem in polynomial time [19] and use $\hat{x}$ to construct t_1.

The above algorithm can be further extended to the case when the cardinality of equivalent sets is larger than 2 by simply removing $t_1[B_1], \ldots, t_1[B_m]$ from $V_1^B, \ldots, V_{m'}^B$, respectively, after constructing t_1. Then we can generate another virtual tuple t_2 for v by repeating the process that generates t_1. For a dataset of n tuples, the computational complexity of Algorithm 1 is $O(n \cdot m'^3)$.

Framework with true tuples

The feasibility of privacy-preserving framework is established in "Framework with virtual tuples" section. Recall that the domain-expert adversaries hold the prior knowledge of non-trivial attributes. In "Domain-expert adversaries" section, we firstly study the ability of domain-expert adversaries and how this ability can break the privacy guarantee of the privacy-preserving framework proposed in "Framework with virtual tuples" section. In "Design" section, we introduce a new implementation of privacy-preserving framework and prove its privacy guarantee. In "Utility optimization" section, we analyze its utility loss and propose an optimal solution to minimize the utility loss. We give a practical heuristic algorithm of constructing the equivalent sets for this privacy-preserving framework in "Heuristic algorithm" section.

Domain-expert adversaries

As we mentioned in "Introduction" section, domain-expert adversaries could view public attributes, as well as they are able to validate the authenticity of private attribute values. Here, we start by showing that if adversaries hold the prior knowledge of the non-trivial attribute correlation, the privacy guarantee $P_{v[B_j]}$ for the equivalent set with virtual tuples cannot be achieved.

We consider a simple case that domain-expert adversaries can validate a private attribute B_1 by knowing a set of public attributes S_A. We denote the prior knowledge of the attribute correlation as $S_A \Rightarrow B_1$, where S_A is a determinant *set* of public attributes, and B_1 is a *dependent* private attribute. Note that this attribute correlation exists when there are functional dependency and/or inevitable dependency between attributes in the dataset. In reality, the adversary can acquire such prior knowledge of non-trivial attribute correlation by being/consulting a domain expert or using data mining techniques [20] to explore correlations between attributes. For example, based on the personal information (e.g., gender, ethnicity, age, blood type) which stored as public attributes and published in public medical data repositories, genetic epidemiologists can generally conclude a dominant gene is associated with a particular disease by the candidate-gene approach. Therefore, domain-expert adversaries could know that some candidate values violate the correlation of $S_A \Rightarrow B_1$.

As stated in "Framework with virtual tuples" section, constructing E_v by virtual tuples for victim tuple v, we can construct an equivalent set $E_v = \{v, t_1, \dots, t_k\}$, where $v \in D$, $t_1, \dots, t_k \notin D$ and $v[B_j], t_1[B_j], \dots, t_k[B_j] \in V_1^B$. We assume that domain-expert adversaries can retrieve $C_j^B = \{v[B_j], t_1[B_j], \dots, t_k[B_j]\}$ by arbitrary attacking methods. From discussed above, according to $S_A \Rightarrow B_j$, domain-expert adversaries could know that

$$\exists v[B_1] = C_1^B \setminus t_e[B_1], \quad e \in \{1, \dots, k\}. \tag{10}$$

We now can prove that the ESVT loses its privacy guarantee $P_{v[B_1]}$ when facing the attack from domain-expert adversaries:

$$P'_{v[B_1]} = \left(1 - \frac{1}{|C_1^B| - k}\right) * 100 < P_{v[B_1]}, \quad \text{where } 1 \le k < |C_1^B|. \tag{11}$$

Remember that when constructing ESVT, we generate equivalent tuples for each v merely based on query workload and the assumption of domain-ignorant adversaries. As shown above, once $t_e[B_1]$ can be excluded from C_1^B when known $S_A \Rightarrow B_1$, the privacy guarantee loses.

Especially, it is a considerable detrimental threat to the ESVT of cardinality 2. Under this circumstances, suppose domain-expert adversaries can retrieve $C_j^B = \{v[B_j], t[B_j]\}$, one can notice that there are only two candidate values in C_j^B: $v[B_j]$ are targets, and $t[B_j]$ are virtual values. Assume that domain-expert adversaries hold the prior knowledge of correlation $S_A \Rightarrow B_j$, in the worst-case scenario, domain-expert adversaries know that every $t[B_j]$ is invalid. Therefore $\forall v[B_j] = C_j^B \setminus t[B_j]$; domain-expert adversaries then can conclude that $C_j^B = \{v[B_j]\}$. Recall that the adversary's goal is to maximize Function (5). In this case, adversaries would have $g_A = 100\%$ for every $v[B_j]$.

So far, we have seen how domain-expert adversaries could break the privacy guarantee mentioned in (3), and thereby cause the detrimental consequence of privacy disclosure. In reality, there are a number of prior knowledge that potentially can be used to validate the private attribute. In following sections, to protect privacy under attacks from domain-expert adversaries, we propose and give the design of constructing equivalent set with true tuples. Then we prove that the privacy guarantee is achieved no matter what prior knowledge domain-expert adversaries hold. We also give an optimization algorithm and investigate the cost.

