Hindawi
Security and Communication Networks
Volume 2019, Article ID 5498375, 16 pages
https://doi.org/10.1155/2019/5498375

Research Article

Social Security and Privacy for Social IoT Polymorphic Value Set: A Solution to Inference Attacks on Social Networks

Yunpeng Gao[1] **and Nan Zhang**[2]

[1]*School of Engineering and Applied Science, George Washington University, Washington, DC, USA*
[2]*Kogod School of Business, American University, Washington, DC, USA*

Correspondence should be addressed to Yunpeng Gao; ypgao@gwu.edu

Received 27 April 2019; Accepted 4 August 2019; Published 28 August 2019

Guest Editor: Houbing Song

Social Internet of Things (SIoT) integrates social network schemes into Internet of Things (IoT), which provides opportunities for IoT objects to form social communities. Existing social network models have been adopted by SIoT paradigm. The wide distribution of IoT objects and openness of social networks, however, make it more challenging to preserve privacy of IoT users. In this paper, we present a novel framework that preserves privacy against inference attacks on social network data through ranked retrieval models. We propose PVS, a privacy-preserving framework that involves the design of polymorphic value sets and ranking functions. PVS enables polymorphism of private attributes by allowing them to respond to different queries in different ways. We begin this work by identifying two classes of adversaries, authenticity-ignorant adversary, and authenticity-knowledgeable adversary, based on their knowledge of the distribution of private attributes. Next, we define the measurement functions of utility loss and propose PVSV and PVST that preserve privacy against authenticity-ignorant and authenticity-knowledgeable adversaries, respectively. We take into account the utility loss of query results in the design of PVSV and PVST. Finally, we show that PVSV and PVST meet the privacy guarantee with acceptable utility loss in extensive experiments over real-world datasets.

1. Introduction

1.1. Motivation. SIoT integrates social network schemes into IoT systems, which provides opportunities for IoT objects to form social networks. The SIoT paradigm is promising as it is believed that SIoT structures are helpful in enhancing the navigability of IoT networks, identifying levels of trustworthiness and reusing existing social network models [5]. In this scenario, privacy and security issues have been extensively studied [6, 7, 24, 48]. However, current studies in privacy preservation of IoT systems focus on access control [23, 46], communication and authentication protocols [4, 34, 44], and attribute-based encryption [39, 43]. The features of social network have not been thoroughly considered.

The nature of online social networks (OSN) requires sharing of information. User information, including activity patterns and descriptive attributes, is mined and analyzed to improve user experience of OSN applications. Third party users also take advantage of the huge amount of data collected by social networks [21]. As part of the improvement of user experience, ranked retrieval models have been extensively studied and applied to many OSN features, e.g., link prediction and recommendation systems [32, 36, 47]. For instance, given a user's information (which can be a tuple in a database), a ranked retrieval model returns a ranking result that serves OSN features, e.g., "People you may know" and "Recommended for you." Furthermore, many OSN providers improve the accuracy of ranked results by taking into account private attributes of users in the ranked retrieval model. For example, sensitive demographics such as race, religion, and income can help in friend recommendation features as intuitively people sharing similar demographics are more likely to be interested in each other.

OSN providers relieve users' concern of privacy leakage by allowing them to mark attributes as "private" and hiding private attributes from profiling pages and ranked results. Users believe that their privacy is well protected since private attributes are invisible to the public in their profiles or any ranked results. However, Rahman et al. [35] proposed Rank Inference and showed that privacy of private attributes in the ranked retrieval model is not guaranteed. In their approach, the value of a private attribute can be inferred through a ranked retrieval interface given the premises that the domain of the private attribute is finite and that the ranking function has both monotonicity property and additivity property [35]. To be more specific, given monotonicity and additivity conditions, an adversary is always able to find a pair of differential queries, q_θ and q'_θ, such that (a) q_θ and q'_θ share the same predicate on all attributes except for a private attribute B_1, (b) the value of B_1 in q'_θ is θ while the value of B_1 in q_θ is not, and (c) the ranked result of q_θ contains the victim tuple v while the ranked result of q'_θ does not. Rahman et al. showed that given the above conditions, the adversary was able to conclude that the value of v's B_1 is not equal to θ. Furthermore, the adversary was able to infer the value of v's B_1 by finding more differential queries and excluding more values from the domain of B_1, as long as the domain is finite.

Privacy issues arise when private attributes of users are taken into account in the ranked retrieval model. Intuitively, the issues can be solved by removing private attributes from ranking functions, which decreases the utility of many OSN features, the recommendation system cannot provide accurate results. From OSN providers' perspective, removing private attributes is not a practical solution. Therefore, this work aims not only to address the issue of rank inference but also to propose a framework that preserves the privacy of users against all inference attacks through the ranked retrieval model while minimizes utility loss.

1.2. Related Work. Encryption technologies have been used throughout history in security and privacy preservation. Searching on encrypted data [10, 38] has been introduced to ensure data privacy. Cao et al. [9] proposed a scheme that allows privacy-preserving ranked search over encrypted data. A query consisting of multiple keywords is conducted by searching over encrypted documents with secure k-nearest neighbor (kNN) technique. Chen et al. [11] took into consideration correlations between documents before conducting a search query and achieved better performance. Vertical fragmentation [13, 15, 16, 22] has been applied to encrypted data, which hides identities of users by separating identifier attributes with descriptive attributes. However, encrypted searching is developed to preserve privacy against adversaries in a cloud computing environment. Adversaries can still access decrypted searching results from which private attribute values can be inferred.

As ONS has been emerging as an important source for big data, many studies have been carried out for privacy-preserving data mining (PPDM) and privacy-preserving data publishing (PPDP). Perturbative methods implement the "camouflage" paradigm where original data are directly modified [28]. Agrawal et al. [3] proposed an algorithm that perturbs data with random additive noise. Liu et al. proposed data perturbation with multiplicative noise. However, random noise has predictable structures in the spectral domain, and thus, privacy provided by additive noise is questionable [25]. Furthermore, additive or multiplicative noise can only be applied to numerical data. Data swapping approaches [19, 30, 31] perturb data by swapping values between records that are close to each other. Other distance-based approaches include [12] in which data points are perturbed without changing their relevant closeness relationships and [2] in which data points are clustered and each data point's value is replaced by the value of the cluster center. However, those approaches rely on a universal measurement of closeness between data points in multidimensional space. Furthermore, they are limited by the distributions of data points.

Generalization is the process of replacing a group of values with a more general value that can represent the group. Suppression is the ultimate state of generalization such that the representative value is "not applicable" and as a result, the group of values is removed from the dataset [45]. k-anonymity [40, 41] is a widely studied approach that preserves privacy of records by grouping at least k records into an equivalence class. The attribute values of the k records are suppressed so that the k records are indistinguishable from adversaries. Machanavajjhala et al. [27] proposed L-diversity that focuses on attribute privacy. L-diversity forces each equivalent class to have at least l different values for each attribute. Li et al. [26] proposed T-closeness that further considers the distribution of attribute values. T-closeness sets a threshold for the variance between the distribution of a private attribute in each equivalence class and the distribution of the same attribute in the entire dataset. However, those approaches are developed to preserve privacy of published data, while the ranked retrieval model of ONS does not directly reveal private attributes to the public. Furthermore, suppression of private attribute values introduces unnecessary utility loss to the ranked retrieval model. Equivalent Set [20] was proposed to preserve privacy against inference attacks through the ranked retrieval model. This approach groups different tuples into a set such that they are indistinguishable in ranked results. However, this approach requires that tuples in the same equivalent set have different values in every private attribute and have the same value in every public attribute. Assume that a kNN query q is sent to a ranked retrieval interface protected by Equivalent Set and that a tuple t is equal to q in most private attributes. Since the other tuples in the same equivalent set of t are different from t in every private attribute, they must be different from q in most private attributes too. Therefore, the original rank of t given q should be much higher than that of any other tuple in the same equivalent set. In this case, the rank of t given q will be significantly lowered by Equivalent Set in order to achieve indistinguishability, which reduces the accuracy of the ranked result of q. Furthermore, it is possible that there is no t' such that t' is different from t in every private attribute and is same with t in every public attribute. In this case, we have to suppress the private attribute values of t, which further introduces utility loss.

Differential privacy [8, 17, 18] is another widely studied framework that preserves privacy of published datasets or hidden databases. It imposes a strong guarantee of privacy on tuples in statistical databases by adding noise to the process of query results. However, the ranked retrieval model outputs ranks of tuples, instead of their values or aggregate statistics. We cannot directly add noise to ranked results as the rank of a tuple is determined by not only the tuple itself but also by other tuples in the database. Furthermore, the optimization of the ranked retrieval model has not been considered.

1.3. Contributions. This work presents a novel scheme for privacy-preserving the ranked retrieval model. We start with an introduction to the adversary model and introduce our definition of privacy guarantee. We identify two categories of adversaries based on their prior knowledge and assume that adversaries can launch optimal inference attacks through ranked results.

We propose the polymorphic value set (PVS), a privacy-preserving framework for the ranked retrieval model. Different from existing methods, PVS does not directly modify values of tuples or query results. Instead, PVS enables polymorphism of private attributes such that a private attribute of a tuple can respond to different queries in different ways. We prove that our framework meets the privacy guarantee stated in Problem Statement. For adversaries with and without prior knowledge, we design and implement the polymorphic value set with true values (PVST) and polymorphic value set with Virtual Values (PVSV), respectively. In the design of PVST and PVSV, we consider utility loss in the ranked retrieval model and propose a practical measurement of utility loss. We prove that the task of minimizing utility loss is NP-hard and present two heuristic algorithms that implement PVST and PVSV, respectively. We run our implementations of PVST and PVSV on a real-world dataset from eHarmony [29] that contains 486,464 tuples. The experiments yield excellent results with respect to privacy guarantee and utility loss.

The remainder of this paper is organized as follows. Problem Statement introduces the adversary model and the privacy guarantee. Privacy-Preserving Framework introduces our privacy-preserving framework. Framework with Virtual Values presents the design and implementation of PVSV, along with our analysis of utility loss. The implementation of PVST and analysis of utility loss are presented in Framework with True Values. Experimental Results contains our experimental evaluation of PVSV and PVST. In Conclusions, we conclude this paper with a summary of our key contributions and a discussion of some open problems.

2. Problem Statement

2.1. Ranked Retrieval Model. In information retrieval, we have witnessed extensive research in the ranked retrieval model. Unlike the Boolean retrieval model where only results that exactly match the predicates can be returned, the ranked retrieval model allows users to retrieve a list of records sorted by a proprietary ranking function. Therefore, the ranked retrieval model provides an alternative solution for users seeking results sorted by their relevance to the query.

As discussed in the introduction, many OSN applications have been using the ranked retrieval model to process incoming queries. Upon a query q, the ranked retrieval model would calculate each tuple t's score according to a proprietary score function $s(t \mid q)$ and return top-k tuples in descending order of their scores. The attributes of tuples could be either categorical or numerical. In this paper, we consider only categorical data, which does not limit the scope of our research. Actually, numerical data can be treated as categorical data by categorizing the numerical domain into small intervals such that no more than one tuple in the database falls into the same interval. Without loss of generality, we also assume that there is no duplicate tuple that is equal to another tuple in every attribute.

