Received January 22, 2018, accepted March 2, 2018, date of publication March 12, 2018, date of current version May 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2814622

Two Privacy-Preserving Approaches for Publishing Transactional Data Streams

JINYAN WANG, CHAOJI DENG, AND XIANXIAN LI

[1]Guangxi Key Lab of Multi-Source Information Mining and Security, Guangxi Normal University, Guilin 541004, China
[2]School of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China

Corresponding author: Xianxian Li (lixx@gxnu.edu.cn)

This paper was supported in part by the National Natural Science Foundation of China (61502111, 61763003, 61672176, 61562007 and 61662008), in part by the Guangxi Natural Science Foundation (2016GXNSFAA380192 and 2015GXNSFBA139246), in part by the Guangxi "Bagui Scholar" Teams for Innovation and Research Project, Guangxi Special Project of Science and Technology Base and Talents (AD16380008), and in part by the Guangxi Collaborative Innovation Center of Multisource Information Integration and Intelligent Processing.

ABSTRACT Recently, data mining over transactional data streams has become an attractive research area. However, releasing raw transactional data streams, in which only explicit identifying information must be removed, may compromise individual privacy. Many privacy-preserving approaches have been proposed for publishing static transactional data. Due to the characteristics of data streams, which must be processed quickly, static data anonymization methods cannot be directly applied to data streams. In this paper, we first analyze the privacy problem in publishing transactional data streams based on a sliding window. Then, we present two dynamic algorithms with generalization and suppression to anonymize continuously a sliding window to make it satisfy ρ-uncertainty by structuring an affected sensitive rules trie, because the removal and addition of transactions may make the current sliding window fail to satisfy ρ-uncertainty. Experimental results show that our methods are more efficient than sliding window anonymization with batch processing by using existing static anonymization methods.

INDEX TERMS Data publishing, privacy preservation, sliding window, transactional data stream.

I. INTRODUCTION

Privacy-preserving data publishing provides methods and tools for publishing useful information while preserving individual privacy. Recently, it has received wide attention from academia and industry, and many approaches have been proposed for different data publishing scenarios [1]–[7]. Transactional data (or set-valued data) refers to data in which each record (or transaction) consists of a set of items drawn from a universe of items [8], [9]. With the advent of big data, the data may arrive continuously at high speed, leading to a potentially unbounded and ever-growing dataset called a data stream [10]. Publishing transactional data streams provides enormous opportunities for various data mining tasks, such as frequent pattern mining for a wide range of online applications including retail chain data analysis, network traffic analysis, and web-server log and click-stream analysis [11]. However, disclosing raw transactional data streams may compromise individual privacy. Recent work has studied the privacy problem for publishing static transactional data. Nevertheless, due to the characteristics of data streams, these techniques cannot be directly applied to transactional data streams. We investigate the privacy protection problem for publishing transactional data streams, which is more challenging than that for publishing traditional static datasets.

A data processing model is required to extract transactions from continuously generated data streams. Data stream mining techniques are classified into three categories according to the stream processing models: the landmark window model, the damped window model and the sliding window model [12]. In the landmark window model, these transactions from a specific point of time, called the landmark, to the current time are considered. In the damped window model, each transaction has a weight that decreases with time. In the sliding window model, a fixed length of recently arrived data is considered, i.e., given a window on a transactional data stream, only the latest w transactions or all transactions in the last w time units are utilized for data mining; these windows are called the transaction-sensitive window and time-sensitive window, respectively. As new transactions arrive, the oldest transactions in the sliding window are removed and

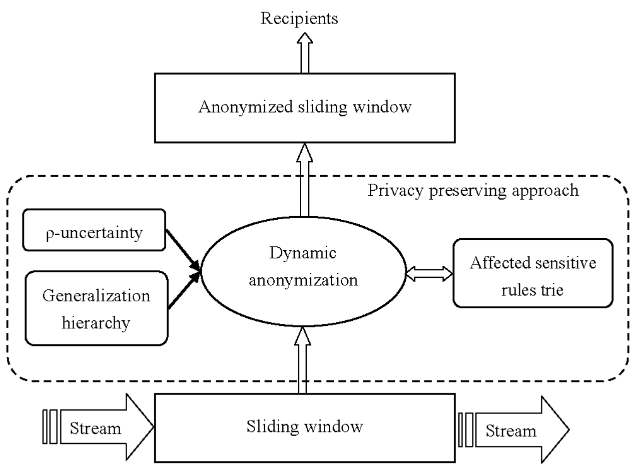


FIGURE 1. **Mining transactional stream based on a sliding window.**

the new transaction is inserted. The sliding window model is widely used for stream mining, because processing recent data is usually important for applications that handle stream-oriented data [11], [13]–[17].

Due to the wide application of sliding windows for data analysis, we consider the privacy problem in processing data stream based on a sliding window. We assume that the recipients of a published data stream are untrustworthy and include the researcher for data mining and other attackers. If an attacker has partial knowledge about a subset of items corresponding to a victim, he/she may attempt to infer sensitive information about the victim. For example, consider a supermarket that released its transactional data stream to a data mining company for marketing strategy analysis. *Mike* from the data mining company is analyzing the data, and he happened to know that his friend *Lily* bought *wine*, *milk* and *scissors* from this supermarket yesterday, so *Mike* knows that the transaction appears in a particular sliding window. If there is only one transaction containing *wine*, *milk* and *scissors* in the sliding window, *Mike* can immediately infer that this transaction corresponds to *Lily*, and he can determine her complete shopping bag contents. If the transaction contains some sensitive items, such as a *pregnancyTest*, *Lily*'s privacy is revealed. In this paper, two privacy-preserving approaches for transactional data streams based on a sliding window are proposed. Fig. 1 shows an overview of the transactional data stream mining environment. We dynamically anonymize the sliding window to make it satisfy ρ-uncertainty, a privacy model proposed in [18]. These anonymization methods for static transactional data are time-consuming, and are not suitable for processing the sliding window over a data stream. Our major contributions are summarized as follows:

1) For a transactional data stream, when the sliding window, which satisfies ρ-uncertainty, is updated, the removal and addition of transactions may make the sliding window fail to satisfy ρ-uncertainty. We analyze the effect on privacy caused by deleting a transaction with/without sensitive items or by adding a transaction with/without sensitive items.
2) By structuring an affected sensitive rules trie *ASRT*, we first present a dynamic algorithm with only suppression to continuously awnonymize a sliding window to make it satisfy ρ-uncertainty. Then, to decrease the information loss, we present a dynamic anonymization algorithm by combining generalization and suppression.
3) Experimental results show that our methods are more efficient than sliding window anonymization with batch processing by using the static anonymization methods proposed in [18].

The paper is organized as follows. We review the related literature in next section. In Section 3, we give some related notions, and analyze the effect on privacy by removal and addition of transactions in a sliding window. The structure of an affected sensitive rules trie and two dynamic anonymization algorithms are presented in Section 4, and experimental results are described in Section 5. In the last section, we conclude this paper.

II. RELATED WORK

In this section, we review the related work about privacy-preserving data publishing for transactional data and data streams, which are directly relevant to our approaches.