Design

We have shown above that domain-expert adversaries can break the privacy guarantee of ESVT in "Framework with virtual tuples" section, so that the necessity of designing a robust privacy-preserving framework is unquestionable. In this subsection, we propose a framework that constructs equivalent sets with true tuples, which we will refer to as *Equivalent Set with True Tuples* (ESTT). This framework offers the same degree of privacy guarantee for private attributes as mentioned in (3), even under the attack from domain-expert adversaries. We must point out that the new design is different with the design of ESVT in "Framework with virtual tuples" section. Intuitively, each equivalent set consists of true tuples only in this new framework. In other words, here we avoid generating tuples that violate the prior knowledge of domain-expert adversaries. Instead, we use existing tuples in the dataset because they comply with potential prior knowledge of domain-expert adversaries (e.g., the attribute correlation as we discussed in "Domain-expert adversaries" section). Note that in this paper, we only consider that users are willing to input true information which complies with prior knowledge.

Definition 3 If there are at least l "valid" candidate values for every private attribute in *equivalent sets*, we call it *l-candidate equivalent set*, where $l \leq \min(|V_j^B|)$, $j \in \{1, \ldots, m'\}$.

As we mentioned in "Ranked retrieval model" section, in the ranking model, for each tuple t, $t[B_1], \ldots, t[B_j] \in V_j^B$. And in ESTT, $t \in D$. Therefore, one can find out that the maximal $|C_j^B|$ is the minimum $|V_j^B|$. Therefore, $l \leq \min(|V_j^B|)$ in the l-candidate equivalent set.

Privacy guarantee: The l-candidate equivalent set can achieve the same privacy guarantee as the framework with virtual tuples even if adversaries are domain-expert adversaries. We denote the equivalent set as $E_v = \{v_1, \ldots, v_k\}$, where $v_1, \ldots, v_k \in D$. Note that, the same as constructing E_v in "Framework with virtual tuples" section, in each E_v we make $s(v_1|q) = s(v_2|q) = \cdots = s(v_k|q)$. It is exceedingly important that $v_1, \ldots, v_k$ share the same public attributes. One can find out that all values in C_j^B satisfy $S_A \Rightarrow B_j$ because $v_1, \ldots, v_k$ are true tuples. Adversaries have to conclude that $v_1[B_j], \ldots, v_k[B_j]$ are possible true values for v_1. Therefore, $S_A \Rightarrow B_j$ cannot be used to exclude any candidate values from C_j^B. Here, we give our privacy guarantee. Assume that the target of adversaries is B_j, and adversaries are able to retrieve C_j^B by arbitrary attacking methods. In this case, since there are at least l "valid" candidate values for private attribute B_j, which means $|C_j^B| \geq l$, according to (3):

$$P'_{v_1[B_j]} = \left(1 - \frac{1}{|C_j^B|}\right) * 100\% \geq \left(1 - \frac{1}{l}\right) * 100\%. \tag{12}$$

The privacy guarantee is achieved by ESTT. As we mentioned above, the framework can defend attacks from domain-expert adversaries who hold the prior knowledge of attribute correlation. We now extend the general assumption that domain-expert adversaries can hold any non-trivial prior knowledge. Consider the simplest ESVT, which is *2-candidate equivalent set*. It guarantees at least 2 valid candidate values for each B_j. Thus, according to (12), $P'_{v_1[B_j]} \geq (1 - \frac{1}{2}) * 100\%$. The privacy guarantee is achieved. Furthermore, for adversaries, the goal is to maximize Function (5). In this case, adversaries would have $g_A = 50\%$ for every $v_1[B_j]$. Obviously, we limit the probability of adversaries to get the correct private attribute value of the victim tuple to 50%.

Secondly, it is important to set proper size for l-candidate equivalent set when considering the practicability in real web databases. Admittedly, if we put n' tuples that share the same public attributes into a single equivalent set E_v, it could satisfy the *l-candidate* requirement. However, it is impractical because every tuple in E_v will return the same score since $s(v_1|q) = s(v_2|q) = \cdots = s(v_{n'}|q)$. Thereby the ranking function loses its functionality. Here we introduce the set size k, which is the number of tuples in a l-candidate equivalent set.

Definition 4 When a l-candidate equivalent set contains k different true tuples, respectively, we call it *l-candidate equivalent set with k-tuples*, where $k \geq l$, and $l \leq \min(|V_j^B|)$, $j \in \{1, \ldots, m'\}$.

Note that each E_v must have at least l "valid" candidate values for every private attribute, and there must be enough tuples in each l-candidate equivalent set so that $k \geq l$.

Thirdly, we introduce a key technique that is adopted in the process of constructing ESTT—*Data Obfuscation*, which transforms the original data to random data [21]. It is widely used in protecting data privacy; for example, in [22], they added a random noise to the victim tuple so that the true value is disguised. Ideally, under the limitation of set size k and requirement of l-candidate values, if we can find enough tuples that their private attributes value are mutually different, or at least $|C_j^B| \geq l, \forall j \in \{1, \ldots, m'\}$, we put them together to construct a E_v. However, we cannot avoid the circumstances that the $\exists |C_j^B| < l$ probably because V_j^B lacks diversity or in the process of constructing last few E_v. Under these circumstances, in order to maintain $|C_j^B| \geq l, \forall j \in \{1, \ldots, m'\}$ for every E_v, otherwise privacy guarantee cannot be achieved, we use the *Data Obfuscation* to conceal original private attribute values by assigning random values. Specifically, regarding a E_v, assign l different values to $v_1[B_j], \ldots, v_k[B_j]$, respectively, so that eventually $|C_j^B| \geq l$ in this E_v. One can know that, for any C_j^B, it is better to keep every true value appearing in $v_1[B_j], \ldots, v_k[B_j]$, and then try to randomly pick other non-repeated value(s) that satisfy the constrain of $S_A \Rightarrow B_j$. In the end, every private attribute looks real for the adversaries and also satisfies the requirement of l-candidate.