We now formalize our ranked retrieval model with categorical attributes. Consider an n-tuple database D with m public attributes $A_1, \ldots, A_m$ and m' private attributes $B_1, \ldots, B_{m'}$. Let V_i^A(resp. V_j^B) denote the value domain of A_i(resp. B_j). Let $t[A]_i$(resp. $t[B]_j$) denote the value of t in A_i(resp. B_j). Upon query q, the score function $s(t \mid q)$ computes a score for each tuple $t \in D$. The ranked retrieval model will then sort all tuples in D in the descending order and return them as the ranked result. We consider the case where the score function is linear. Therefore, $s(t \mid q)$ can be defined as

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

where w_i^A(resp. w_j^B) $\in (0, 1)$ is the weight of attribute A_i(resp. B_j) in the score function, and the matching function $\rho(q[A_i], t[A_i])$, (resp. $\rho(q[B_i], t[B_i])$) indicates if t matches q in attribute A_i (resp. B_j). Therefore, the value of $\rho(\beta_1, \beta_2)$ is 1 if β_1 is equal to β_2, and the value of $\rho(\beta_1, \beta_2)$ is 0 if β_1 is not equal to β_2. Note that our ranked retrieval model satisfies the monotonicity and additivity properties defined in [35].

2.2. Adversary Model. In Motivation, we mentioned that we do not make any assumption about the method adopted by an adversary when attacking a database. We also assume that the adversary has prior knowledge about the metadata of tables in the database, as well as the proprietary ranking function. Furthermore, the adversary is assumed to be able to issue queries to the ranked retrieval model, view ranked results, and insert tuples to the database. As a result, the adversary is able to retrieve all public attribute values by crawling the database through the query interface [37]. We denote the set of queries issued by the adversary as Q_A, the set of tuples inserted by the adversary as I_A, and the corresponding set of ranked results as R_D. R_D is fully determined by Q_A and I_A given fixed D. We name all information regarding a tuple $t \in D$ that an adversary can find in R_D as the *trace* of t and denote the trace of t as T_t. The trace of t includes, but not limited to, the rank of t and the relationship between t and any other tuple (e.g., t has a higher or lower rank than another tuple t') in a ranked result. Therefore, given fixed D, I_A and Q_A, T_t is fully determined by the attribute values of t.

Another capability of the adversary we model is the adversary's prior knowledge. Consider an extreme case where the adversary knows the equivalence relation between two attributes B_1 and A_1. In this case, even without R_D, the adversary is still able to infer the value of B_1 of any $t \in D$. In reality, an adversary can acquire such attribute correlations by being or consulting an expert in the domain of the dataset or by adopting data mining methods [42]. For example, based on the personal information (e.g., gender, ethnicity, age, and blood type which can be used to infer the gene) stored as public attributes and published in public medical data repositories, genetic epidemiologists can generally conclude that an individual does not have some diseases, merely based on the fact that these diseases would never be found by the candidate gene among historic medical datasets. Therefore, an adversary with prior knowledge could eliminate the possibility of a certain tuple in the database. We model prior knowledge as a function $\mathrm{PK}(t \mid D)$ that takes as input t and D and returns either 0 or 1. $\mathrm{PK}(t \mid D) = 0$ indicates that, given prior knowledge, the possibility of $t \in D$ is zero. $\mathrm{PK}(t \mid D) = 1$ indicates that the adversary cannot eliminate the possibility of $t \in D$. As prior knowledge helps adversaries in launching an inference attack, adversaries can be partitioned into two classes: adversaries with prior knowledge and adversaries without prior knowledge of the dataset.

Definition 1. We name adversaries without prior knowledge of the authenticity of any tuple as authenticity-ignorant adversaries. For authenticity-ignorant adversaries, $\mathrm{PK}(t \mid D)$ always outputs 1. We name adversaries with such prior knowledge as authenticity-knowledgeable adversaries. For authenticity-knowledgeable adversaries, $\mathrm{PK}(t \mid D) = 1$ if $t \in D$ and $\mathrm{PK}(t \mid D) = 0$ if $t \notin D$.

The objective of both classes of adversaries is to maximize the following $g(v, B_j)$ value when inferring the value of victim tuple v in attribute B_j:

$$g\left(v, B_j\right) = \Pr\left(v\left[B_j\right] = a\right), \tag{2}$$

where $\Pr(v[B_j] = a)$ is the probability of $v[B_j] = a$ and a is the value inferred by the adversary given prior knowledge and ranked results.

For authenticity-ignorant adversaries, $\mathrm{PK}(t \mid D)$ always outputs 1 regardless of t and D. We only assume the cases where users input true information to the databases. Therefore, for authenticity-knowledgeable adversaries, $\mathrm{PK}(t \mid D) = 1$ if $t \in D$ and $\mathrm{PK}(t \mid D) = 0$ if $t \notin D$. In this paper, we assume a strong adversary that can infer the private attribute values of an arbitrary tuple, given the premise that the trace of the target tuple is unique. The premise can be easily met because as long as there is no duplicate tuple in the dataset, the adversary can always construct well-designed I_A and Q_A such that the trace of the target tuple is different from the trace of any other tuple. Therefore, the adversary can always find a such that $\Pr(v[B_j] = a) = 100\%$.

2.3. Problem Statement. A privacy breach can be described by a successful inference of a private attribute value in the database. We view privacy of $v[B_j]$ as the upper bound on the possibility that an adversary succeeds in inferring the value of $v[B_j]$. Note that we do not make any assumption about the adversary's attacking method. Our objective in this paper is to present a framework that sets an upper bound on the probability of successful inference of an arbitrary private attribute for tuple $t \in D$. Therefore, we define the objective of the framework as

$$\forall t \in D, j \in \{1, \ldots, m'\}, g\left(v, B_j\right) \leq \varepsilon. \tag{3}$$

We present the upper bound ϵ as our privacy guarantee.

However, a privacy-preserving framework should provide not only a privacy guarantee but also a notion of utility—after all, a framework that removes all private attribute values or replaces them with randomly generated values can surely preserve privacy. Therefore, we use a measurement based on the variance of ranked results before and after adopting our framework. Given D and a set of all possible queries denoted as Q, we define the utility loss for our ranked retrieval model as follows:

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

where $\mathrm{Rank}(t \mid q)$ and $\mathrm{Rank}'(t \mid q)$ refer to the ranks of tuple t in the ranked result given query q before and after applying our frameworks, respectively.

3. Privacy-Preserving Framework

The only information an adversary can obtain from a database through the ranked retrieval model is public attribute values and ranked results. For an adversary without prior knowledge, information regarding private attribute values can only be retrieved from ranked results. Therefore, in order to preserve privacy, we have to modify the ranked retrieval model such that the adversary cannot retrieve any useful information about private attributes from ranked results.

An idea is to group different tuples together in ranked results. As in our prior work [20], we can group two tuples v_1 and v_2 together such that they share the same rank in any ranked result. This can be achieved by adopting a new ranking function $s'(t \mid q)$ such that $s'(v_1 \mid q) = s'(v_2 \mid q)$ for all q. If v_1 and v_2 have different values on every private attributes, then the adversary is unable to infer the private values of v_1 since v_1 and v_2 are indistinguishable in any ranked results. However, this method suffers from high utility loss. In order to preserve the privacy of all private attributes, v_1 and v_2 have to be different over all private attributes. Thus, the original scores of v_1 and v_2, i.e., $s(v_1 \mid q)$ and $s(v_2 \mid q)$, differ a lot, which leads to a higher variance between the rank of v_1 before and after adopting this method.

In this work, we present a novel framework that preserves privacy of private attributes while minimizes the utility loss. We observe that for a tuple v's private attribute $v[B_j]$ if there are at least two potential values for $v[B_j]$ and an adversary cannot differentiate any one of them, then the privacy of $v[B_j]$ can be preserved. For instance, if the probability of $v[B_j] = a$ is equal to the probability of $v[B_j] = a'$, given ranked results and prior knowledge, then the adversary cannot exclude any one of them. If both the probabilities are 50%, then the adversary may choose to randomly pick a value from a and a' as the inferred result. In this case, $g(v, B_j)$ will not exceed 50% and privacy of $v[B_j]$ can be preserved. To prove this statement, suppose that v is an arbitrary tuple in database D, and we want to preserve the privacy of $v[B_j]$. Let β_j^B be an arbitrary value in $V_j^B \backslash \{v[B_j]\}$. We construct tuple v' such that v' and v differ in only one attribute B_j: $v'[B_j] = \beta_j^B$. We also construct database D' such that D and D' differ in only one tuple $v \in D$ while $v' \in D'$. We define a new score function $s'(t \mid q)$:

$$s'(t \mid q) = \sum_{i=1}^{m} w_i^A \rho(q[A_i], t[A_i]) + \sum_{j=1}^{m'} w_i^B \rho'(q[B_j], t[B_j]), \tag{5}$$

where

$$\rho'(q[B_j], t[B_j]) = \rho(q[B_j], t[B_j]), \quad \text{if } t \neq v \text{ and } t \neq v',$$
$$\rho'(q[B_j], t[B_j]) = \text{Max}\left(\rho(q[B_j], v[B_j]), \rho(q[B_j], v'[B_j])\right), \tag{6}$$

if $t = v$ or $t = v'$

Imagine a case where an adversary queries D and D' with the same query workload Q_A. We denote the ranked results from D as R_D and the ranked results from D' as $R_{D'}$. As in the new score function, $s'(v \mid q) = s'(v' \mid q)$ for $\forall q \in Q_A$, R_D is identical to $R_{D'}$. Therefore, given only ranked results, the adversary cannot tell the difference between the two databases being queried. Furthermore, even if we exchange the values of v and v', the adversary still cannot observe any change in R_D or $R_{D'}$. As a result, the value of $v[B_j]$ and β_j^B are equivalent from the perspective of the adversary, and the privacy of $v[B_j]$ can be well preserved. Intuitively, v can be seen as a tuple that has two polymorphic forms in B_j: $v[B_j]$ and β_j^B. When calculating the score of v with $s'(t \mid q)$, we always choose the value that can maximize $s'(v \mid q)$.