A. PRESERVING PRIVACY FOR PUBLISHING TRANSACTIONAL DATA

Transactional data is high-dimensional, and the traditional privacy protection methods for relational data are not adapted directly for transactional data. The existing research on anonymizing transactional data for the purpose of data mining can be divided into two categories according to whether the items are distinguished as sensitive and non-sensitive [8].

In the first case, the distinction between sensitive and non-sensitive is eliminated, and all items are considered to be equally sensitive. Terrovitis *et al.* [19] proposed the k^m-anonymization model, which assumes the background knowledge of an adversary is at most m items and is a relaxation of k-anonymity for relational data [20]. They achieved k^m-anonymity by global generalization. Liu and Wang [9] provided an anonymization method by integrating suppression and generalization. However, it is difficult to bound the background knowledge of an adversary. He and Naughton [21] did not restrict the background knowledge, and applied k-anonymity to transactional data, which means that for any transaction, there are at least $k-1$ other identical transactions. Additionally, they proposed a top-down local generalization to achieve k-anonymity.

In fact, it is reasonable to divide all items in the universe into sensitive and non-sensitive. Ghinita *et al.* [22] assumed that an adversary has background knowledge of an arbitrary number of non-sensitive items, and proposed a permutation method that first groups transactions with close proximity and then associates each group with a set of diversified sensitive values. Xue *et al.* [23] anonymized data in a generalized form by nonreciprocal recoding instead of using a permutation and generalization hierarchy tree. Xu *et al.* [24] introduced (h, k, p)-coherence by confining an attacker with maximum knowledge of p items to identify each transaction from k others in which no more than $h\%$ share a common private item. Additionally, they utilized global suppression with the goal of preserving as many item instances as possible. The methods above all assumed that an adversary's background knowledge is restricted to non-sensitive items. However, an attacker may also obtain partial knowledge about sensitive items. Therefore, Cao *et al.* [18] proposed the ρ-uncertainty privacy model, which does not allow an attacker knowing any subset of a transaction t to infer a sensitive item $\alpha \in t$ with confidence greater than ρ, and they anonymized transaction data by suppression and generalization.

B. PRESERVING PRIVACY FOR PUBLISHING DATA STREAMS

Data streams are continuous, transient, and usually unbounded [25], and some research on privacy-preserving data publishing in relational data streams has been published. Cao *et al.* [25], [26] presented a cluster framework to continuously anonymize data stream with k-anonymity or l-diversity [27] with maximum allowed delay guarantee δ between incoming data and the corresponding anonymized output. Guo and Zhang [28] proposed a novel k-anonymization approach for data streams based on clustering by considering the time constraints on tuple publication and cluster reuse, to accelerate the anonymization process and reduce the information loss. Li *et al.* [29] proposed the *SKY* method based on the specialization tree to continuously facilitate δ-constraint k-anonymity on data stream for privacy protection. These methods all enforce a user-defined time constraint to limit the delay of the published data, and consider anonymizing nearly-expired data. This process damages the utility of the anonymized data. Zhou *et al.* [30] developed a novel approach for k-anonymizing a stream of relational data, which creates and fills clusters with arriving data elements, and publishes them when the clusters satisfy k-anonymity. Also, they considered the statistical distribution of the data stream to limit information loss due to generalization. Kim *et al.* [31] presented delay-free anonymization by using the accumulation-based method to preserve the privacy of electronic health data streams. The method does not generate an accumulation delay, because the input streams are anonymized immediately with counterfeit values. They further devised late validation to increase the data utility of the anonymization results and to manage the counterfeit values.

Data ming usually focuses on finding and extracting useful information from recently arrived data elements, since processing recent data is usually important for applications that handle stream-oriented data. The sliding window is an efficient and commonly used approach to handle or mine continuously generated data streams [11]. The above anonymization methods for data streams do not guarantee that every sliding window satisfies a given privacy requirement, and still lead to privacy leakage. The problem of privacy protection based on a sliding window is challenging because the sliding window is updated frequently and rapidly. Wang *et al.* [32] proposed a sliding window anonymization framework *SWAF* to solve this problem for relational data stream by continuously facilitating k-anonymity on a sliding window. Al-Hussaeni *et al.* [33] devised an incremental trajectory stream anonymizer to incrementally anonymize a sequence of sliding windows on trajectory stream under the *LKC*-privacy requirement [34]. We consider privacy preserving for publishing transactional data streams based on sliding windows.

TABLE 1. Summary of notations.

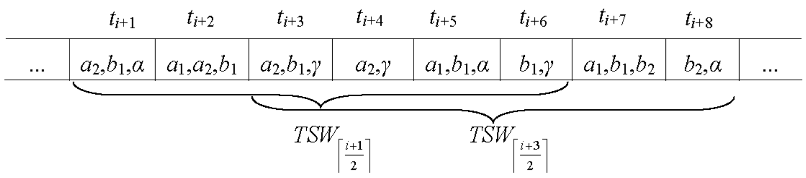
Symbol	Description
TDS	a transactional data stream
I	a finite set of distinct items
I_S	the set of sensitive items
I_N	the set of non-sensitive items
$\mathcal{H}$	generalization hierarchy tree for items in I_N
TSW	transactional sliding window
W_{del}	the set of deleted anonymous transactions
W_{add}	the set of added raw transactions
w	window size of sliding window
p	step size of sliding window
SAR	sensitive association rule
$ASRT$	affected sensitive rules trie

III. PROBLEM DESCRIPTION

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of distinct items, which are units of information. $I = I_S \cup I_N$ and $I_S \cap I_N = \emptyset$, where I_S and I_N are the sets of sensitive and non-sensitive items, respectively. A transaction is a subset of I with a unique transaction identifier *tid*. A transactional data stream $TDS = \{T_1, T_2, \ldots, T_n, \ldots\}$ is an infinite sequence of transactions, where T_i is the ith arrived transaction. A transactional sliding window $TSW_{\lceil \frac{n-w+1}{p} \rceil} = [T_{n-w+1}, T_{n-w+2}, \ldots, T_n]$ over data stream TDS contains w recent transactions of TDS, and TSW slides forward with step size p when p new transactions arrive from TDS by inserting the new transactions into TSW and deleting the oldest p transactions from the window, where w and $\lceil \frac{n-w+1}{p} \rceil$ are the window size and window identifier, respectively. For convenience, Table 1 summarizes the symbols used in this paper.

A transactional data stream based on sliding window is shown in Fig. 2, where the window size and step size are 6 and 2, respectively. The items a_1, a_2, b_1 and b_2 are non-sensitive, and α and γ are sensitive. Transactional sliding window $TSW_{\lceil \frac{i+1}{2} \rceil}$ contains transactions $t_{i+1}, \ldots, t_{i+6}$. When new transactions t_{i+7} and t_{i+8} arrive, the oldest 2 transactions t_{i+1} and t_{i+2} are deleted from the transactional sliding window, and the current window is $TSW_{\lceil \frac{i+3}{2} \rceil}$.