Implementation We now provide the implementation of the algorithm for constructing privacy-preserving equivalent sets with true tuples for a given database D and two input variables l and k. The algorithm is straightforward—we start with partitioning D into small groups based on public attributes. Then constructing l-candidate equivalent set with k-tuples recursively among each partition by adding unprotected tuples until the size of set reaches k. Specifically, for a given E_v, where $E_v = \{v_1, \ldots, v_k\}$, if there are at least l "valid" values existing in every private attribute, no more action is needed. However, if the number of "valid" values for any private attribute is smaller than l, the Data Obfuscation on corresponding private attributes is enforced. Thirdly, we mark all tuples in E_v as *protected*. Continue constructing the l-candidate equivalent set with k-tuples until no more E_v with the size of k can be constructed. Gather the remaining unprotected tuples into an equivalent set and enforce Data Obfuscation for any private attribute which cannot hold the privacy guarantee. Repeat the process of constructing E_v in each partition. In the end, each E_v achieves l-candidate condition with k-tuples. We give detail in Algorithm 2.

Algorithm 2: *l-candidate equivalent set with k-tuples*

```
Input: $l$, $k$, $D$, $m$, $m'$
Output: $E_v$
if $l > \min(V_i^B)$ or $l > k$ then
    return FALSE;
else
    Partition($D$, $m$);
    for $i = 0; i < (Count_{Partition(D,m)})$ do
        Pick($v_1, \ldots, v_k \in Partition_i(D, m)$);
        for $i = 0; i < m'$ do
            if $Lcandidate\ (v_1[B_i], \ldots, v_k[B_i]) \geq l$ then
                return $(v_1[B_i], \ldots, v_k[B_i])$;
            else
                Obfuscation $(v_1[B_i], \ldots, v_k[B_i])$;
                if $Lcandidate\ (v_1[B_i], \ldots, v_k[B_i]) \geq l$ then
                    return $(v_1[B_i], \ldots, v_k[B_i])$;
                end
            end
        end
        return $E_v$;
    end
end
```

For example, the construction of *2-candidate equivalent set with 2-tuples*, that $E_v = \{v_1, v_2\}$, where $v_1[B_j] \neq v_2[B_j]$, $j \in \{1, 2, \ldots, m'\}$. We start with partitioning data into different partitions based on public attributes. Then construct equivalent set by selecting two true tuples among the partition. If necessary, enforce *Data Obfuscation* on private attributes where both tuples share the same attribute value. Then mark these two tuples as *protected*. Continue constructing equivalent sets until no more equivalent set can be constructed. In the end, for each partition group, either no more tuple need to be protected so that we can jump to next partition, or one tuple v is still *unprotected* then we need to enforce the *Data Obfuscation* on v. Recursively execute the constructing process until every tuple in D is marked as *protected*. Next, we analyze its utility loss and provide an optimal solution.

Utility optimization

In "Utility loss measurement" section we give the measurement of $\mathbf{U}$. In "Utility optimization" section, an optimal algorithm is given with minimum $\mathbf{U}$, where the minimum $\mathbf{U}$ can always be found since the algorithm keeps constructing and screening virtual tuples based on query workload W. It uses the following Eq. (13), that given input W, minimize $\sum_{i=1}^{|W|} |\text{Rank}(v|q_i) - \text{Rank}'(v|q_i)|$.

$$\mathbf{U} = \sum_{j=1}^{n} \left(\sum_{i=1}^{|W|} |\text{Rank}(v_j|q_i) - \text{Rank}'(v_j|q_i)| + \sum_{i=|W|}^{n} |\text{Rank}(v_j|q_i) - \text{Rank}'(v_j|q_i)| \right). \tag{13}$$

However, minimizing U is not a practical optimal algorithm here. Firstly, one can observe that optimizing $\sum_{i=1}^{|W|} |\text{Rank}(v_j|q_i) - \text{Rank}'(v_j|q_i)|$ in Eq. (13) is difficult. Because we enforce the data obfuscation in constructing the equivalent sets, the $\text{Rank}'(v_j|q_i)$ is dynamically changing when different values are assigned to $v[B_j]$. Secondly, one can observe that fundamentally, Algorithm 2 is looking for equivalent tuples that have as many of the same private attributes as possible regarding query workload W, and put them into the same equivalent set. It indeed reduce the utility loss for given W; however, for the rest of $n - |W|$ possible queries, the utility loss actually increases.

One can know that the expected global utility loss can be represented as Eq. (14). For all possible n queries, the $\mathbf{Exp}(U)$ is the same. However, data obfuscation brings information loss which leads to decrease in the functionality of score function $s(t|q)$. Therefore, without considering query workload and maximizing the functionality of score function $s(t|q)$, in this subsection, we introduce a better measurement for the utility loss of l-candidate equivalent set with k-tuples, which is the number of data obfuscation.

$$\mathbf{Exp}(U) = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{n} |\text{Rank}(v_j|q_i) - \text{Rank}'(v_j|q_i)|. \tag{14}$$

In real web databases, minimizing the total number of data obfuscation is an optimization because it reduces information loss while protecting privacy. Here we define the utility optimization problem of l-candidate equivalent set with k-tuples as follows:

Definition 5 The optimization problem of l-candidate equivalent set with k-tuples is to find the solution that obfuscates the fewest number of private attribute values.