We can further extend the statement to a more general case. For each tuple $v \in D$ and each private attribute B_j, we can select e distinct values $\beta_1, \beta_2, \ldots, \beta_e$ from $V_j^B \backslash \{v[B_j]\}$, $e < |V_j^B| - 1$. The new score function can be defined as

$$s'(t \mid q) = \sum_{i=1}^{m} w_i^A \rho(q[A_i], t[A_i]) + \sum_{j=1}^{m'} w_i^B \rho'(q[B_j], t[B_j]), \tag{7}$$

where

$$\rho'(q[B_j], t[B_j]) = \rho(q[B_j], t[B_j]), \quad \text{if } t \neq v,$$
$$\rho'(q[B_j], t[B_j]) = \text{Max}\left(\rho(q[B_j], t[B_j]), \rho(q[B_j], \beta_1), \ldots, \rho(q[B_j], \beta_e)\right), \quad \text{if } t = v. \tag{8}$$

Consider e tuples $v(1)', \ldots, v(e)'$ which are identical to v in all attributes except for B_j. Let $v(i)'[B_j]$ be β_i, $\forall i = 1, \ldots, e$. Then for $\forall q$, we have $s'(v \mid q) = s'(v(i)' \mid q)$. As a result, from the adversary's perspective, there are $e + 1$ potential values for $v[B_j]$: $v[B_j], \beta_1, \ldots, \beta_e$, which are indistinguishable from each other from any ranked results. The privacy of $v[B_j]$ can be preserved by grouping it with e equivalent values.

As such, we introduce the construction of the polymorphic value set (PVS). We put $v[B_j]$ into a set in which all values are indistinguishable when calculating the score with respect to B_j in the ranking function, i.e., $\rho'(q[B_j], v[B_j])$. We name the above set as the polymorphic value set and denote the polymorphic value set of tuple v in attribute B_j as $P_v^{B_j}$. We define $P_v^{B_j}$ as follows.

Definition 2. $P_v^{B_j}$ is a set containing all indistinguishable values of tuple v's private attribute B_j. Assigning $v[B_j]$ with an arbitrary value in $P_v^{B_j}$ will not change the value of $s'(v \mid q)\ \forall q$, i.e., $\rho'(q[B_j], v[B_j]) = \rho'(q[B_j], \beta), \forall \beta \in P_v^{B_j}$ and $\forall q$.

Since the adversary cannot distinguish different values in $P_v^{B_j}$ by launching any inference attacks based only on ranked results, the privacy guarantee of $v[B_j]$ is

$$\varepsilon = \frac{1}{\left|P_v^{B_j}\right|}. \tag{9}$$

In order to achieve the privacy guarantee defined in (3), for each $v \in D$ and each private attribute B_j, we have to ensure that (a) there is one and only one polymorphic value set $P_v^{B_j}$ for v's attribute B_j and (b) the privacy guarantee defined in (9) is always valid.

4. Framework with Virtual Values

4.1. Design. In this section, we introduce how polymorphic value sets can be constructed with generated values to meet the privacy guarantee in (9) against authenticity-ignorant adversaries. We name values generated by our framework as virtual values.

As we proved in Privacy-Preserving Framework, an authenticity-ignorant adversary cannot distinguish the value of $v[B_j]$ from other valid values in $P_v^{B_j}$, given ranked results. Since an adversary without prior knowledge cannot validate the authenticity of any value in $P_v^{B_j}$, all values in $P_v^{B_j}$ are "valid" in the perspective of the adversary, no matter if they are generated by our framework or collected from real data in D. Therefore, we observe that $P_v^{B_j}$ can be formed by any values in V_j^B.

In order to achieve the privacy guarantee of ε,, we have to ensure that $|P_v^{B_j}| > (1/\varepsilon)$, $\forall v \in D$, and $\forall j \in \{1, \ldots, m'\}$. An intuitive algorithm to generate $P_v^{B_j}$ of size $1/\varepsilon$ is to randomly pick $(1/\varepsilon) - 1$ values from $V_j^B \backslash v[B_j]$. Specifically, let the initial $P_v^{B_j} = \{v[B_j]\}$. Then, we can randomly pick distinct values from $V_j^B \backslash v[B_j]$ and insert them into $P_v^{B_j}$ until $P_v^{B_j}$ contains at least $(1/\varepsilon)$ distinct values. In the same manner, we can construct a polymorphic value set for each tuple's each private attribute.

4.1.1. Privacy Guarantee. For a database D where every tuple v's every private attribute B_j is included by one polymorphic value set with virtual values (PVSV) whose size is at least l, if the adversary has no prior knowledge of D, a privacy level of $\varepsilon = (1/l)$ is achieved.

For an authenticity-ignorant adversary, $\mathrm{PK}(v \mid D) = 1$ $\forall v$. As proved in Privacy-Preserving Framework, for an authenticity-ignorant adversary, it is impossible to distinguish $v[B_j]$ with at least $l-1$ other values. Thus we have $g(v, B_j) \le (1/l)$ for $\forall v \in D$ and $\forall j \in \{1, \ldots, m'\}$. According to equation (3), a privacy guarantee of $(1/l)$ can be achieved.

4.2. Utility Optimization. In this section, we discuss how to reduce utility loss caused by polymorphic value sets. We introduced a metric of utility loss in (4) that calculates the sum of difference in ranked results given all possible queries. To practically calculate utility loss, we limit the range of queries to a finite set named *query workload*. In practice, the query workload of a database D can be a set of queries that are more frequently issued than any other queries. A query workload may contain duplicate queries, which reflect the distribution of frequent queries. With a query workload, denoted as Q, we can define the practical utility loss as

$$U_Q = \sum_{q\in Q}\sum_{v\in D}\left|\mathrm{Rank}(v \mid q) - \mathrm{Rank}'(v \mid q)\right|. \tag{10}$$

In order to reduce utility loss, we have to find assignments of $P_v^{B_j}$ for $\forall v \in D$ and $\forall j \in \{1, \ldots, m'\}$ such that the overall U_Q can be minimized. Without loss of generality, we only consider constructing polymorphic value sets of size 2. In this case, the privacy guarantee is 1/2. For each $v[B_j]$, we need to find a value that is indistinguishable from $v[B_j]$. We denote the polymorphic value of $v[B_j]$ as $v'[B_j]$.

Definition 3. We define the 2-PVSV problem as follows: given database D and query workload Q, find a polymorphic value $v'[B_j]$ from $V_j^B \backslash \{v[B_j]\}$ for each $v[B_j]$, $\forall v \in D$ and $\forall j \in \{1, \ldots, m'\}$, such that U_Q defined in (10) is minimized.

Theorem 1. *The 2-PVSV problem is NP-hard.*

The proof of Theorem 1 in detail can be found in Appendix A.

4.3. Heuristic Algorithm. We have proved that the 2-PVSV problem is an NP-hard problem that may not be solved in polynomial time. Therefore, we propose PVSV-Constructor, a heuristic algorithm that can return an approximate solution in polynomial time.

We observe that $|s(v \mid q) - s'(v \mid q)|$ is relevant to $|\mathrm{Rank}(v \mid q) - \mathrm{Rank}'(v \mid q)|$. A smaller difference between $s(v \mid q)$ and $s'(v|q)$ leads to a smaller difference between $\mathrm{Rank}(v \mid q)$ and $\mathrm{Rank}'(v \mid q)$. Therefore, $|\mathrm{Rank}(v \mid q) - \mathrm{Rank}'(v \mid q)|$ can be approximately minimized by a solution that minimizes $|s(v \mid q) - s'(v \mid q)|$. As such, we propose an approximation of U_Q that calculates the score difference before and after adopting our framework. We denote the score difference as U_Q^S and define U_Q^S as follows:

$$U_Q^S = \sum_{q\in Q}\sum_{v\in D}\left|s(v \mid q) - s'(v \mid q)\right|. \tag{11}$$

As shown in Algorithm 1, given input database D, the number of public and private attributes m and m' respectively, query workload Q and privacy guarantee ϵ, PVSV-Constructor constructs an equivalent value set for each $v \in D$ and $j \in \{1, \ldots, m'\}$ that minimizes U_Q^S. Recall the definition of $s'(t|q)$ in (7), and we have

$$U_Q^S = \sum_{v\in D}\sum_{j=1}^{m'}\sum_{q\in Q} w_j^B\left|\rho'\left(q[B_j], v[B_j]\right) - \rho\left(q[B_j], v[B_j]\right)\right|. \tag{12}$$

We denote the score difference of $v[B_j]$ contributed by $P_v^{B_j}$ as $U(P_v^{B_j})$.

$$U\left(P_v^{B_j}\right) = \sum_{q\in Q} w_j^B\left|\rho'\left(q[B_j], v[B_j]\right) - \rho\left(q[B_j], v[B_j]\right)\right|. \tag{13}$$

Therefore, U_Q^S is the sum of $U(P_v^{B_j})$ over all tuples and private attributes:

$$U_Q^S = \sum_{v\in D}\sum_{j=1}^{m'} U\left(P_v^{B_j}\right). \tag{14}$$

Since the construction of each $P_v^{B_j}$ is independent from other polymorphic value sets, we can minimize U_Q^S by minimizing the score difference contributed by each $P_v^{B_j}$ for $\forall j \in \{1, \ldots, m'\}$ and $\forall v \in D$. Note that $|\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])| = 0$ if $q[B_j] = v[B_j]$ or $q[B_j] \notin P_v^{B_j}$. Also note that $|\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])| = 1$ when $q[B_j] \neq v[B_j]$ and $q[B_j] \in P_v^{B_j}$. Therefore, if $\forall q \in Q$, $v[B_j] = q[B_j]$, then any assignment of $P_v^{B_j}$ cannot contribute to a higher $U(P_v^{B_j})$ since $\rho'(q[B_j], v[B_j]) = 1$, $\rho(q[B_j], v[B_j]) = 1$, and $|\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])|$ is always zero. In this situation, $U(P_v^{B_j}) = 0$. On the contrary, if $\exists q \in Q$, $v[B_j] \neq q[B_j]$, then $\rho(q[B_j], v[B_j]) = 0$, and thus, $|\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])| = \rho'(q[B_j], v[B_j])$. According to equation (13), the value of $U(P_v^{B_j})$ is

$$\begin{aligned}\sum_{q\in Q} w_j^B\left|\rho'\left(q[B_j], v[B_j]\right) - \rho\left(q[B_j], v[B_j]\right)\right| \\ = \sum_{q\in Q, q[B_j]\neq v[B_j]} w_j^B \rho'\left(q[B_j], v[B_j]\right).\end{aligned} \tag{15}$$

In order to minimize $U(P_v^{B_j})$, we have to find an assignment of $P_v^{B_j}$ which minimizes $|\{q \in Q \mid q[B_j] \neq v[B_j], \exists \beta \in P_v^{B_j}, q[B_j] = \beta\}|$. Consider the simplest case where we want to construct $P_v^{B_j}$ of size 2: $\{v[B_j], \beta\}$. We have

```
 (i) Input: $\varepsilon$, $D$, $m$, $m'$, $Q$
(ii) Output: $P_v^{B_j}$, $\forall v \in D$ and $\forall j \in \{1, \ldots, m'\}$
 (1) $k = (1/\varepsilon) - 1$
 (2) for $t$ in $D$ do
 (3)     for $j \in \{1, \ldots, m'\}$ do
 (4)         $P_v^{B_j} = \{v[B_j]\}$
 (5)         Set $= V_j^B \backslash \{v[B_j]\}$
 (6)         Sort elements of Set by their frequencies in $\{q[B_j] | q \in Q\}$ in ascending order.
 (7)         Remove the last $|\text{Set}| - k + 1$ elements in Set.
 (8)         $P_v^{B_j} = P_v^{B_j} \cup \text{Set}$
 (9)     end
(10) end
```

Algorithm 1: PVSV-Constructor.

$$\sum_{q \in Q, q[B_j] \neq v[B_j]} w_j^B \rho'\left(q\left[B_j\right], v\left[B_j\right]\right) = w_j^B \left|\left\{q \in Q \mid q\left[B_j\right] = \beta\right\}\right|. \tag{16}$$

In this case, β has to be a value in $V_j^B \backslash v[B_j]$ such that β has the lowest frequency among all $q[B_j]$ values for $q \in Q$. Note that β does not have to be an element of $\{q[B_j] \mid q \in Q\}$. If $\beta \notin \{q[B_j] \mid q \in Q\}$, its frequency is 0. Similarly, if we want to construct $P_v^{B_j}$ of size k, then we should insert the $k-1$ least frequent values among all $q[B_j]$ values into $P_v^{B_j}$.