Assume that *Mike* knows that *Lily* bought a_2 and b_1 from a store on one day and that the store publishes its shop-



t_{i+1} t_{i+2} t_{i+3} t_{i+4} t_{i+5} t_{i+6} t_{i+7} t_{i+8}
... a_2,b_1,α a_1,a_2,b_1 a_2,b_1,γ a_2,γ a_1,b_1,α b_1,γ a_1,b_1,b_2 b_2,α ...
$TSW_{\lceil\frac{i+1}{2}\rceil}$
$TSW_{\lceil\frac{i+3}{2}\rceil}$

FIGURE 2. A transactional data stream based on a sliding window.

ping data stream for mining, as shown in Fig. 2. We also assume that *Mike* can monitor the data stream by sliding window, and knows the transaction of *Lily* in $TSW_{\lceil\frac{i+3}{2}\rceil}$. Then, *Mike* can infer that *Lily* also bought γ, so *Lily*'s privacy is compromised.

A. PRIVACY MODEL

ρ-uncertainty is a privacy model for publishing static transactional data [18], and we extend it to data streams. Assume that an attacker can monitor a data stream by sliding window model, and knows a subset of items χ of the victim's transaction t in a transactional sliding window. If an attacker can infer with high probability that t contains χ and it also contains a sensitive item α from the transactional sliding window, where $\alpha \notin \chi$, the privacy is compromised. It means to mine the association rule $\chi \rightarrow \alpha$ from the transactional sliding window, where χ is the antecedent and α is the consequent. Such an association rule is called a sensitive association rule (*SAR*). The confidence of a *SAR* $\chi \rightarrow \alpha$ in a transactional sliding window TSW_i can be computed by $sup(\chi \cup \alpha)/sup(\chi)$, where $sup(\chi)$ is the number of transactions in TSW_i which contains χ. To prevent these inferences, we need to anonymize every transactional sliding window and require that the confidence of each *SAR* appearing in a transactional sliding window is less than a threshold value ρ.

Definition 3.1: For a transactional data stream *TDS*, a transactional sliding window TSW_i over *TDS* is said to satisfy ρ-uncertainty if and only if for $\forall t \in TSW_i$, $\forall \chi \subset t$, and $\forall \alpha \notin \chi$, $\alpha \in I_S$, the confidence of the *SAR* $\chi \rightarrow \alpha$ is less than a threshold value ρ, $\rho \in (0, 1]$. If any transactional sliding window satisfies ρ-uncertainty, then *TDS* satisfies ρ-uncertainty.

B. THE ANALYSIS OF VIOLATING PRIVACY FOR SLIDING WINDOW

When a sliding window satisfies ρ-uncertainty, as p new transactions arrive, the window slides forward, i.e., the oldest p transactions in the sliding window are removed and the new p transactions are inserted into the window. The current sliding window may violate ρ-uncertainty because of the deletion and addition of transactions. In the following, when a sliding window satisfies ρ-uncertainty, we analyze the influence of the deletion or addition of a transaction on the privacy of the sliding window.

Theorem 3.1: Assume that a sliding window satisfies ρ-uncertainty. For a transaction t, it contains the set of items I_t. When t is deleted from the sliding window, we have

1) any *SAR* $\chi \rightarrow \alpha$ in the sliding window, in which $\chi \subseteq I_t$ and $\alpha \notin I_t$ may violate ρ-uncertainty;
2) any *SAR* $\chi \rightarrow \alpha$ in the sliding window, in which $\chi \nsubseteq I_t$ and $\alpha \in I_t$ does not violate ρ-uncertainty;
3) any *SAR* $\chi \rightarrow \alpha$ in the sliding window, in which $\chi \subseteq I_t$ and $\alpha \in I_t$ does not violate ρ-uncertainty.

Proof: For any *SAR* $\chi \rightarrow \alpha$, we know that its confidence can be computed by $sup(\chi \cup \alpha)/sup(\chi)$ and $sup(\chi \cup \alpha)/sup(\chi) < \rho$, i.e., the association rule satisfies ρ-uncertainty. When t is deleted from sliding window, we consider the following cases.

1) For any *SAR* $\chi \rightarrow \alpha$, in which $\chi \subseteq I_t$ and $\alpha \notin I_t$, $sup(\chi \cup \alpha)$ is not changed, and $sup(\chi)$ is decreased. So $sup(\chi \cup \alpha)/sup(\chi)$ is increased. It may lead to $sup(\chi \cup \alpha)/sup(\chi) \geq \rho$. Therefore, $\chi \rightarrow \alpha$ may violate ρ-uncertainty.
2) For any *SAR* $\chi \rightarrow \alpha$ in which $\chi \nsubseteq I_t$ and $\alpha \in I_t$, it is clearly that the deletion of t does not influence the confidence of the association rule.
3) For any *SAR* $\chi \rightarrow \alpha$ in which $\chi \subseteq I_t$ and $\alpha \in I_t$, i.e., $\chi \cup \alpha \subseteq I_t$. We have that

$$\begin{aligned}\frac{sup(\chi \cup \alpha)}{sup(\chi)} &= \frac{sup(\chi \cup \alpha)}{sup(\chi \cup \alpha) + sup(\chi \cup \neg\alpha)} \\ &= \frac{sup(\chi \cup \alpha) + sup(\chi \cup \neg\alpha)}{sup(\chi \cup \alpha) + sup(\chi \cup \neg\alpha)} \\ &\quad - \frac{sup(\chi \cup \neg\alpha)}{sup(\chi \cup \alpha) + sup(\chi \cup \neg\alpha)} \\ &= 1 - \frac{sup(\chi \cup \neg\alpha)}{sup(\chi \cup \alpha) + sup(\chi \cup \neg\alpha)},\end{aligned}$$

where $sup(\chi \cup \neg\alpha)$ represents the number of transactions which contain χ but does not contain α. When t is deleted, $sup(\chi \cup \alpha)$ is decreased, and so $\frac{sup(\chi \cup \alpha)}{sup(\chi)}$ is decreased. The sensitive association rule does not violate ρ-uncertainty after t is deleted. □

Theorem 3.2: Assume that a sliding window satisfies ρ-uncertainty. For a transaction t, it contains the set of items I_t. When t is added to the sliding window, we have

1) any *SAR* $\chi \rightarrow \alpha$ in the sliding window, in which $\chi \subseteq I_t$ and $\alpha \notin I_t$ does not violate ρ-uncertainty;
2) any *SAR* $\chi \rightarrow \alpha$ in the sliding window, in which $\chi \nsubseteq I_t$ and $\alpha \in I_t$ does not violate ρ-uncertainty;
3) any *SAR* $\chi \rightarrow \alpha$ in the sliding window, in which $\chi \subseteq I_t$ and $\alpha \in I_t$ may violate ρ-uncertainty.

Proof: The proof is similar Theorem 3.1. □

When the sliding window is updated, it may violate ρ-uncertainty. According to Theorems 3.1 and 3.2, we need to

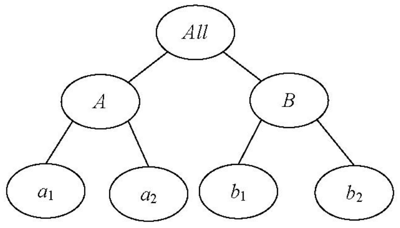


FIGURE 3. A generalization hierarchy tree.

consider only these *SAR*s, which may violate ρ-uncertainty. If we utilize the static anonymization method proposed in [18], all the *SAR*s in the current sliding window are checked for whether they violate ρ-uncertainty.