For l-candidate equivalent set with k-tuples E_v, $k \geq l$, and $l \leq \min(V_j^B)$ for any $j \in \{1, \ldots, m'\}$, we construct E_v which satisfy the fewest number of data obfuscation on private attribute values. We prove that the optimization problem of 2-candidate equivalent set with 2-tuples is a NP-complete problem.

Lemma 1 *Minimum length Hamiltonian circuit problem is NP-complete.*

Table 1 Vertex matrix

	v_1	v_2	v_3	v_4	$\dots$	v_e
v_1	0	Same(v_1, v_2)	Same(v_1, v_3)	Same(v_1, v_4)	$\dots$	Same(v_1, v_e)
v_2	Same(v_1, v_2)	0	Same(v_2, v_3)	Same(v_2, v_4)	$\dots$	Same(v_2, v_e)
v_3	Same(v_1, v_3)	Same(v_2, v_3)	0	Same(v_3, v_4)	$\dots$	Same(v_3, v_e)
v_4	Same(v_1, v_4)	Same(v_2, v_4)	Same(v_3, v_4)	0	$\dots$	Same(v_4, v_e)
$\dots$	Same$(\dots)$	Same$(\dots)$	Same$(\dots)$	Same$(\dots)$	$\dots$	$\dots$
v_e	Same(v_1, v_e)	Same(v_2, v_e)	Same(v_3, v_e)	Same(v_4, v_e)	$\dots$	0

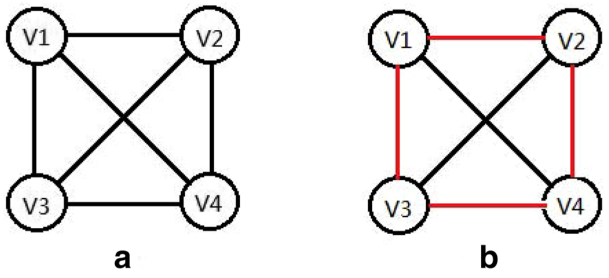

Fig. 1 **a** Hamiltonian circuit in **b**

Proof Let Same(v_α, v_β) be the number of same attribute values between v_α and v_β. Suppose we have n tuples in D, and choose a partition which contains e tuples that have the same public attributes. We can get the following matrix table:

Based on Table 1, we can construct an undirected graph G, where the vertex is the tuple v_i where $i \in \{1, 2, \dots, e\}$ and the edge weights are Same(v_α, v_β) from the matrix. As we can know, G is a complete graph that every pair of distinct vertices is connected by a unique edge. Figure 1a is an example based on 4 tuples in a P to construct a complete graph. Therefore, G exists *Hamiltonian Circuit*. However, according to Lemma 1, finding the Minimum Length Hamiltonian Path is NP-complete. In Fig. 1b, the red line path shows a Hamiltonian Circuit.

Lemma 2 *Minimum Length Hamiltonian Circuit* $\leq_P$ *Optimization problem of 2-candidate equivalent set with 2-tuples*

We now proof optimization problem of 2-candidate equivalent set with 2-tuples is NP-complete. Without loss of generality, we select a partition P where $\{v_1, v_2, ..., v_e\} \in P$ as well as $v_1, v_2, ..., v_e$ share the same public attributes. Next, we reduced the optimization problem of 2-candidate equivalent set with 2-tuples to Minimum Length Hamiltonian Path. In the optimization problem of 2-candidate equivalent set with 2-tuples, intuitively, we pair every two tuples and the goal is to minimize the total weights. Since we have the Minimum Length Hamiltonian Path, which visits each vertex exactly once, we can find a polynomial time function, f, that constructs the optimization solution of 2-candidate equivalent set with 2-tuples by removing edges from the Minimum Length

Hamiltonian Path. Therefore, Minimum Length Hamiltonian Path $\leq_P$ Optimization problem of 2-candidate equivalent set with 2-tuples. □

Heuristic algorithm

We have shown that the optimization problem of *2-candidate equivalent set with 2-tuples* is NP-complete. We thereby propose a heuristic algorithm to construct *2-candidate equivalent set with 2-tuples*. The heuristic algorithm is a slight modification of Algorithm 2—we make a slight modification on the function of *Pick* by applying the greedy algorithm. We call it *Sorted Weight Algorithm*.

We give the detail of *Sorted Weight Algorithm* as follows: we start with partitioning D into small groups based on public attributes. Without loss of generality, we assume that each partition has n' tuples. Among each partition, compute $\mathrm{Same}(t_\alpha, t_\beta)$, where $\alpha \neq \beta$. Store $\mathrm{Same}(t_\alpha, t_\beta)$ to the corresponding cell in the matrix as shown in Table 1. Sort the values in matrix by increasing weight. Then start to construct *2-candidate equivalent set with 2-tuples* by recursively picking tuples that have the smallest weight edges. If at least l "valid" values exist in every private attribute regarding to the two picked tuples, no more action is needed. However, if the number of "valid" values for any private attribute is smaller than l, the *Data Obfuscation* on corresponding private attributes is enforced. Then we mark these two tuples as "protected" and decrease their weights regarding other tuples to $+\infty$. Continue constructing the *l-candidate* equivalent set with k-tuples until no more E_v with the size of k can be constructed. Gather the remaining *unprotected* tuples into an equivalent set and enforce *Data Obfuscation* for any private attribute which cannot hold the privacy guarantee. Repeat the process of constructing E_v in each partition. In the end, each E_v achieves *2-candidate* condition with *2-tuples*. The time complexity of *Sorted Weight Algorithm* is $O(n)$.