In line 4 of Algorithm 1, we initialize the polymorphic value set of $v[B_j]$ by inserting $v[B_j]$ into $P_v^{B_j}$. For $v[B_j]$, a set *Set* is constructed from $V_j^B \backslash \{v[B_j]\}$, which contains all values that can be inserted into $P_v^{B_j}$. In order to minimize $U(P_v^{B_j})$, we insert the $k-1$ least frequent values from Set into $P_v^{B_j}$ by their frequencies in $\{q[B_j] \mid q \in Q\}$. The above construction of $P_v^{B_j}$ is repeated for each $t \in D$ and $j \in \{1, \ldots, m'\}$. The computational complexity of Algorithm 1 is $O(|D| \cdot |Q| \cdot m')$.

5. Framework with True Values

5.1. Authenticity-Knowledgeable Adversaries. As we mentioned in the adversary model, an authenticity-knowledgeable adversary is able to tell if $t \in D$ is possible. As a result, the authenticity-knowledgeable adversary can launch a more efficient attack on private attributes by examining the authenticity of values learned from R_A. We show a simple case where an authenticity-knowledgeable adversary breaks the privacy guarantee provided by polymorphic values sets constructed with virtual values. Consider that the objective of the adversary is to infer the value of $v[B_0]$ and B_0 is the only private attribute in D. $P_v^{B_0}$ is the polymorphic value set generated for $v[B_0]$ and $|P_v^{B_0}| = 1/\varepsilon$. Without prior knowledge, $P_{v[B_0]} = \varepsilon$ and the privacy guarantee is achieved. However, for a value $\beta \in P_v^{B_0} \backslash \{v[B_0]\}$, if the adversary can conclude that $\text{PK}(v' \mid D) = 0$ for any tuple v' such that v' is equal to v in all public attributes and $v'[B_0] = \beta$, then the adversary can exclude β from $P_v^{B_0}$. Therefore,

$$g\left(v\left[B_0\right]\right) = \frac{\varepsilon}{1-\varepsilon} > \varepsilon, \tag{17}$$

and equation (9) no longer holds.

As shown above, if values in $P_v^{B_0}$ are marked by an adversary as invalid for v given $\text{PK}(v' | D)$, then the adversary can successfully break the privacy guarantee defined in framework with virtual values.

5.2. Design. In this section, we propose polymorphic value sets with true values (PVST) that construct polymorphic value sets with values that cannot be excluded by $\text{PK}(t \mid D)$. PVST considers nontrivial prior knowledge of adversaries and presents the same degree of privacy guarantee introduced in (9).

We have shown above that the implementation with virtual values can be compromised by adversaries with prior knowledge. Consider a tuple $v \in D$. Given privacy guarantee ϵ, we construct m' polymorphic values sets $P_v^{B_0}, \ldots, P_v^{B_{m'}}$ for each private attributes of v where $|P_v^{B_j}| \geq 1/\varepsilon$. Let set M_v^A be

$$M_v^A = \{v[A_0]\} \times \cdots \times \{v[A_m]\} \times P_v^{B_0} \times \cdots \times P_v^{B_{m'}}. \tag{18}$$

M_v^A is the Cartesian product of sets each of which contains v's all equivalent values in an attribute. For public attribute A_i, the corresponding set is $\{v[A_i]\}$ since public attribute values are open to the adversary. For private attribute B_j, the corresponding set is $P_v^{B_j}$. Therefore, M_v^A contains all possible tuples that are indistinguishable with v with respect to R_A (including v itself).

With prior knowledge on $\text{PK}(t \mid D)$, an adversary is able to exclude a value β from $P_v^{B_j}$ if

$$\forall t \in \left\{t \mid t \in M_v^\text{A}, t\left[B_j\right] = \beta\right\}, \text{PK}(t \mid D) = 0. \tag{19}$$

As described above, it is safe for the adversary to conclude that $\beta \neq v[B_j]$, if there is no $t \in M_v^A$ such that $t[B_j] = \beta$ and $\text{PK}(t \mid D) = 1$. Alternatively, if for every β in $P_v^{B_j}$, there is a t such that $t[B_j] = \beta$ and $\text{PK}(t \mid D) = 1$, then the adversary cannot exclude any value in $P_v^{B_j}$, and therefore, $P_{v[B_j]} \leq \varepsilon$ is guaranteed. Since $\text{PK}(t \mid D) = 1$ iff $t \in D$, we construct polymorphic value sets with true values from $t \in D$.

Definition 4. If there exists l distinct values $\beta_1, \ldots, \beta_l \in P_v^{B_j}$ such that $\forall r \in \{1, \ldots, l\}$, β_r holds the following property:

$$\exists t \in M_v^{\mathrm{A}}, t[B_j] = \beta_r \text{ and } t \in D. \tag{20}$$

Then, we say that $P_v^{B_j}$ *covers* l true values. We denote $T_v^{B_j} = \{\beta_1, \ldots, \beta_l\}$ as the *true value set* of t in B_j.

Privacy guarantee: for a database D where every tuple's every private attribute is included by one equivalent value set which covers at least l true values, a privacy level of $\varepsilon = 1/l$ is achieved.

Assume that the adversary's objective is to infer the value of $v[B_j]$. As mentioned in the adversary model, we make no assumption on the attacking methods adopted by an adversary. Consider M_v^{A} defined in (18), $\forall t, t' \in M_v^A$ and $\forall q \in Q_{\mathrm{A}}, s(t \mid q) = s(t' \mid q)$. Therefore, the adversary cannot distinguish tuples in M_v^A by observing R_{A}. In this situation, the adversary would use $\mathrm{PK}(t \mid D)$ to exclude all tuple t in M_v^{A} such that $\mathrm{PK}(t \mid D) = 0$. However, as $P_v^{B_j}$ covers l true values, there exists l tuples $t_1, \ldots, t_l \in M_v^{\mathrm{A}}$ such that $\mathrm{PK}(t \mid D) = 1$ and $t_r[B_j] \neq t_s[B_j]$ for arbitrary $r, s \in \{1, \ldots, l\}$. As a result, values in $\{t_1[B_j], \ldots, t_l[B_j]\}$ are indistinguishable given R_{A} and $\mathrm{PK}(t \mid D)$. Thus, $P_{v[B_j]} = 1/l$. According to (3), a privacy level of $1/l$ is achieved.

5.3. Utility Optimization. An intuitive method of constructing $P_v^{B_j}$ is to insert the value of B_j of all tuples that share the same public attribute values with v into $P_v^{B_j}$, i.e., $P_v^{B_j} = P_v^{B_j} = \{v'[B_j] \mid \forall i \text{ and } v' \in D, v'[A_i] = v[A_i]\}$. The privacy guarantee is met if $P_v^{B_j}$ covers at least $1/\varepsilon$ true values. Nevertheless, utility loss cannot be ignored as in the intuitive method, $s(v' \mid q)$ would be the same for all $v' \in \{v' \mid \forall i$ and $v' \in D, v'[A_i] = v[A_i]\}$, and thus information of private attributes are missing in the ranked result of v'. Therefore, the size of $P_v^{B_j}$ is critical in balancing privacy and utility. With no loss of generality, we limit the size of each polymorphic value set to 2. We show that constructing such polymorphic value sets is an NP-hard problem.

Definition 5. We define the 2-PVST problem as follows: given database D and a query workload Q, construct $P_v^{B_j}$ of size 2 for each tuple $v \in D$, $\forall j \in \{1, \ldots, m'\}$, and minimize U_Q defined in (10).

Theorem 2. *The 2-PVST problem is NP-hard.*

The proof of Theorem 2 can be found in Appendix B.

5.4. Heuristic Algorithm. We have shown that the 2-PVST problem is NP-hard. In this subsection, we present PVST-Constructor, a heuristic algorithm that constructs PVST within polynomial time. PVST-Constructor tries to minimize $|P_v^{B_j}|$ and $\sum_{q \in Q} |s'(v \mid q) - s(v \mid q)|$ for each $v \in D$ and $j \in \{1, \ldots, m'\}$ with the greedy algorithm.

The pseudo-code of PVST-Constructor is shown in Algorithm 2. In lines 2 to 6, we initialize each $P_v^{B_j}$ with $\{v[B_j]\}$. Then, for each $v \in D$, PVST-Constructor constructs $P_v^{B_1}, \ldots, P_v^{B'_m}$ by finding a v' with the greedy algorithm and inserting $v'[B_j]$ into $P_v^{B_j}$, $\forall j \in \{1, \ldots, m'\}$. The above process will be taken multiple times until $|P_v^{B_j}| \geq 1/\varepsilon$. Since every v' is a real tuple existing in D and we insert at least k tuples in the above processes, $P_v^{B_j}$ covers at least $k+1$ true values, and thus, the privacy guarantee is met. As mentioned in Utility Optimization, the size of $P_v^{B_j}$ is critical in minimizing utility loss. Therefore, the heuristic algorithm tries to minimize the utility loss by minimizing the size of each $P_v^{B_j}$. We count the number of polymorphic value sets of v, if the set contains less than k values, that can be enlarged by inserting v''s private attribute values. We denote this count as $\mathrm{cover}_{v'}$:

$$\mathrm{cover}_{v'} = \left|\left\{ j \mid \left|P_v^{B_j}\right| < k, v'\left[B_j\right] \notin P_v^{B_j} \right\}\right|. \tag{21}$$

Tuple v' with a higher $\mathrm{cover}_{v'}$ value can enlarge the size of more polymorphic value sets of v, and thus, we can reduce the number of tuples that we have to insert into $P_v^{B_1}, \ldots, P_v^{B'_m}$.