C. INFORMATION METRIC

For a transactional data stream *TDS*, if a transactional sliding window TSW_i over *TDS* does not satisfy ρ-uncertainty, we can transform it to satisfy ρ-uncertainty by generalization and suppression. For the items in I_N, a generalization hierarchy tree is constructed, in which a leaf node represents a non-sensitive item, an internal node represents a generalized value for some items, and the root is the generalized value for all items. This process is identical to that in [18]. An example is shown in Fig. 3. a_1, a_2, b_1 and b_2 are non-sensitive items. a_1 and a_2 can be generalized to A and b_1 and b_2 can be generalized to B. Furthermore, A and B can be generalized to *All*.

Given a hierarchy tree $\mathcal{H}$ for non-sensitive items, let k be a node in it. The information loss of k is defined as $\mathcal{IL}_k = |leaves(k)|/|I_N|$, where $leaves(k)$ is the set of leaves of the subtree with k as the root node in $\mathcal{H}$. When k is a leaf, $\mathcal{IL}_k = 0$. For example, the information loss of node A in Fig. 3 is $\frac{2}{4} = \frac{1}{2}$. Given an item a, if a is generalized to node $k \in \mathcal{H}$, the information loss for a is defined as $InfoLoss(a) = \mathcal{IL}_k$; if a is suppressed, $InfoLoss(a) = 1$.

Definition 3.2: For a transactional sliding window TSW_i over a transactional data stream *TDS*, the information loss of anonymizing TSW_i is

$$InfoLoss(TSW_i) = \frac{\sum_{a\in I} sup(a) \times InfoLoss(a)}{\sum_{a\in I} sup(a)}, \quad (1)$$

where $sup(a)$ is the number of transactions in TSW_i that contain a.

Definition 3.3: For a transactional data stream *TDS*, the average information loss of the sliding window is

$$AveInfoLoss(TSW) = \frac{\sum_{i=1}^{\omega} InfoLoss(TSW_i)}{\omega}, \quad (2)$$

where ω is the number of different sliding windows.

IV. ANONYMOUS ALGORITHM FOR PUBLISHING TRANSACTIONAL DATA STREAMS

To calculate the confidence of a *SAR* $\chi \to \alpha$, the supports of itemsets χ and $\chi\alpha$ must be computed. For a transactional sliding window TSW_i, we can transform it into a soft set [35], [36]. Let U be an initial universe set, E be a set of parameters and $\eta \subseteq E$, and $\mathscr{P}(U)$ be the power set of U. A pair (F, η) is called a soft set over U, where F is a mapping given by $F : \eta \to \mathscr{P}(U)$. For TSW_i, let U be the set of transactions appearing in TSW_i, E be the set of items I, and η be the subset of items appearing in TSW_i. Then, $F(a)$ is the set of transactions which contain item a in TSW_i, $a \in \eta$. The support of a is $|F(a)|$. For $\forall a, b \in \eta$, the support of ab is $|F(a) \cap F(b)|$.

For the $TSW_{\lceil \frac{i+1}{2} \rceil}$ in Fig. 2, the corresponding soft set (F, η) is

$$\begin{aligned}
F(a_1) &= \{t_{i+2}, t_{i+5}\},\\
F(a_2) &= \{t_{i+1}, t_{i+2}, t_{i+3}, t_{i+4}\},\\
F(b_1) &= \{t_{i+1}, t_{i+2}, t_{i+3}, t_{i+5}, t_{i+6}\},\\
F(b_2) &= \{t_{i+1}, t_{i+2}, t_{i+3}, t_{i+5}, t_{i+6}\},\\
F(\alpha) &= \{t_{i+1}, t_{i+5}\},\\
F(\gamma) &= \{t_{i+3}, t_{i+4}, t_{i+6}\}.
\end{aligned}$$

The support of a_1 is $|F(a_1)| = |\{t_{i+2}, t_{i+5}\}| = 2$, and the support of a_1a_2 is $|F(a_1) \cap F(a_2)| = |\{t_{i+2}, t_{i+5}\} \cap \{t_{i+1}, t_{i+2}, t_{i+3}, t_{i+4}\}| = |\{t_{i+2}\}| = 1$.

Our purpose is to dynamically anonymize the sliding window to make it satisfy ρ-uncertainty. We can use the static anonymization method for every transactional sliding window. However, because the sliding window is updated frequently and rapidly, an efficient algorithm is needed. When the sliding window is updated, it does not satisfy ρ-uncertainty because of the deletion and addition of transactions. We need to find and handle only the affected SARs that may violate ρ-uncertainty. We first introduce the suppression algorithm, and then propose the generalization algorithm with suppression.

A. SUPPRESSION ALGORITHM

To avoid the generation of false association rules [37], we use global suppression. A suppressed item is deleted from all transactions in a TSW_i which contain it. The initial sliding window is anonymized by *SuppressControl* in [18]. When the sliding window is updated, the anonymous sliding window may not satisfy ρ-uncertainty due to the removal and addition of transactions. We use Algorithm 1 to continuously handle it and make it satisfy ρ-uncertainty.

Let TSW_i' be the anonymous window of raw window TSW_i and I_{supp}^i contains the suppressed items from TSW_i. In line 1, the initial TSW_{i+1}' is obtained by sliding forward TSW_i' with step size p in the transactional data stream *TDS*, i.e., $TSW_{i+1}' = TSW_i' \backslash W_{del} \cup W_{add}$, where W_{del} is the set of deleted anonymous transactions and W_{add} is the set of added raw transactions. To ensure that there are no false association rules in the anonymous sliding window, in lines 2 to 6, if an item has been deleted in previous anonymized processes from the raw transactions about $TSW'_i \backslash W_{del}$, we delete the item in W_{add}. In line 8, we build the *ASRT* according to these transactions in W_{del} and W_{add} to find the *SAR*s which violate ρ-uncertainty, and then suppress them by suppressing some items in TSW'_{i+1}. Finally, we obtain the anonymous sliding window TSW'_{i+1} which satisfies ρ-uncertainty.

Algorithm 1 *SuppTDS*$(TSW'_i, I^i_{supp}, TDS, p, w, \rho, (F, \eta))$

```
Input: anonymous window $TSW'_i$; suppressed items set $I^i_{supp}$; transactional data stream $TDS$; step size $p$; window size $w$; privacy requirement $\rho$; soft set $(F, \eta)$ for $TSW'_i$;
Output: anonymous window $TSW'_{i+1}$;
1: slide $TSW'_i$ over $TDS$ to obtain $TSW'_{i+1} = TSW'_i \backslash W_{del} \cup W_{add}$;
2: for $a \in I^i_{supp}$ do
3:     if $i_{supp}$ does not appear in the raw transactions about $TSW'_i \backslash W_{del}$ then
4:         delete $i_{supp}$ from $I^i_{supp}$;
5:     else
6:         suppress $i_{supp}$ from $W_{add}$;
7: $I^{i+1}_{supp} = I^i_{supp}$;
8: return BuildASRTandSupp$((F, \eta), W_{del}, W_{add})$;
```

In Algorithm 2, we first initialize *ASRT*, find the violated *SAR*s whose antecedent length is 1, and suppress them by suppressing some items in TSW'_{i+1}. In lines 5 to 9, when the nodes of the *lev*th level are not $\emptyset$, we continue to build the next level nodes, find the violated *SAR*s whose antecedent length is *lev* and suppress them by suppressing some items.