Experimental results

Experimental setup

Hardware and platform

All our experiments were performed on a Mac machine running Mac OS with 8 GB of RAM. The algorithms were implemented in Python and Matlab. We implement the ESTT as 2-diversity equivalent set with 2-tuples.

Dataset

We used a real-world dataset from eHarmony to verify the utility and efficiency of our privacy-preserving framework. eHarmony dataset contains 486,464 tuples and 53 attributes, of which more than 30 are boolean [23]. In the experiment of computing attack success guess rates, we picked 5 attributes as users' public attributes as well as picked other 5 attributes as users' private attributes. The respective domain size of public attributes are 5, 2, 2, 5, 6, and the respective domain size of private attributes are 5, 2, 2, 5, 6. In the experiment of measuring performance, among all available attributes, we randomly picked 10 attributes as public attributes, and varied the number of private attributes as 5, 10, and 15, respectively. We also picked 10 attributes as private attributes, and varied the number of public attributes as 5, 10, and 15, respectively. After removal of duplicate tuples, we sampled $\mathrm{n} = 300{,}000$ tuples without replacement as our testing bed.

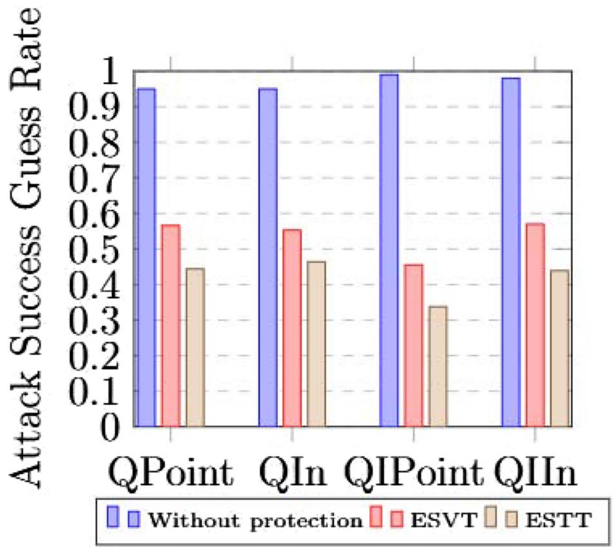

Fig. 2 Attack guess success rate

By default, we constructed a query workload of size 10 by randomly picking 10 tuples from the sample. And we used the ranking function from "Framework" section with all weights set to 1.

Performance measures

As explained in 2, our framework provides a certain degree of guarantee to privacy while minimize the total utility loss. In this section, we measure privacy by the probability of success guess in rank inference attack [1] and utility loss by one criterion: average top-k rank difference.

Experiments over real-world dataset

In "Utility loss measurement" section, the utility of the database under our framework is defined by (4). In this section, we introduce average top-k utility loss as a practical metric to describe utility loss in real-world databases. The average top-k utility loss of a given database under privacy-preserving framework is defined as

$$U_{\text{avg}} = \frac{1}{|W| * k * |D'|} \sum_{i=1}^{|W|} \sum_{t \in D'} |\min\{R_t, k+1\} - \min\{R'_t, k+1\}|, \tag{15}$$

where $D' = \{t \in D | R_t < k \text{ or } R'_t < k\}$ and R_t/R'_t denote the rank of tuple t before/after defense mechanism. Intuitively, U_{avg} is a measure of average change in rank within tuples that we are interested, i.e., tuples in top-k.

Evaluation of attack success guess rate

Figure 2 shows the probability of success guess over the dataset without protection, with ESVT and with ESTT. Theoretically, the adversary can achieve 100% success guess rate by using inference attack. In the case of attacking on the dataset with ESVT

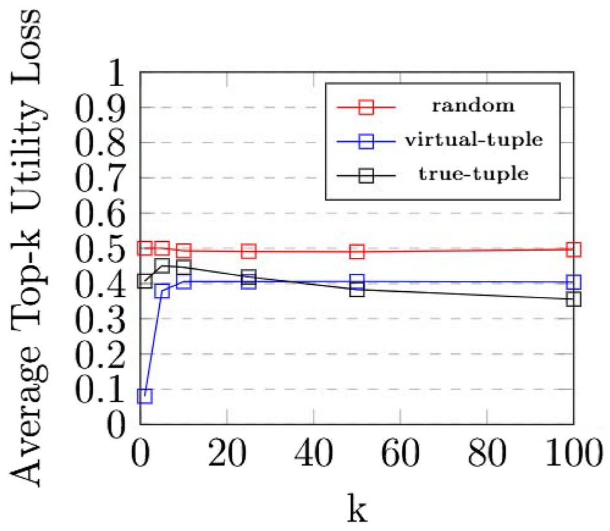

Fig. 3 EVSV and EVST vs. random

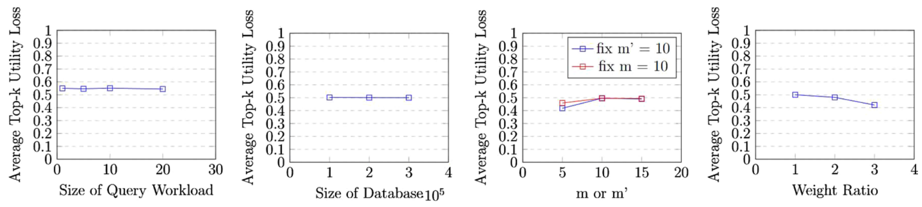

Fig. 4 Utility loss of EVSV

and ESTT, the success guess rates are around 50%. It is because the adversary can retrieve two possible values for a private attribute from a equivalent set and cannot identify the true value from them. We can observe that with ESTT, the success guess rate is lower than 50%. It is because the victim may be applied suppression when constructing ESTT.