Furthermore, we take into consideration queries in Q. In order to minimize U_Q, we have to minimize $|s'(v \mid q) - s(v \mid q)|$ for $q \in Q$. Note that for each attribute B_j, $|\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])| = 1$ if $v[B_j] \neq q[B_j]$ and $q[B_j] \in P_v^{B_j}$. We denote the value of $\sum_j |\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])|$ as $\mathrm{loss}_{v'}$. Thus, we have

$$\begin{aligned}\mathrm{loss}_{v'} &= \sum_{q \in Q} \sum_{j=1,\ldots,m'} \left|\rho'\left(q\left[B_j\right], v\left[B_j\right]\right) - \rho\left(q\left[B_j\right], v\left[B_j\right]\right)\right| \\ &= \sum_{q \in Q} \sum_{j=1,\ldots,m'} \delta(v, v', q, j),\end{aligned} \tag{22}$$

where $\delta(v, v', q, j) = 1$ if $v[B_j] \neq q[B_j]$ and $v'[B_j] = q[B_j]$ and $\delta(v, v', q, j) = 0$ otherwise. Therefore, tuple v' with a smaller $\mathit{loss}_{v'}$ can reduce the value of $\sum_j |\rho'(q[B_j], v[B_j]) - \rho(q[B_j], v[B_j])|$, and thus, we can reduce the value of U_Q.

The computation of $\mathrm{cover}_{v'}$ and $\mathrm{loss}_{v'}$ is done in line 11. Then, we compute $\mathrm{score}_{v'}$, the score of v' that indicates how preferable v' is, relative to other tuples in $D \backslash \{v\}$. We adopt the greedy algorithm to find the next v' for $P_v^{B_1}, \ldots, P_v^{B_{m'}}$, i.e., in each iteration, we choose the tuple that has the highest score value. In line 15, we insert the private attribute values of the chosen tuple (denoted as $v_{\max}$) into $P_v^{B_1}, \ldots, P_v^{B_{m'}}$. The above process is repeated until the sizes of $P_v^{B_1}, \ldots, P_v^{B_{m'}}$ are no less than k.

6. Experimental Results

6.1. Experimental Setup. To validate PVSV-Constructor and PVST-Constructor algorithms, we conducted experiments on a real world dataset [29] from eHarmony which contains 58 attributes and 486,464 tuples. We removed 5 noncategorical attributes and randomly picked 20 categorical attributes from the remaining 53 attributes. The domain sizes of the 20 attributes range from 2 to 15. After removing duplicate tuples, we randomly picked 300,000 tuples as our testing bed.

By default, we use the ranking function from the ranked retrieval model with all weights set to 1. All experimental results were obtained on a Mac machine running Mac OS

```
(i) Input: $\varepsilon$, $D$, $m$, $m'$, $Q$
(ii) Output: $P_v^{B_j}$, $\forall v$ and $\forall B_j$
(1) $k = (1/\varepsilon) - 1$
(2) for $v$ in $D$ do
(3)     for $j \in \{1, \ldots, m'\}$ do
(4)         $P_v^{B_j} = \{v[B_j]\}$
(5)     end
(6) end
(7) for $v$ in $D$ do
(8)     while $\exists j \in \{1, \ldots, m'\}, |P_v^{B_j}| < k + 1$ do
(9)         for $v' \in \{t \in D | \forall A_i, t[A_i] = v[A_i]\}$ do
(10)            compute $\text{cover}_{v'}$, $\text{loss}_{v'}$
(11)            $\text{score}_{v'} = \text{cover}_{v'} = 1/1 + \exp(-\text{loss}_{v'}) =$
(12)        end
(13)        $v_{\max} = \text{argmax}_{v' \in \{|t \in D | \forall A_i, t[A_i] = v[A_i]\}} \text{score}_{v'}$
(14)        for $j \in \{1, \ldots, m'\}$ do
(15)            $P_v^{B_j} = P_v^{B_j} \cup \{t_{\max}[B_j]\}$
(16)        end
(17)    end
(18) end
```

Algorithm 2: PVST-Constructor.

with 8 GB of RAM. The algorithms were implemented in *Python.*

6.2. Privacy. The privacy guarantee of PVSV-Constructor and PVST-Constructor were tested by performing Ranked Inference attack [35], including Point-Query, In-Query, Point-Query&Insert, and In-Query&Insert attacking methods, on the dataset. From a total of 20 attributes, 5 attributes were randomly chosen as public attributes and another 5 attributes were randomly chosen as private attributes. We randomly picked 20,000 distinct tuples from the dataset as the testing bed and randomly generated 10 tuples as the query workload. For PVSV-Constructor, we constructed a polymorphic value set of size 2 for each private attribute and each tuple in the testing bed. For PVST-Constructor, we constructed a polymorphic value set that covers at least two true values for each private attribute and each tuple. We randomly picked 1,000 tuples from the testing bed as our targets and performed 1,000 Rank Inference attacks (250 attacks for each of the four methods) on the five private attributes of target tuples. We measured the attack success guess rates based on the frequency of successful inference among all inference attempts. Figure 1 shows the success guess rates of Rank Inference attacks on the unprotected testing bed, the testing bed with PVSV, and the testing bed with PVST. As the size of each polymorphic value set is 2, the success guess rates on PVSV are around 50%, which are significantly lower than those of unprotected dataset. We also observe that the success guess rates on PVST are slightly lower than those of PVSV. The reason is that PVSV-Constructor will be inserting values into tuple v's polymorphic value sets until all v's polymorphic value sets cover at least 2 true values. Thus, some polymorphic value sets of v may contain more than 2 values.

6.3. Utility. In this subsection, we quantify utility loss of PVSV-Constructor and PVST-Constructor algorithms. The privacy guarantee of both PVSV-Constructor and PVST-Constructor is 1/2, i.e., the polymorphic value sets constructed by PVSV-Constructor contains 2 values, and the polymorphic value sets constructed by PVST-Constructor contains at least 2 true values. The key parameters here are the size of query workload Q, the size of database D, the number of public and private attributes, and the weight ratios in the ranking function. We randomly generated 20 tuples as the query workload. By default, we picked 10 tuples from Q and set $|D| = 300,000$, $m = 10$, and $m' = 10$. Therefore, we randomly picked 10 attributes from the testing bed and set them as public attributes. The rest of the 10 attributes were set as private attributes.

Many recommendation systems of ONS applications feature top-k recommendation [1, 14, 47] where the ranked result contains a set of k tuples that will be of interest to a certain user, as it is impractical and unnecessary to return all tuples in the database to the user. Therefore, we introduce average top-k utility loss U_{aul}, a variant of U_Q that focuses on utility loss of the top-k tuples in a ranked result. U_{aul} is defined as

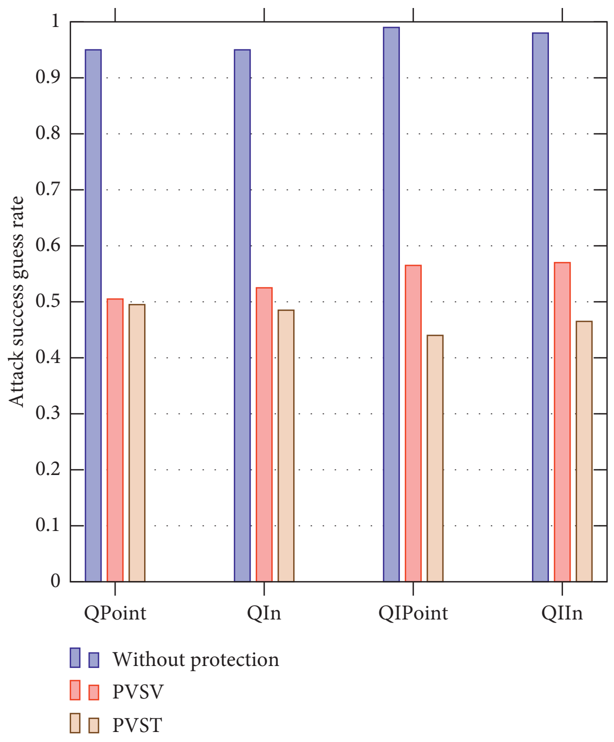

FIGURE 1: Attack success guess.

$$U_{aul} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{|D'|}\sum_{v\in D'}\frac{\left|\min\{\operatorname{Rank}(v\mid q),k+1\}-\min\{\operatorname{Rank}'(v\mid q),k+1\}\right|}{k}, \tag{23}$$

where $D' = \{v \in D \mid \operatorname{Rank}(v \mid q)) < k \text{ or } \operatorname{Rank}(v \mid q)) < k\}$. Intuitively, U_{aul} represents the average percentage rank difference relative to k over all queries and all tuples in top-k. U_{aul} is equivalent to $U_Q/|Q||D|^2$ when $k = |D|$. By default, we set $k = 100$.

6.3.1. Evaluation of U_{aul} with Varying k. We first discuss the average top-k utility loss of PVSV-Constructor, PVST-Constructor, and a baseline algorithm on varying k with other parameters set to default values. For a tuple v's attribute B_j, the baseline algorithm constructs $P_v^{B_j}$ with $v[B_j]$ and a value randomly picked from $V_j^B\backslash\{v[B_j]\}$. The results are presented in Figure 2, which shows that the U_{aul} of PVSV-Constructor is significantly lower than that of the baseline algorithm. The U_{aul} of PVST-Constructor is also lower than that of the baseline algorithm when $k \geq 5$, even though the baseline algorithm cannot preserve privacy against authenticity-knowledgeable adversaries. The experimental results show that both PVSV and PVST can reduce utility loss with respect to rank differences. Also, note that PVST-Constructor constructs polymorphic value sets with true values, and thus, from Figure 3, we can see that some polymorphic value sets constructed by PVST-Constructor contains more than 2 values.

6.3.2. Evaluation of U_{aul} with Varying Sizes of Q. Figure 4 presents the average top-k utility loss of PVSV-Constructor on varying |Q|. When |Q| is set to 1, 5, or 10, we randomly picked 1, 5, or 10 queries from the original Q, respectively. With increasing number of queries in the query workload, the U_{aul} of PVSV-Constructor increases monotonically. The reason is that PVSV-Constructor always generates the least frequent value (denoted as $v'[B_j]$) in $\{q[B_j] \mid q \in Q\}$ for $v[B_j]$. If |Q| is small, then it is possible that $v'[B_j] \notin \{q[B_j] \mid q \in Q\}$, and thus, $s(v \mid q) = s'(v \mid q)$ $\forall q \in Q$. However, a larger query workload covers more private attributes values, and thus, it will be harder for PVSV-Constructor to generate a value for $v[B_j]$ that has no impact on the rank of v for any $q \in Q$. Figure 5 presents the average top-k utility loss of PVST-Constructor on varying |Q|. We can see that the size of Q has no significant impact on the U_{aul} of PVST-Constructor, as the PVST-Constructor always pick a tuple v' that is different from v in most attributes and then insert $v'[B_j]$ into $P_v^{B_j}$.