Algorithm 2 *BuildASRTandSupp*$((F, \eta), W_{del}, W_{add})$

```
1: InitASRT$((F, \eta), W_{del}, W_{add})$;
2: Suppress(ASRT, lev);
3: $I^{i+1}_{supp} = I^{i+1}_{supp} \cup Supp_{ASRT}$;
4: $lev = 2$;
5: while the set of $lev$th nodes is not $\emptyset$ do
6:     BuildNextLevel(ASRT);
7:     Suppress(ASRT, lev);
8:     $I^{i+1}_{supp} = I^{i+1}_{supp} \cup Supp_{ASRT}$;
9:     $lev++$;
10: return $TSW'_{i+1}$;
```

In Algorithm 3, we first initialize *ASRT* which contains the root node and the first level nodes. I_w is the set of the first level nodes, which consist of sensitive items appearing in TSW'_{i+1}. Let $root \rightarrow \alpha \rightarrow i_1 \rightarrow \ldots \rightarrow i_i$ be a path in *ASRT*, where $\alpha \in I_w$ and $i_1, \ldots, i_k \in I$. The value of i_k is $F(\alpha) \cap F(i_1) \cap \ldots \cap F(i_k)$.

For every transaction t_{del} in W_{del}, we update (F, η) in line 3. That is, if $i_{del} \in I_{del}$, we delete t_{del} from $F(i_{del})$, where I_{del} is the set of items in t_{del}. From lines 4 to 6, if the set S_{del} of sensitive items in t_{del} is not $\emptyset$, for every $i_s \in S_{del}$, we update the values of its children contained in $I_{del} \backslash i_s$. If the value of a child is $\emptyset$, we delete it from *ASRT*. From Theorem 3.1(3), we know that any *SAR* whose antecedent and consequent are contained in t_{del} does not violate ρ-uncertainty, but we need to update the values of its antecedent if the *SAR* exists in the *ASRT*. From Theorem 3.1(1), for any *SAR* $i_{del} \rightarrow i_s$ whose antecedent i_{del} is contained in t_{del} and consequent i_s is not contained in t_{del}, it may violate ρ-uncertainty. Therefore, we add i_{del} to *ASRT* as a child of i_s if i_{del} is not a child of i_s in *ASRT* and $F(i_s) \cap F(i_{del}) \neq \emptyset$, as shown in lines 7 to 9. If i_{del} is a child of i_s, the deletion of t_{del} does not affect the value of i_{del}, and so we do not to handle it in this case.

For every transaction t_{add} in W_{add}, we update (F, η) in line 11. That is, if an item i_{add} appears in t_{add}, we add t_{add} to $F(i_{add})$. I_{add} is the set of items in t_{add}, and S_{add} is the set of sensitive items in t_{add}. From Theorem 3.2(3), we know that any *SAR* $i_{add} \rightarrow i_s$ whose antecedent and consequent are contained in t_{add}, may violate ρ-uncertainty. As shown in lines 12 to 14, if i_{add} is a child of i_s, then we update the value of i_{add}; otherwise, we add i_{add} as a child of i_s. In line 15, the algorithm returns *ASRT* with the affected *SAR*s whose antecedent length is 1.

Algorithm 3 *InitASRT*$((F, \eta), W_{del}, W_{add})$

```
1: initialize ASRT which contains the root node and the first level nodes $I_w$;
2: for $t_{del} \in W_{del}$ do
3:     update $(F, \eta)$;
4:     if $S_{del} \neq \emptyset$ then
5:         for $i_s \in S_{del}$ do
6:             update the values of $i_s$' children contained in $I_{del} \backslash i_s$, and delete the child from ASRT if its value is $\emptyset$;
7:     for $i_s \in I_w \backslash S_{del}$ do
8:         if $i_{del} \in I_{del}$ is not an child of $i_s$ in ASRT and $F(i_s) \cap F(i_{del}) \neq \emptyset$ then
9:             add $i_{del}$ as a child of $i_s$;
10: for $t_{add} \in W_{add}$ do
11:     update $(F, \eta)$;
12:     if $S_{add} \neq \emptyset$ then
13:         for $i_s \in S_{add}$ do
14:             update the values of $i_s$' children contained in $I_{add} \backslash i_s$, and add it as a child of $i_s$ if there exists $i_{add} \in I_{add} \backslash i_s$ which does not appear in ASRT;
15: return ASRT;
```

In Algorithm 4, we find the set SAR_{lev} of violated *SAR*s with antecedent size *lev* from the *ASRT*. Then we use the *payoff tree* [18] to delete the *SAR*s in SAR_{lev} by deleting some items, and these deleted items are saved to $Supp_{ASRT}$, as shown in line 4. In line 5, we update (F, η), *ASRT* and TSW'_{i+1} according to $Supp_{ASRT}$.

In Algorithm 5, when the set of the *lev*th level nodes is not $\emptyset$, we build the $lev + 1$th level nodes. For an affected *SAR* $\chi \rightarrow \alpha$, we know that $\chi \subseteq \alpha_{dchild}$, where α_{dchild} is

Algorithm 4 *Suppress*($ASRT$, lev, ρ)

```
1: for every $SAR$ $Rule$ in $ASRT$ do
2:     if $Rule$ violates $\rho$-uncertainty then
3:         Add it to $ASR_{lev}$;
4: $Supp_{ASRT} = payoff(ASR_{lev})$;
5: update $(F, \eta)$, $ASRT$ and $TSW'_{i+1}$ according to $Supp_{ASRT}$;
6: return $Supp_{ASRT}$;
```

the set of α' direct children. For every leaf i_{leaf} of the levth level in $ASRT$, there is an affected SAR $i_1 \dots i_{leaf} \rightarrow \alpha$, where $root \rightarrow \alpha \rightarrow i_1 \rightarrow \dots \rightarrow i_{leaf}$ is a path from $root$ to i_{leaf}. Next, for every sibling node i_{lsib} of i_{leaf}, we check whether the $NewRule$ $i_1 \dots i_{leaf} i_{lsib} \rightarrow \alpha$ violates ρ-uncertainty. If $value(i_{leaf}) \cap F(i_{lsib}) \neq \emptyset$, i.e., $i_1 \dots i_{leaf} i_{lsib} \alpha$ appears in TSW'_{i+1}, and there is a transaction which contains the antecedent of $NewRule$, denoted by $Ant(NewRule)$, but does not contain α in Win_{del}, i.e., consequent of $NewRule$, denoted by $Con(NewRule)$ (Theorem 3.1(1)) or there is a transaction which contains $i_1 \dots i_{leaf} i_{lsib} \alpha$ in Win_{add} (Theorem 3.2(3)), the confidence of $NewRule$ is affected. We add i_{lsib} as a child of i_{leaf}. Then, the algorithm returns $ASRT$.