Evaluation of utility versus k We first investigated the performance of our algorithms for different values of *k*. Figure 3 shows the average top-*k* utility loss of our algorithms with true tuples(true-tuple), virtual tuples(virtual-tuple), and a baseline algorithm(random), which constructs equivalent sets with randomly picked tuples. As expected, the average top-*k* utility loss of the baseline method is around 0.5 given diverse *k* values. The average utility loss of both virtual-tuple and true-tuple is lower than that of the baseline method, which indicates that our heuristic algorithms can reduce the utility loss while preserving privacy. We also observe that when the value of *k* increases, the utility loss will show a trend of first increase and then decrease. The reason is that as the number of tuples increases in top-*k* results, it is likely that tuples that are originally in the top-*k* results would have better chance to stay in the new top-*k* results after applying algorithms.

Utility loss versus m and m' The first chart in Figs. 4 and 5 demonstrates the impact of query workload on average top-*k* utility loss of our algorithms with virtual tuples

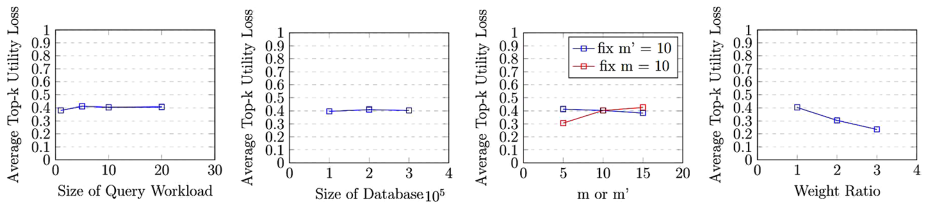

Fig. 5 Utility loss of EVST

and true tuples, respectively. As expected, the size of query workload does not have any significant impact on the utility since the tuples in the query workload are randomly generated.

Utility loss versus database size The second chart in Figs. 4 and 5 shows the impact of query workload on average top-k utility loss of our algorithms with virtual tuples and true tuples, respectively. As expected, the size of database does not have any significant impact on the utility when k(10 by default) is much smaller than the size of database.

Utility loss versus domain size of m We then investigate the utility loss under different numbers of public attributes with a fixed number of private attributes and under different numbers of private attributes with a fixed number of public attributes. The third chart in Figs. 4 and 5 shows the result produced by our algorithms with virtual tuples and with true tuples, respectively. In the case of the framework with true tuples, the utility will decrease as the size of public attributes increases. When the size of public attributes increases, the number of tuples that have the same public attributes would decrease. Therefore, it is unlikely to decrease suppressions because of less choices under the circumstances of limited number of candidate true tuples. In the case of the framework with virtual tuples, the utility loss increases with more private attributes and decreases with more public attributes. It is due to the fact that with higher proportion of public attributes, tuples in the same equivalent set would have a higher proportion of common attribute values, which leads to less difference in score.

Utility loss versus weight ratio In this experiment, we fixed the weight of all private attributes to 1 and varied the weights of all public attributes from 1 to 3. The last chart in Figs. 4 and 5 shows that when weight ratio increases, the utility loss will decrease. The reason is that when the weight ratio increases, the suppressions and alternations applied to private attributes would have less influence to the score. Therefore, more tuples that are originally in the top-k results would have less rank differences after applying our algorithms.

Final remarks

In this paper, we addressed the issue related to privacy preserving in ranked retrieval model, which has been adopted widely to web databases. We proposed a privacy-preserving framework to protect private attributes privacy which can defend not only the rank-based inference attack but also arbitrary attacks. We introduced a classification of adversaries and their capability. For domain-ignorant adversaries, we designed ESVT and proved its privacy guarantee. For domain-expert adversaries, we designed ESTT and proved its privacy guarantee. For both ESVT and ESTT, we developed heuristic algorithm, respectively, for practical situations under the consideration of minimizing the utility loss.

We would like to remark that in this paper, we showed that simple and efficient solutions can be developed to deal with the privacy disclosure in ranked retrieval model with little utility loss.

Our works can be readily applied to existing web databases. We hope that these works initiate a new topic of privacy preserving in ranked retrieval model, and future research will discuss optimization of ESVT and ESTT.

Acknowledgements
We are thankful to the School of Engineering and Applied Science, George Washington University for the support that enabled the study to be carried out successfully. We are also thankful to Weimo Liu for research assistance.

Authors' contributions
TY, YG, and NZ developed the concept. TY and YG drafted the manuscript. All authors read and approved the final manuscript.

Funding
The authors were supported in part by the National Science Foundation under grants 0852674, 0915834, 1117297, and 1343976, and by the Army Research Office under grant W911NF-15-1-0020. Any opinions, findings, conclusions, and/or recommendations expressed in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsors listed above.