6.3.3. Evaluation of U_{aul} with Varying Sizes of the Dataset. Figures 6 and 7 depict the impact of the size of datasets on U_{aul} of PVSV-Constructor and PVST-Constructor. Datasets of 100,000 and 200,000 tuples were randomly sampled from

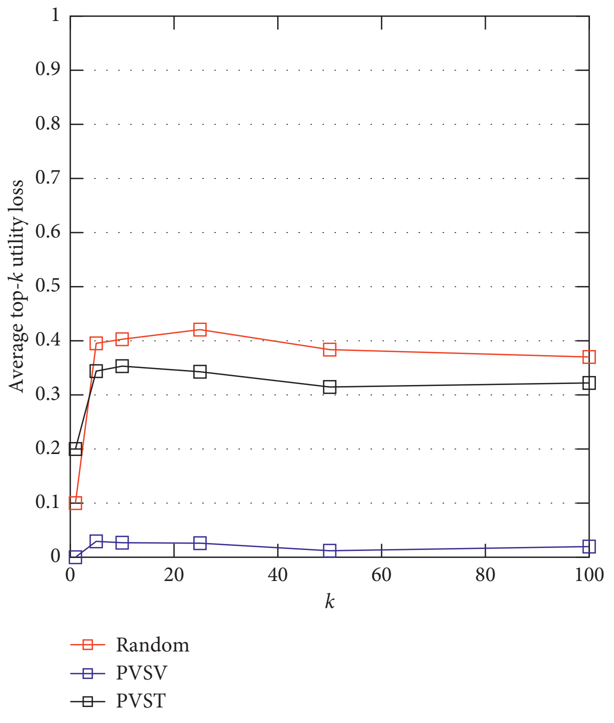

Figure 2: Utility loss vs. k.

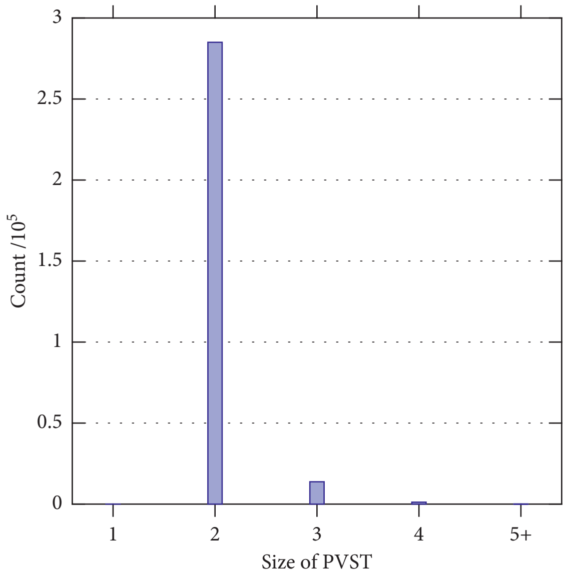

Figure 3: Size of polymorphic value set (PVST).

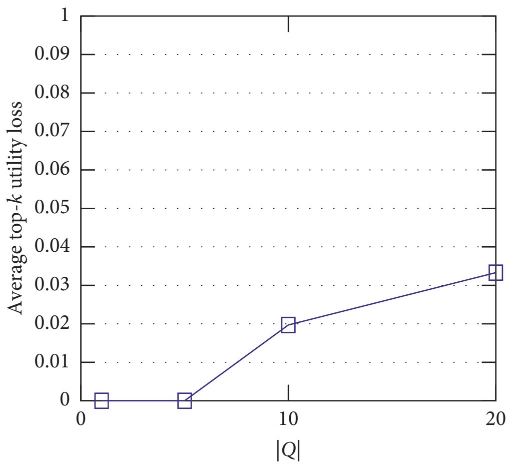

Figure 4: Utility loss vs. $|Q|$.

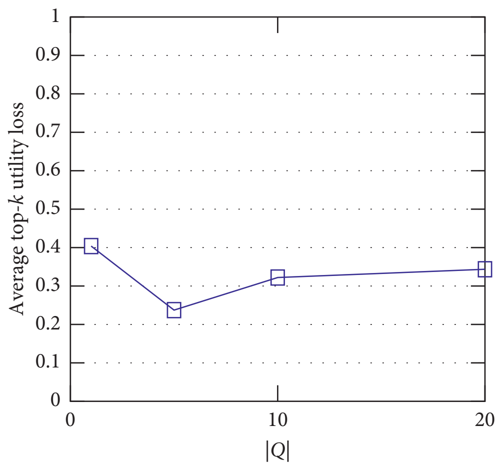

Figure 5: Utility loss vs. $|Q|$.

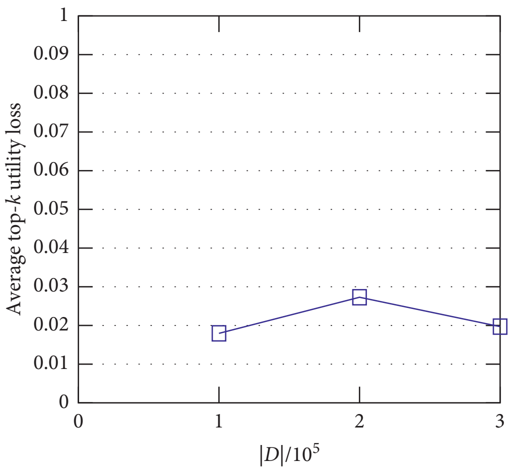

Figure 6: Utility loss vs. $|D|$.

the testing bed of 300,000 tuples. As expected, $|D|$ has no significant impact on the U_{aul} of PVSV-Constructor since PVSV-Constructor generates values from the domain of each private attributes. $|D|$ has no impact on the U_{aul} of PVST-Constructor, which indicates that a dataset containing 100,000 tuples is sufficient for PVST-Constructor to generate polymorphic value sets with true values.

6.3.4. Evaluation of U_{aul} with Varying m. We investigate the impact of the number of private and public attributes on average top-k utility loss. Figure 8 presents the U_{aul} of PVSV-Constructor with fixed m' and varying m and with fixed m and varying m'. When m/m' is set to 5, we randomly removed 5 attributes from the testing bed. When m/m' is set to

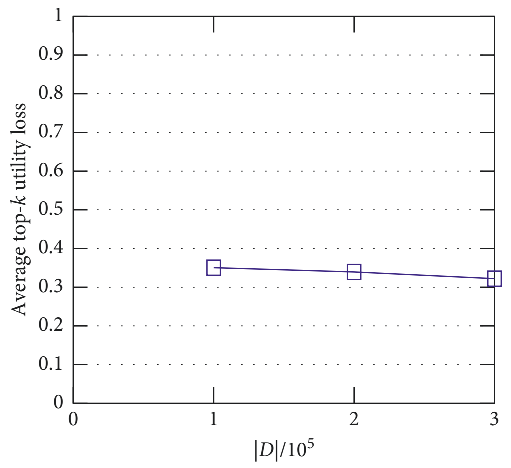

Figure 7: Utility loss vs. $|D|$

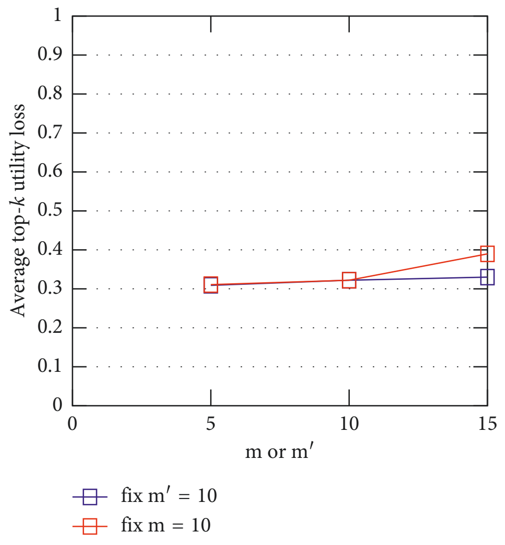

Figure 9: Utility loss vs. m and m'.

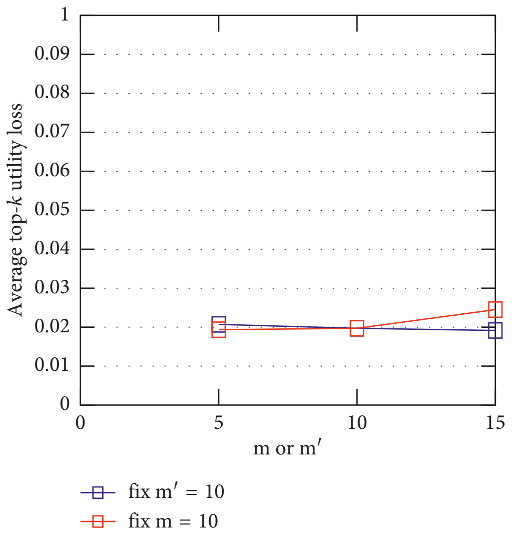

Figure 8: Utility loss vs. m and m'.

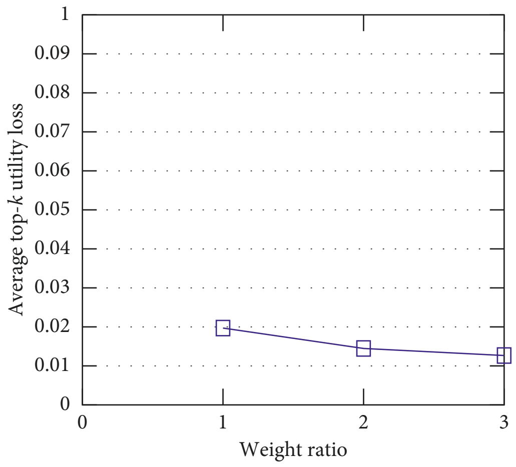

Figure 10: Utility loss vs. weight ratio.

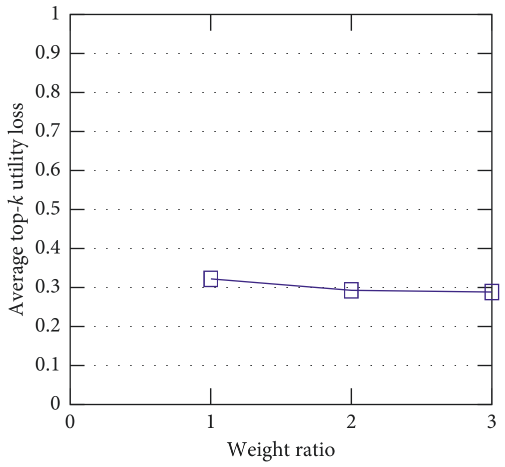

Figure 11: Utility loss vs. weight ratio.

15, we added 5 more categorical attributes randomly chosen from the unused attributes. We observe that the U_{aul} monotonically decreases with increasing number of public attributes and monotonically increases with increasing number of private attributes. As expected, a higher proportion of public attributes leads to less variant between $s(v \mid q)$ and $s'(v \mid q)$. The results of the same experiment with PVST-Constructor are shown in Figure 9. U_{aul} increases as increasing number of private attributes as expected. However, U_{aul} also increases slightly with increasing number of public attributes. This is due to the fact that with more public attributes, there will be few tuples that share the same public attribute values. Since PVST-Constructor inserts only private attribute values from tuples sharing same public attribute values, more values will be inserted to each polymorphic value set, which introduces higher utility loss.