Algorithm 5 *BuildNextLevel(ASRT)*

```
1: for every leaf $i_{leaf}$ of $lev$th level in $ASRT$ do
2:     for every sibling node $i_{lsib}$ of $i_{leaf}$ do
3:         if $value(i_{leaf}) \cap F(i_{lsib}) \neq \emptyset$ and ($\exists t_{del} \in W_{del}$
           ($Ant(NewRule) \subseteq t_{del} \wedge Con(NewRule) \notin t_{del}$)
           or $\exists t_{add} \in W_{add}$ ($NewRule \subseteq t_{add}$)) then
4:             add $i_{lsib}$ as a child of $i_{leaf}$;
5: return $ASRT$;
```

B. GENERALIZATION ALGORITHM WITH SUPPRESSION

Similar to suppression, we use global generalization to avoid the generation of false association rules [18]. For a generalization hierarchy tree $\mathcal{H}$, global generalization maps all instances of an item, as well as all items of the same subtree of hierarchy $\mathcal{H}$ to the same level in $\mathcal{H}$ [22]. The algorithm *TDControl* [18] anonymizes static transactional data by applying global generalization over the hierarchy tree $\mathcal{H}$ of the non-sensitive items in I_N along with selective global suppression of some items, to make the data satisfy ρ-uncertainty. In this paper, we use *TDControl* to anonymize the initial sliding window. When the sliding window is updated, the removal and addition of transactions may make the sliding window fail to satisfy ρ-uncertainty. We use Algorithm 6 to continuously handle it and make it satisfy ρ-uncertainty.

The particularization tree $\mathcal{T}$ is a part of $\mathcal{H}$ and contains the root of $\mathcal{H}$ [18]. The set of its leaves denotes the generalization case of the current window. In line 1, the initial TSW'_{i+1} is obtained by sliding forward TSW'_i with step size p in the transactional data stream TDS, i.e., $TSW'_{i+1} = TSW'_i \backslash W_{del} \cup W_{add}$. In lines 2 to 6, to ensure that there

Algorithm 6 *GeneSuppTDS(TSW'_i, I^i_{supp}, $\mathcal{T}$, TDS, p, w, ρ, (F, η))*

Input: Anonymous window TSW'_i; suppressed items set I^i_{supp}; particularization tree $\mathcal{T}$; transactional data stream TDS; step size p; window size w; privacy requirement ρ; soft set (F, η) for TSW'_i;
Output: Anonymous window TSW'_{i+1};

```
1: slide $TSW'_i$ over $TDS$ to obtain $TSW'_{i+1} = TSW'_i \backslash W_{del} \cup$
   $W_{add}$;
2: for $i_{supp} \in I^i_{supp}$ do
3:     if the items in $leaves(i_{supp})$ do not appear in raw
       transactions about $TSW'_i \backslash W_{del}$ then
4:         delete $i_{supp}$ from $I^i_{supp}$;
5:     else
6:         suppress these items in $leaves(i_{supp})$ from $W_{add}$;
7: generalize the items in $W_{add}$ according to current partic-
   ularization tree $\mathcal{T}$;
8: save the generalized values in $W_{add}$ to set $Gene$;
9: $BuildASRTandSupp((F_s, \eta_{I_s}), W^{I_s}_{del}, W^{I_s}_{add})$;
10: update $(F, \eta)$, $TSW'_{i+1}$, $W_{del}$ and $W_{add}$ according to
    $Supp_{ASRT}$;
11: $I^{i+1}_{supp} = I^i_{supp} \cup Supp_{ASRT}$;
12: $BuildASRTandGeneSupp((F, \eta), W_{del}, W_{add})$;
13: while $Gene \neq \emptyset$ do
14:     $comsplit = Special(Gene)$;
15:     if $comsplit$==0 then
16:         break;
17: return $TSW'_{i+1}$;
```

are no false association rules in the anonymous sliding window, if the items in $leaves(i_{supp})$ ($i_{supp} \in I^i_{supp}$) have been deleted in previous anonymized processes from the raw transactions about $TSW'_i \backslash W_{del}$, we delete them in W_{add}. Then, we generalize the items in W_{add} according to the current particularization tree $\mathcal{T}$, and save the generalized values in W_{add} to $Gene$. We do not generalize sensitive items, because that would reveal sensitive information [18]. In line 9, we call the *BuildASRTandSupp* function to suppress some sensitive items and to ensure that the SARs, which contain sensitive items only, do not violate ρ-uncertainty. Additionally, we update (F, η), TSW'_{i+1}, W_{del} and W_{add} according to $Supp_{ASRT}$. In line 12, we build the $ASRT$ to find the SARs, which violate ρ-uncertainty, and combine generalization and suppression to handle them. Then, we check whether information loss can be decreased by specializing the items in *Gene* in lines 13 to 16. Finally, we obtain the anonymous sliding window TSW'_{i+1}.

In Algorithm 7, we initialize $ASRT$ which contains the affected sensitive rules with antecedent length 1. In line 2, we combine generalization with suppression to handle the SARs which violate ρ-uncertainty. When the set of levth level nodes in $ASRT$ is not $\emptyset$, we continue to add nodes for the next level, find the violated SARs with antecedent length lev, and handle them by generalization and suppression, as shown

Algorithm 7 *BuildASRTandGeneSupp*$((F, \eta), W_{del}, W_{add})$

```
1:  $InitASRT((F, \eta), W_{del}, W_{add})$;
2:  $GeneOrSupp(ASRT, lev, \rho)$
3:  $I^{i+1}_{supp} = I^{i+1}_{supp} \cup Supp_{ASRT}$;
4:  $lev = 2$;
5:  while the set of $lev$th nodes is not $\emptyset$ do
6:      $BuildNextLevel(ASRT)$;
7:      $GeneOrSupp(ASRT, lev, \rho)$;
8:      $I^{i+1}_{supp} = I^{i+1}_{supp} \cup Supp_{ASRT}$;
9:      $lev++$;
10: return $TSW'_{i+1}$;
```

in lines 5 to 9. Finally, we obtain an anonymous sliding window TSW'_{i+1}.

In Algorithm 8, we first find the set SAR_{lev} of violated *SAR*s in *ASRT* whose antecedent size is *lev*. When $ASR_{lev} \neq \emptyset$, we find the item i_x in ASR_{lev} with the maximum *payoff ratio* [18]. Should we generalize or suppress it? We select the way which causes less information loss, as shown in lines 6 to 10. Due to generalization or suppression, there may exist *SAR*s whose antecedent length is less than *lev*. These *SAR*s do not violate ρ-uncertainty, so we delete them from ASR_{lev}. In line 11, we update (F, η), *ASRT*, TSW'_{i+1} and *Gene* according to $Supp_{ASRT}$ and $\mathcal{T}$. Then, we return $Supp_{ASRT}$.