Availability of data and materials
The data that support the findings of this study are available from the corresponding author upon reasonable request.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] School of Engineering and Applied Science, George Washington University, 2121 I St NW, Washington, DC 20052, USA.
[2] Kogod School of Business, American University, 4400 Massachusetts Ave NW, Washington, DC 24105, USA.

Appendix

Definition 6 For a tuple v in D, we create v's equivalent set of cardinality 2, denoted by $E_v = \{v, t_1\}$. The score of v considering E_v is $s'(v|q) = \frac{s(v|q)+s(t_1|q)}{2}$. We say t_1 satisfies query q when for any tuple $t(t \notin E_v)$ in D: (1) If $s(v|q) < s(t|q)$, then $s'(v|q) < s'(t|q)$, and if $s(v|q) \geq s(t|q)$, then $s'(v|q) \geq s'(t|q)$.

The 2-Equivalent Set Problem is defined as: given a query workload Q and a database D, constructing $|D|$ equivalent sets of cardinality 2 which satisfy the most queries in Q.

Definition 7 Given a 3-CNF formula Φ and a number k, decide whether there exists an assignment satisfying at least k of the clauses. We call the above problem Max-3Sat Problem.

Theorem 3 *Max-3Sat* $\leq_P$ *2-Equivalent Set Problem.*

Proof Let the reduction function $f(\Phi) = (D, s, Q, T)$ take a *Max-3sat* problem as input and return a 2-Equivalent Set Problem. Without loss of generality, suppose Φ is a conjunction of l clauses and each clause is a disjunction of 3 variables from set $X = \{x_1, \ldots, x_n\}$. We define database D as follows: D has 0 public attribute and $n+2$ private attributes $B_1, \ldots, B_n, C_1, C_2$. Let V_i^B and V_j^C be the attribute domains for B_i and

C_j, respectively, $i \in \{1 \ldots n\}$ and $j \in \{1, 2\}$. Let $V_i^B = \{0, 1, 2, \ldots, r\}$ and $V_j^C = \{0, 1\}$. We define the score function $s(t|q)$ as follows:

$$s(t|q) = \sum_{i=1}^{n} D_b(t[B_i], q[B_i]) + \sum_{i=1}^{l} D_c(t[C_i], q[C_i]), \tag{16}$$

where

a. $D_b(t[B_i], q[B_i]) = 1$, iff $t[B_i] = q[B_i]$,
b. $D_b(t[B_i], q[B_i]) = 0$, if $t[B_i] \neq q[B_i]$ or $t[B_i]$ is null or $q[B_i]$ is null,
c. $D_c(t[C_i], q[C_i]) = 0$ iff $t[C_i] \neq q[C_i]$ and,
d. $D_c(t[C_i], q[C_i]) = 1$ iff $t[C_i] = q[C_i]$.

Now we want to construct a set of queries Q and a set of tuples T. First we construct a tuple v_1, assign $v_1[B_i]$ for $i \in \{1, \ldots, n\}$ with 0, and assign $v_1[C_i]$ for $i \in \{1, 2\}$ with 0. Then we construct a tuple v_2, assign $v_2[B_i]$ for $i \in \{1, \ldots, n\}$ with $i + 2$ and assign $v_2[C_1]$, $v_2[C_2]$ with 1, 0, respectively. Then for each clause in Φ we construct a query q such that $q[B_i] = i$ iff the corresponding variable x_i appears in the clause as x_i, and $q[B_i] = i + 1$ iff x_i appears in the clause as $\neg x_i$. We also set $q[C_1] = q[C_2] = 0$. For example, if we have a clause $C_1 = (x_1 \vee \neg x_2 \vee x_3)$, then we should construct a query q_1 such that $q_1[B_1] = 1, q_1[B_2] = 3, q_1[B_3] = 3, q_1[C_1] = 0$, and $q_1[C_2] = 0$.

After constructing a query for each of the l clauses, we have a set of queries $Q = \{q_1, \ldots, q_l\}$ and a set of tuples $T = \{v_1, v_2\}$. Assume that we have already constructed a perfect equivalent set for v_2 such that $s'(v_2|q) = s(v_2|q)\ \forall q \in Q$. Now we have a instance of Rank Inference Problem and we still need to construct the equivalent set for v_1. For an arbitrary query $q_i \in Q$, we have $s(v_1|q_i) = 1 + 1 = 2$ and $s(v_2|q_i) = 1$. Suppose that the virtual tuple of v_1, denoted as t_1, satisfies q_i. Because $v_1[C_1] = v_1[C_2] = 0$, we have to set $t_1[C_1] = t_1[C_2] = 1$ for all q_i. Note that $s(v_2|q_i) = 1 < s(v_1|q_i)$. Therefore, we have to ensure that $s'(v_1|q_i) > s'(v_2|q_i) = s(v_2|q_i)$. Thus we have

$$\begin{aligned} & s'(v_1|q_i) > 1 \\ \Leftrightarrow\ & s(t_1|q_i) > 1 \\ \Leftrightarrow\ & \text{there exists at least one attribute } B_z \text{ such that } t_1[B_z] = q_i[B_z]. \end{aligned} \tag{17}$$

Remember that we assign $q_i[B_z]$ with z or $z + 1$ based on the fact that x_z appears in clause i in the form of x_z or $\neg x_z$, respectively. Thus, $t_1[B_z] = q_i[B_z]$ infers that clause i is satisfiable.