6.3.5. Evaluation of U_{aul} with Varying Weight Ratios. Figures 10 and 11 illustrate the impact of weight ratios on the average utility loss. The experiment was conducted with a fixed private attribute weight of 1 and varying public attribute weights of 1, 2, and 3. As expected, U_{aul} of both PVSV-Constructor and PVST-Constructor decreases as increasing weight ratio of public attributes. The reason is that as the public attribute weight increases, the part in $s'(v \mid q)$ caused by private attributes decreases. Therefore, less impact would be made to $s'(v \mid q)$ by PVSV and PVST. As a result, $s'(v \mid q)$ would be closer to $s(v \mid q)$ and the utility loss could be decreased.

7. Conclusions

In this paper, we proposed a novel framework that preserves privacy of private attributes against arbitrary attacks through the ranked retrieval model. Furthermore, we identify two categories of adversaries based on varying adversarial capabilities. For each kind of adversaries, we presented implementation of our framework. Our experimental results suggest that our implementations efficiently preserve privacy against Rank Inference attack [35]. Moreover, the implementations significantly reduce utility loss with respect to the variance in ranked results.

It is our hope that this paper can motivate further research in privacy preservation of SIoT with consideration of social network features and/or variant information retrieval models, e.g., text mining.

Appendix

A. 2-PVSV

In this subsection, we prove that constructing an optimal PVSV for each attribute of a tuple v is an NP-hard problem.

Definition A.1. For a tuple v in D, we create $P_v^{B_j}$ for each B_j. We say v satisfies query q if the following hold for any tuple t $(t \neq v)$ in D: (1) If $s(v \mid q) < s(t \mid q)$, then $s'(v \mid q) < s'(t \mid q)$, and (2) if $s(v \mid q) \geq s(t \mid q)$, then $s'(v \mid q) \geq s'(t \mid q)$

For a database containing 2 tuples, the 2-PVSV problem can be redefined as follows: given a query workload Q, a database D, construct $P_v^{B_1}, \ldots, P_v^{B'_m}$ of size 2 such that v satisfies the most queries in Q.

Definition A.2. Max-3Sat Problem: given a 3-CNF formula Φ, find the truth assignment that satisfies that most clauses.

Lemma A.1. *Max-3Sat $\leq_P$ 2-PVSV Problem.*

Proof. We construct a reduction function $f(\Phi) = (D, s, Q)$ which takes a Max-3Sat instance as input and returns a 2-PVSV instance. Without loss of generality, we suppose that Φ is a conjunction of l clauses and each clause is a disjunction of 3 literals from set $X = \{x_1, \ldots, x_n\}$. We construct database D as follows: D has 0 public attributes and $n+1$ private attributes $B_1, \ldots, B_n, C_1$. Let V_j^B be the attribute domain of B_j, $j \in \{1, \ldots, n\}$ and V_1^C be the attribute domain of C_1. Let $V_i^B = \{-1, 0, 1, 2, \ldots, n+2\}$ and $V_1^C = \{0, 1\}$. Two tuples, v_1 and v_2, are inserted into D: $v_1[B_j] = 0$ and $v_2[B_j] = j+2$ for $j \in \{1, \ldots, n\}$, while $v_1[C_1] = 0$ and $v_2[C_1] = 1$ We simplify the score function defined in (1) by setting all weights to 1. Also, note that $\rho(q[B_j], t[B_j]) = 0$, if $q[B_j]$ is null.

We construct query workload Q based on Φ. For each clause $L_k \in \Phi$, we construct a query q_k such that $q_k[B_i] = i$ iff the corresponding literal x_i is a positive literal in L_k, and $q[B_i] = i+1$ iff x_i is a negative literal in the clause. For example, given a clause $(x_1 \vee x_2 \vee x_3)$, the corresponding query q should satisfy $q[B_1] = 1, q[B_2] = 3, q[B_3] = 3$, and $q[C_1] = 0$. All other attributes in q are set to *null* by default. Therefore, $s(v_1 \mid q) = 1$ and $s(v_2 \mid q) = 0$, $\forall q \in Q$.

Since $s(v_2 \mid q) < s(v_1 \mid q)$ $\forall q \in Q$, in order to minimize utility loss, the value of $s'(v_2 \mid q)$ should be as small as possible. We observe that the minimum possible $s'(v_2 \mid q)$ is 1 because $P_{v_2}^{C_1}$ must be $\{0, 1\}$ as $V_1^C = \{0, 1\}$. Without loss of generality, we assume that we have already constructed $P_{v_2}^{B_1}, \ldots, P_{v_2}^{B_{m'}}, P_{v_2}^{C_1}$ for v_2 such that $s'(v_2 \mid q) = 1$, $\forall q \in Q$.

Now, we have an instance of 2-PVSV problem that given D, Q, constructs $P_{v_1}^{B_1}, \ldots, P_{v_1}^{B'_m}, P_{v_1}^{C_1}$ that satisfies the most queries in Q. Since $V_1^C = \{0, 1\}$, $P_{v_1}^{C_1}$ must be $\{0, 1\}$ too as $P_{v_1}^{C_1}$ contains two distinct values. Therefore, for an arbitrary query $q \in Q$, we have $\rho'(q[C], v_1[C]) = 1$ and $\rho'(q[C], v_2[C]) = 1$. In order to let v_1 satisfy q, we have to ensure that $s'(v_1 \mid q) > s'(v_2 \mid q) = 1$. Thus, we have

$$\begin{aligned} & s'(v_1 \mid q) > 1, \\ & \Longleftrightarrow \sum_{i=1}^{m'} \rho'(q[B_i], v_1[B_i]) > 1, \\ & \Longleftrightarrow \exists i \in \{1, \ldots, m'\}, \rho'(q[B_i], v_1[B_i]) = 1, \\ & \Longleftrightarrow \exists i \in \{1, \ldots, m'\}, q[B_i] \in P_{v_1}^{B_i}. \end{aligned} \tag{A.1}$$

Now, we show that the solution of 2-PVSV problem constructed above can answer the corresponding Max-3Sat Problem. Suppose that we have the solution $P_{v_1}^{B_1}, \ldots, P_{v_1}^{B_{m'}}$ such that v_1 satisfies the most queries in Q. As in (A.1), if v_1 satisfies q_k, we have $q_k[B_i] \in P_{v_1}^{B_i}$. If v_1 does not satisfy q_k, then $q_k[B_i] \notin P_{v_1}^{B_i}$. Recall that we assign value i or $i+1$ to $q_k[B_i]$ if x_i is a positive or negative literal in clause L_k, respectively. Therefore, the assignment of x_i given $P_{v_1}^{B_i}$ is

$$\begin{aligned} x_i &= \text{true}, \quad \text{if } i \in P_{v_1}^{B_i}, \\ x_i &= \text{false}, \quad \text{if } i+1 \in P_{v_1}^{B_i}. \end{aligned} \tag{A.2}$$

Furthermore, from equation (A.1), we have

$$s'(v_1 \mid q) > 1 \Longleftrightarrow L = \text{true}, \tag{A.3}$$

where L is q's corresponding clause in Φ. Note that $\{i, i+1\} \not\subset Q$ as $|P_{v_1}^{B_i}| = 2$ and $0 \in P_{v_1}^{B_i}$. Thus, the value of x_i is either true or false.

As we proved above, v_1 satisfies q_i if and only if L_i is true given assignment $\{x_1, \ldots, x_l\}$ constructed according to equation (A.2). Therefore, $\{x_1, \ldots, x_l\}$ satisfies the most clauses in ϕ if and only if v_1 satisfies the most queries in Q.

We now prove that function f can be conducted in polynomial time. Given a formula Φ with n variables and l clauses, we construct a 2-PVSV instance with 2 tuples each of which has $n+1$ attributes and l queries each of which has 4 attributes. Therefore, $O(2n+4l)$ assignments are needed and f can be conducted in polynomial time. □

Proof of Theorem 1. We now prove that the 2-PVSV problem is NP-hard. In Lemma 3 we proved that Max-3Sat Problem can be reduced to the 2-PVSV problem in polynomial time. Furthermore, as Max-3Sat Problem is a NP-hard problem [33], the 2-PVSV problem is NP-hard. □

B. 2-PVST

In this section, we prove that constructing an optimal PVST for each private attribute of a tuple $v \in D$ is NP-hard. We use the definition of v satisfying q from (9). The 2-PVST problem can be redefined as follows.

Definition B.1. 2-PVST problem: given a query workload Q, a database D, the optimization problem of 2-PVST is to construct an arbitrary tuple v's polymorphic sets, $P_v^{B_1}, \ldots, P_v^{B'_m}$, that satisfies the most queries in Q. The size of each polymorphic vale set is 2.

Lemma B.1. *Max-3Sat* $\leq_P$ *2-PVST problem.*

Proof. We construct a reduction function $f(\Phi) = (D, s, Q)$ which takes a Max-3Sat instance as input and returns a 2-PVST instance. We assume that Φ is a conjunction of l clauses and each clause is a disjunction of 3 literals from set $X = \{x_1, \ldots, x_n\}$. Let D have no public attribute and $n+1$ private attributes $B_1, \ldots, B_n, C_1$. For each literal x_i, we insert two tuples, v_i and v'_i, into D where

$$\begin{aligned} &v_i[C_1] = 1, v_i[B_i] = 1, v_i[B_j] = -1\ \forall j \neq i,\\ &v'_i[C_1] = 1, v'_i[B_i] = -1, v'_i[B_j] = 1\ \forall j \neq i. \end{aligned} \tag{B.1}$$

Then, we insert tuple v into D where $v[C_1] = 0$ and $v[B_j] = 0\ \forall j \in \{1, \ldots, m'\}$. We also set the domain of each B_j as $\{-1, 0, 1\}$ and the domain of C_1 as $\{0, 1\}$.

Without loss of generality, the score function defined in (1) is simplified by setting all weights to 1.

Next, we construct query workload Q. For each clause $L_k \in \Phi$, we construct a query q_k in which $q_k[B_j] = -1$ iff x_j is a positive literal in L_k, and $q_k[B_j] = 1$ iff x_j is a negative literal in L_k. We also set $q_k[C_1] = 1$. The rest of the attribute values are set to *null* by default. Therefore, if $L_k = (x_1 \vee \neq x_2 \vee x_3)$, then we have $q_k[C_1] = 1$, $q_k[B_1] = 1$, $q_k[B_2] = 1$, $q_k[B_3] = -1$, and $q_k[B_j] = \text{null}$ for $j \in \{4, \ldots, m'\}$.