Algorithm 8 *GeneOrSupp*$(ASRT, lev, \rho)$

```
1:  for every $SAR$ $Rule$ in $ASRT$ do
2:      if $Rule$ violates $\rho$-uncertainty then
3:          Add it to $ASR_{lev}$;
4:  while $ASR_{lev} \neq \emptyset$ do
5:      select item $i_x$ with maximum $payoff$ $ratio$;
6:      if $LosSupp(i_x) \leq LosGene(i_x)$ then
7:          suppress $i_x$ in $ASR_{lev}$;
8:          put $i_x$ into $Supp_{ASRT}$;
9:      else
10:         generalize $i_x$ in $ASR_{lev}$;
11:     delete the $SAR$s whose antecedent length is less than $lev$;
12: update $(F, \eta)$, $ASRT$, $TSW'_{i+1}$ and $Gene$ according to $Supp_{ASRT}$ and $\mathcal{T}$;
13: return $Supp_{ASRT}$;
```

In Algorithm 9, we use the *infoGain* function [18] to find the item i_{split} in *Gene* with the maximum information gain. If the information gain of i_{split} is less than 0, then there is no item to specialize; otherwise, we specialize i_{split}. That is, we remove i_{split} from *Gene*, and update (F, η) and TSW'_{i+1} with the children of i_{split}.

Algorithm 9 *Special*$(Gene)$

```
1:  bestsplit=0;
2:  for every $i_{gene} \in Gene$ do
3:      $gain = infoGain(Gene, i_{gene}, \mathcal{H})$;
4:      if $BestSplit < gain$ then
5:          $BestSplit = gain$;
6:          $i_{split} = i_{gene}$;
7:  if $BestSplit < 0$ then
8:      return 0;
9:  remove $i_{split}$ from $Gene$;
10: update $(F, \eta)$ and $TSW'_{i+1}$;
```

TABLE 2. The description for dataset.

dataset	$\|D\|$	$\|I\|$	$max\|t\|$	$avg\|t\|$
BMS-WebView-2	77512	3340	161	5.0
BMS-POS	306983	1177	5	2.65

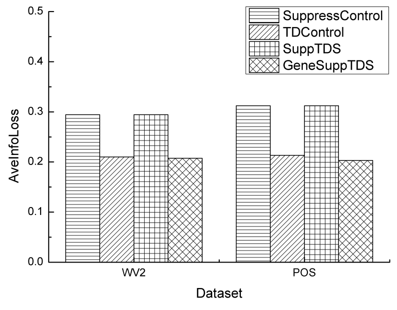
V. EXPERIMENTAL RESULTS

We evaluate the performance of our algorithms *SuppTDS* and *GeneSuppTDS* in terms of data quality and efficiency by comparison with approaches that anonymize the sliding window with batch processing by using the static anonymization methods *SuppressControl* and *TDControl* [18]. The algorithms are implemented in C++, and run on a computer with a four-core 3.2GHz CPU and 8GB RAM running Windows 7.

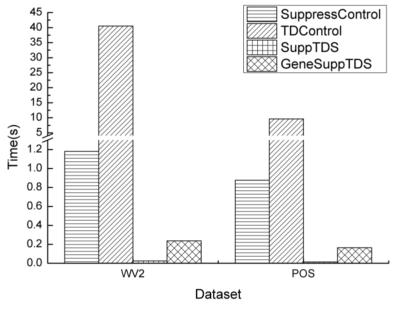
In our experiments, we use two datasets, *BMS-WebView-2* and *BMS-POS*, which are introduced in [38] and are benchmarks in the data mining community. Their characteristics are described in Table 2, where $max|t|$ and $avg|t|$ represent the maximum and average transaction size, respectively. As the items are not differentiated into sensitive and non-sensitive items, we randomly select 40% of the items as sensitive items. The default values of the window size w, step size p and privacy requirement ρ are 10000, 50 and 0.5, respectively.

We consider the information losses by varying the dataset, and the values of the window size w, step size p and privacy requirement ρ, as shown in Figs. 4 to 7.

Fig. 4(a) shows that the average information losses of *GeneSuppTDS* and *TDControl* are less than those of *SuppTDS* and *SuppressControl*, because the former combine generalization and suppression, while the latter use only suppression. The average information loss of *SuppTDS* is the same as that of *SuppressControl*, because we use the same method to handle the *SAR*s which violate ρ-uncertainty. The difference is that our *SuppTDS* finds the *SAR*s which violate ρ-uncertainty by scanning only W_{del} and W_{add}, while *SuppressControl* needs to scan the whole sliding window. The average information loss of *GeneSuppTDS* is near that of *SuppressControl*. For every sliding window, *GeneSuppTDS* and *TDControl* first call *SuppTDS* and *SuppressControl* to find and handle the *SAR*s which violate ρ-uncertainty and contain only sensitive items, respectively, and the information losses caused by them are same. Then, *TDControl* applies a top-down particularization method by initially assuming that all items are generalized to the top level of $\mathcal{H}$. It greedily selects the generalized item for particularization with max-

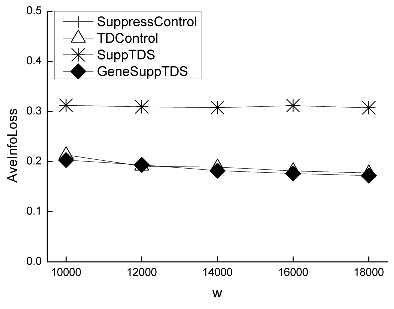


(a)

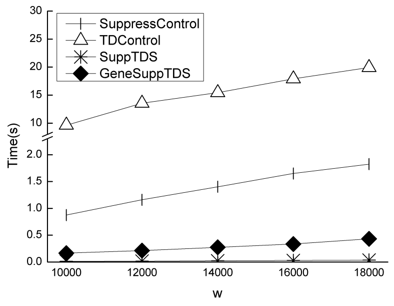


(b)

FIGURE 4. The performance by varying datasets. (a) Information loss. (b) Execution time.

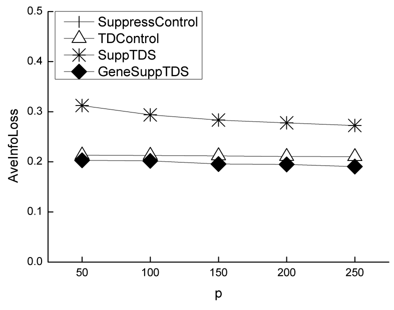


(a)

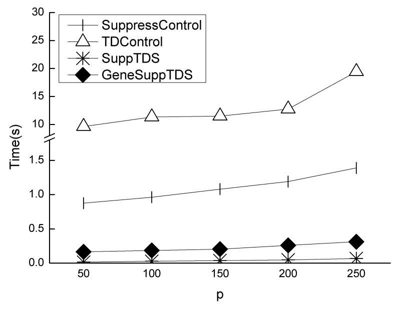


(b)

FIGURE 5. The performance by varying window size *w*. (a) Information loss. (b) Execution time.