As we proved above, the fact that t_1 satisfies q_i infers that the corresponding clause i is satisfiable. Thus suppose we have a solution t_1 which satisfies at least k queries in Q. Then the assignment S:

$$x_i = 1 \text{ if } t_1[B_i] = i \text{ or } 0 \text{ if } t_1[B_i] = i + 1, \quad \text{for all } i = 1 \cdots n \tag{18}$$

satisfies at least k clauses in Φ.

Now suppose that we have a solution $x_1, x_2, \ldots, x_l$ for max-3sat problem which satisfies at least k clauses. Obviously the assignment S:

$$t_1[B_i] = i \text{ if } x_i = 1 \text{ or } i + 1 \text{ if } x_i = 0, \quad \text{for all } i = 1 \cdots n, t_1[C_1] = t_1[C_2] = 1 \tag{19}$$

satisfies at least k queries in Q. □

We now show that *f* is a polynomial time function. For a formula ϕ with n variables and l clauses, we construct 2 tuples which have $n + 2$ attributes individually and l queries, each of which have 5 attributes. Thus *f* fulfills $2 * (n + 2) + 5 * l$ assignments, which can be done in polynomial time.

Theorem 4 *2-Equivalent Set Problem is NP-hard*

Proof In 3 we proved that Max-3Sat problem can be reduced to 2-Equivalent Set Problem in polynomial time. Furthermore, an answer to 2-Equivalent Set Problem obviously can be validate within polynomial time. Thus 2-Equivalent Set Problem is a NP-complete problem. □

Received: 26 April 2019 Accepted: 7 July 2019
Published online: 24 July 2019

References

1. Rahman MF, Liu W, Thirumuruganathan S, Zhang N, Das G. Privacy implications of database ranking. Proc VLDB Endow. 2015;8(10):1106–17.
2. Kim J, Winkler W. Multiplicative noise for masking continuous data. Statistics. 2003;1:9.
3. Aggarwal CC, Philip SY. A condensation approach to privacy preserving data mining. In: International conference on extending database technology. Berlin: Springer; 2004. pp. 183–199.
4. Verykios VS, Elmagarmid AK, Bertino E, Saygin Y, Dasseni E. Association rule hiding. IEEE Trans Knowl Data Eng. 2004;16(4):434–47.
5. Evfimievski A, Gehrke J, Srikant R. Limiting privacy breaches in privacy preserving data mining. In: Proceedings of the 22 ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems. New York: ACM; 2003. pp. 211–22.
6. Fienberg SE, McIntyre J. Data swapping: variations on a theme by dalenius and reiss. In: International workshop on privacy in statistical databases. Berlin: Springer; 2004. pp. 14–29.
7. Muralidhar K, Sarathy R. Data shuffling—a new masking approach for numerical data. Manag Sci. 2006;52(5):658–70.
8. Sweeney L. k-Anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl Based Syst. 2002;10(05):557–70.
9. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. l-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data. 2007;1(1):3.
10. Li N, Li T, Venkatasubramanian S. t-Closeness: Privacy beyond k-anonymity and l-diversity. In: IEEE 23rd international conference on data engineering, 2007. ICDE 2007. New Jersey: IEEE; 2007. pp. 106–15.
11. Sweeney L. Guaranteeing anonymity when sharing medical data, the datafly system. In: Proceedings of the AMIA annual fall symposium. Bethesda: American Medical Informatics Association; 1997. pp. 51.
12. Hundepool A, Willenborg L. μ-and τ-argus: software for statistical disclosure control. In: Third international seminar on statistical confidentiality; 1996.

13. Sweeney L. Towards the optimal suppression of details when disclosing medical data, the use of sub-combination analysis. Stud Health Technol Inf. 1998;2:1157.
14. Dwork C. Differential privacy. Encycl Cryptogr Secur. 2011;4877:338–40.
15. Dwork C, Roth A, et al. The algorithmic foundations of differential privacy. Found Trends® Theor Comput Sci. 2014;9(3–4):211–407.
16. Wasserman L, Zhou S. A statistical framework for differential privacy. J Am Stat Assoc. 2010;105(489):375–89.
17. Nabar SU, Kenthapadi K, Mishra N, Motwani R. A survey of query auditing techniques for data privacy. In: Privacy-preserving data mining. New York: Springer; 2008. pp. 415–31.
18. Marks DG. Inference in mls database systems. Knowl Data Eng IEEE Trans. 1996;8(1):46–55.
19. Achterberg T. Scip: solving constraint integer programs. Math Program Comput. 2009;1(1):1–41.
20. Wang JHYFW, Koperski JCWGK, Li D, Stefanovic YLARN, Zaiane BXOR. Dbminer: A system for mining knowledge in large relational databases. In: Proceedings of intlligence conference on data mining and knowledge discovery (KDD'96). 1996. pp. 250–5.
21. Bakken DE, Rarameswaran R, Blough DM, Franz AA, Palmer TJ. Data obfuscation: anonymity and desensitization of usable data sets. IEEE Secur Priv. 2004;2(6):34–41.
22. Zhu J, He P, Zheng Z, Lyu MR. A privacy-preserving qos prediction framework for web service recommendation. In: 2015 IEEE international conference on web services. New Jersey: IEEE; 2015. pp. 241–8.
23. McFee B, Lanckriet G. Metric learning to rank. In: Proceedings of the 27th international conference on machine learning (ICML'10). 2010.

Publisher's Note