Note that $\rho(q[B_j], t[B_j]) = 0$ if $q[B_j]$ is null. Thus, $s(v, q_k) = 0\ \forall q \in Q$ and $s(v_i, q_k)$, $s(v'_i, q_k) \geq 1\ \forall q \in Q$. We observe that for $q \in Q$ and $i \in \{1, \ldots, n\}$, $s(v \mid q) < s(v_i \mid q)$ and $s(v \mid q) < s(v'_i \mid q)$. In order to reduce U_Q, we have to maximize the number of $q \in Q$ such that $s'(v \mid q) < s'(v_i \mid q)$ and $s'(v \mid q) < s'(v'_i \mid q)$. The maximum value of $s'(v_i \mid q)$ and $s'(v'_i \mid q)$ is 4, which can be achieved by inserting $v_i[B_j]$ into $P_{v'_i}^{B_j}$ and inserting $v'_i[B_j]$ into $P_{v_i}^{B_j}$. Since $P_v^{C_1} = \{0, 1\}$, the value of $s'(v \mid q)$ is at least 1. Therefore, we have

$$\begin{aligned} &s'(v \mid q) < s'(v_i \mid q),\\ &\Longleftrightarrow s'(v \mid q) < 4,\\ &\Longleftrightarrow \sum_{j=1}^{m'} \rho'(q[B_j], v[B_j]) < 3,\\ &\Longleftrightarrow \exists j \in \{1, \ldots, m'\},\ \rho'(q[B_j], v[B_j]) = 0,\\ &\Longleftrightarrow \exists j \in \{1, \ldots, m'\},\ v[B_j] \neq q[B_j],\\ &\Longleftrightarrow \exists j \in \{1, \ldots, m'\},\ -q[B_j] \in P_v^{B_j}. \end{aligned} \tag{B.2}$$

Since $|P_v^{B_j}|$ is limited to 2, $P_v^{B_j} = \{0, 1\}$ or $\{0, -1\}$, i.e., we must insert either v_i's or v'_i's private attribute values into $P_v^{B_1}, \ldots, P_v^{B_m}$. Recall that we assign -1 or 1 to $q_k[B_j]$ if x_j is positive or negative, respectively, in L_k. In order to reduce Max-3Sat to 2-PSVT, we assign literals in the following way:

$$\begin{aligned} &x_j = \text{true}, \quad \text{if } -1 \in P_v^{B_j},\\ &x_j = \text{false}, \quad \text{if } 1 \in P_v^{B_j}. \end{aligned} \tag{B.3}$$

Therefore, we denote the corresponding clause of q as L, and from equation (B.2), we have

$$\begin{aligned} &\exists j \in \{1, \ldots, m'\},\ -q[B_j] \in P_v^{B_j},\\ &\Longleftrightarrow \exists j \in \{1, \ldots, m'\}, x_j = \text{true} \quad \text{if } x_j \text{ is positive in } L,\\ &\text{or } x_j = \text{false if } x_j \text{ is negative in } L,\\ &\Longleftrightarrow L = \text{true}. \end{aligned} \tag{B.4}$$

From the above equation, we observe that v satisfying q_k is equivalent to L_k being true. Therefore, we can reduce Max-3Sat to 2-PVST. Given a solution to the 2-PVST problem that maximizes the number of queries satisfied by v, assignment $\{x_1, \ldots, x_n\}$ produced by equation (B.3) can satisfy the largest number of clauses in Φ.

Given a Max-3Sat instance Φ, the construction of the 2-PVST instance can be conducted in polynomial time as we construct n queries with 3 attributes and $2n+1$ tuples with n attributes. The transformation from the optimal solution of 2-PVST to the optimal solution of Max-3Sat also takes polynomial time as we assign the value to each one of $x_1, \ldots, x_n$ once. Therefore, f is a polynomial time function. □

Proof of Theorem 2. In 4, we proved that a Max-3Sat instance can be reduced to a 2-PSVT instance in polynomial time. Furthermore, as Max-3Sat Problem is an NP-hard problem, the 2-PVSV problem is an NP-hard problem. □

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was partially supported by the National Science Foundation (grant nos. 0852674, 0915834, 1117297, and 1343976) and the Army Research Office (grant no. W911NF-15-1-0020).

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] C. C. Aggarwal and S. Y. Philip, "A condensation approach to privacy preserving data mining," in *Proceedings of the International Conference on Extending Database Technology*, pp. 183–199, Springer, Heraklion, Crete, Greece, March 2004.

[3] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM Sigmod Record*, vol. 29, no. 2, pp. 439–450, ACM, 2000.

[4] A. Alcaide, E. Palomar, J. Montero-Castillo, and A. Ribagorda, "Anonymous authentication for privacy-preserving IoT target-driven applications," *Computers & Security*, vol. 37, pp. 111–123, 2013.

[5] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT)—when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[6] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[7] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 24, p. 1, 2018.

[8] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 17, 2019.

[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[10] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 442–455, Springer, New York, NY, USA, June 2005.

[11] C. Chen, X. Zhu, P. Shen et al., "An efficient privacy-preserving ranked keyword search method," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 951–963, 2016.

[12] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, p. 4, IEEE, Houston, TX, USA, November 2005.

[13] V. Ciriani, S. D. C. Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragmentation and encryption to enforce privacy in data storage," in *Proceedings of the Fifth European symposium on research in computer security*, pp. 171–186, Springer, Dresden, Germany, September 2007.

[14] M. Deshpande and G. Karypis, "Item-based top-*N* recommendation algorithms," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 143–177, 2004.

[15] S. D. C. di Vimercati, R. F. Erbacher, S. Foresti, S. Jajodia, G. Livraga, and P. Samarati, "Encryption and fragmentation for data confidentiality in the cloud," in *Foundations of Security Analysis and Design VII*, A. Aldini, J. Lopez, and F. Martinelli, Eds., pp. 212–243, Springer, Berlin, Germany, 2013.

[16] S. D. C. di Vimercati, S. Foresti, G. Livraga, S. Paraboschi, and P. Samarati, "Confidentiality protection in large databases," in *A Comprehensive Guide through the Italian Database Research over the Last 25 Years*, pp. 457–472, Springer, Berlin, Germany, 2018.

[17] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.

[18] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[19] S. E. Fienberg and J. McIntyre, "Data swapping: variations on a theme by dalenius and reiss," in *Privacy in Statistical Databases*, pp. 14–29, Springer, Berlin, Germany, 2004.

[20] Y. Gao, T. Yan, and N. Zhang, "A privacy-preserving framework for rank inference," in *Proceedings of the IEEE Symposium on Privacy-Aware Computing (PAC)*, pp. 180-181, IEEE, Washington DC, USA, August 2017.

[21] Z. He, Z. Cai, and J. Yu, "Latent-data privacy preserving with customized data utility for social network data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 665–673, 2017.

[22] T. Hong, S. Mei, Z. Wang, and J. Ren, "A novel vertical fragmentation method for privacy protection based on entropy minimization in a relational database," *Symmetry*, vol. 10, no. 11, p. 637, 2018.

[23] X. Huang, R. Fu, B. Chen, T. Zhang, and A. Roscoe, "User interactive internet of things privacy preserved access control," in *Proceedings of the International Conference for Internet Technology and Secured Transactions*, pp. 597–602, IEEE, London, UK, December 2012.

[24] Y. Huo, X. Fan, L. Ma, X. Cheng, Z. Tian, and D. Chen, "Secure communications in tiered 5G wireless networks with cooperative jamming," *IEEE Transactions on Wireless Communications*, vol. 18, no. 6, pp. 3265–3280, 2019.

[25] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data miningin latex," *IEEE Transactions on knowledge and Data Engineering*, vol. 18, no. 1, pp. 92–106, 2005.

[26] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: privacy beyond k-anonymity and l-diversity," in *Proceedings of the IEEE 23rd International Conference on Data Engineering*, pp. 106–115, IEEE, Istanbul, Turkey, April 2007.

[27] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, pp. 3–es, 2007.

[28] S. Matwin, "Privacy-preserving data mining techniques: survey and challenges," in *Studies in Applied Philosophy, Epistemology and Rational Ethics*, pp. 209–221, Springer, Berlin, Germany, 2013.

[29] B. McFee and G. Lanckriet, "Metric learning to rank," in *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, Haifa, Israel, June 2010.

[30] K. Muralidhar and R. Sarathy, "Data shuffling—a new masking approach for numerical data," *Management Science*, vol. 52, no. 5, pp. 658–670, 2006.

[31] J. Nin, J. Herranz, and V. Torra, "Rethinking rank swapping to decrease disclosure risk," *Data & Knowledge Engineering*, vol. 64, no. 1, pp. 346–364, 2008.

[32] K. Okamoto, W. Chen, and X.-Y. Li, "Ranking of closeness centrality for large-scale social networks," in *Frontiers in Algorithmics*, pp. 186–195, Springer, Berlin, Germany, 2008.

[33] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *Journal of computer and system sciences*, vol. 43, no. 3, pp. 425–440, 1991.

[34] L. Peng, R.-C. Wang, X.-Y. Su, and C. Long, "Privacy protection based on key-changed mutual authentication protocol in internet of things," in *Communications in Computer and Information Science*, pp. 345–355, Springer, Berlin, Germany, 2013.

[35] M. F. Rahman, W. Liu, S. Thirumuruganathan, N. Zhang, and G. Das, "Privacy implications of database ranking," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1106–1117, 2015.

[36] R. Schenkel, T. Crecelius, M. Kacimi et al., "Efficient top-*k* querying over social-tagging networks," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 523–530, ACM, Singapore, July 2008.

[37] C. Sheng, N. Zhang, Y. Tao, and X. Jin, "Optimal algorithms for crawling a hidden database in the web," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1112–1123, 2012.

[38] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, IEEE, Berkeley, CA, USA, May 2000.

[39] J. Su, D. Cao, B. Zhao, X. Wang, and I. You, "ePASS: an expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the Internet of Things," *Future Generation Computer Systems*, vol. 33, pp. 11–18, 2014.

[40] L. Sweeney, "*k*-anonymity: a Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[41] T. Tassa, A. Mazza, and A. Gionis, "*k*-concealment: An alternative model of *k*-type anonymity," *Transactions on Data Privacy*, vol. 5, no. 1, pp. 189–222, 2012.

[42] J. H. Y. F. W. Wang, J. C. W. G. K. Koperski, D. Li, Y. L. A. R. N. Stefanovic, and B. X. O. R. Z.. Dbminer, "A system for mining knowledge in large relational databases," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining and Knowledge Discovery (KDD'96)*, pp. 250–255, San Diego, CA, USA, August 1996.

[43] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: toward data privacy in the IoT," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 725–730, IEEE, Sydney, Australia, June 2014.

[44] Y. Wang and Q. Wen, "A privacy enhanced dns scheme for the internet of things," in *Proceedings of the IET International Conference on Communication Technology and Application (ICCTA 2011)*, The Institution of Engineering & Technology, Beijing, China, October 2011.

[45] Y. Xu, T. Ma, M. Tang, and W. Tian, "A survey of privacy preserving data publishing using generalization and suppression," *Applied Mathematics & Information Sciences*, vol. 8, no. 3, pp. 1103–1116, 2014.

[46] J.-C. Yang and B.-X. Fang, "Security model and key technologies for the internet of things," *The Journal of China Universities of Posts and Telecommunications*, vol. 18, pp. 109–112, 2011.

[47] X. Yang, H. Steck, Y. Guo, and Y. Liu, "On top-*k* recommendation using social networks," in *Proceedings of the sixth ACM conference on Recommender systems—RecSys'12*, pp. 67–74, ACM, Dublin, Ireland, September 2012.

[48] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow but no track: privacy preserved profile publishing in cyber-physical social systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1868–1878, 2017.