(a)



(b)

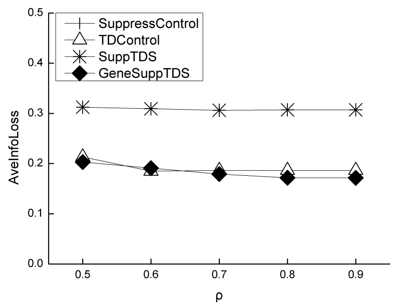
FIGURE 6. The performance by varying step size *p*. (a) Information loss. (b) Execution time.

imum information gain until the maximum information gain is less than 0. *GeneSuppTDS* utilizes the W_{del} and W_{add} to find the *SAR*s which violate ρ-uncertainty and considers generalization or suppression of items according to the information losses they cause. Finally, the algorithm checks whether the information loss can be reduced by particularizing some generalized items. *TDControl* and *GeneSuppTDS* both ensure that further particularization does not decrease the information loss. Therefore, their information losses are near for a sliding window.

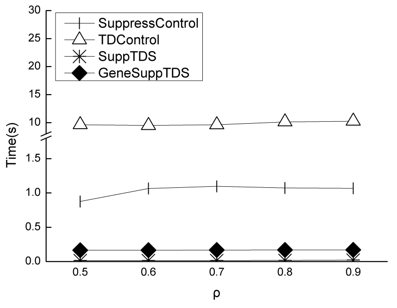
The execution time of algorithms is showed in Fig. 4(b). The efficiency of our *SuppTDS* and *GeneSuppTDS* is higher than that of *SuppressControl* and *TDControl*, because *SuppressControl* and *TDControl* need to consider the whole sliding window, while our algorithms scan only W_{del} and W_{add} to anonymize the sliding window. *GeneSuppTDS* requires more time than *SuppTDS*, because *GeneSuppTDS* first handles the *ASR*s which contain only sensitive items by suppression, then deals with the *ASR*s whose antecedent does not contain sensitive items by generalization and suppression, and finally particularizes some items to decrease information loss, while *SuppTDS* scans only W_{del} and W_{add} to find and handle the *SAR*s which violate ρ-uncertainty.

As the results in the two datasets show similar trends, next we use the *BMS-POS* data to test the performance of our algorithms and the comparison algorithms.

Form Fig. 5(a), we can see that the average information losses of the four algorithms decrease with increasing *w*. Because the larger *w* is, i.e., the more transactions there are in *w*, the lower the probability of a *SAR* violating ρ-uncertainty is. In Fig. 5(b), when *w* increases, i.e., the number of transactions in a sliding window increases, the execution time of the algorithms increases.



(a)



(b)

FIGURE 7. The performance by varying privacy requirement ρ. (a) Information loss. (b) Execution time.

As shown in Fig. 6(a), when the step size p increases, the average information losses of the four algorithms decrease. Since the intersection of two sliding windows decreases with increasing step size, the influence of suppression and generalization in previously anonymized processes decreases to the current sliding window. In Fig. 6(b), when p increases, the execution time of the algorithms increases. With increasing p, the number of newly arrived transactions in a sliding window increases, so there are more items to be handled. For *SuppTDS* and *GeneSuppTDS*, they need to handle more transactions because of the increase in W_{del} and W_{add}.

The performance of algorithms with varying privacy requirement ρ is shown in Fig. 7. Fig. 7(a) shows that the average information losses for the four algorithms decrease with increasing ρ, because the larger ρ is, the weaker the privacy preservation is, and the smaller the number of *SAR*s violating ρ-uncertainty is. From Fig. 7(b), we know that execution time is minimally affected by increasing ρ, because the algorithms can quickly handle the *SAR*s violating ρ-uncertainty.

VI. CONCLUSION

Data mining over transactional data streams is one of the most important problems with broad applications. However, the privacy problem in publishing transactional data streams has not attracted sufficient attention, and the existing static data anonymization methods cannot be directly applied to data streams. In this paper, we present two rapid anonymization methods with generalization and suppression to ensure that every anonymous sliding window satisfies ρ-uncertainty and that the information losses are near to those caused by batch processing with the existing static anonymization methods.

REFERENCES

[1] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Comput. Surv.*, vol. 42, no. 4, Jun. 2010, Art. no. 14.

[2] M. Terrovitis, J. Liagouris, N. Mamoulis, and S. Skiadopoulos, "Privacy preservation by disassociation," in *Proc. VLDB*, Istanbul, Turkey, 2012, pp. 944–955.

[3] R. Chen, B. C. M. Fung, P. S. Yu, and B. C. Desai, "Correlated network data publication via differential privacy," *VLDB J.*, vol. 23, no. 4, pp. 653–676, Aug. 2014.

[4] E. G. Komishani, M. Abadi, and F. Deldar, "PPTD: Preserving personalized privacy in trajectory data publishing by sensitive attribute generalization and trajectory local suppression," *Knowl.-Based Syst.*, vol. 94, pp. 43–59, Feb. 2016.

[5] I. Ozalp, M. E. Gursoy, M. E. Nergiz, and Y. Saygin, "Privacy-preserving publishing of hierarchical data," *ACM Trans. Privacy Secur.*, vol. 19, no. 3, Sep. 2016, Art. no. 7.

[6] Y. Xin, Z.-Q. Xie, and J. Yang, "The privacy preserving method for dynamic trajectory releasing based on adaptive clustering," *Inf. Sci.*, vol. 378, pp. 131–143, Feb. 2017.

[7] H. Zakerzadeh, C. C. Aggarwal, and K. Barker, "Managing dimensionality in data privacy anonymization," *Knowl. Inf. Syst.*, vol. 49, no. 1, pp. 341–373, Oct. 2016.

[8] R. Chen, N. Mohammed, B. C. Fung, B. C. Desai, and L. Xiong, "Publishing set-valued data via differential privacy," in *Proc. VLDB*, Seattle, WA, USA, 2011, pp. 1087–1098.

[9] J. Liu and K. Wang, "Anonymizing transaction data by integrating suppression and generalization," in *Proc. PAKDD*, Hyderabad, India, 2010, pp. 171–180.

[10] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, May 2017.

[11] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, "Sliding window-based frequent pattern mining over data streams," *Inf. Sci.*, vol. 179, no. 22, pp. 3843–3865, Nov. 2009.

[12] Y. Zhu and D. Shasha, "StatStream: Statistical monitoring of thousands of data streams in real time," in *Proc. VLDB*, Hong Kong, 2002, pp. 358–369.

[13] Z. Farzanyar, M. Kangavari, and N. Cercone, "Max-FISM: Mining (recently) maximal frequent itemsets over data streams using the sliding window model," *Comput. Math. Appl.*, vol. 64, no. 6, pp. 1706–1718, Sep. 2012.

[14] J. Kim and B. Hwang, "Real-time stream data mining based on CanTree and Gtree," *Inf. Sci.*, vols. 367–368, pp. 512–528, Nov. 2016.

[15] F. Nori, M. Deypir, and M. H. Sadreddini, "A sliding window based algorithm for frequent closed itemset mining over data streams," *J. Syst. Softw.*, vol. 86, no. 3, pp. 615–623, Mar. 2013.

[16] H. Chen, L. Shu, J. Xia, and Q. Deng, "Mining frequent patterns in a varying-size sliding window of online transactional data streams," *Inf. Sci.*, vol. 215, pp. 15–36, Dec. 2012.

[17] H. Ryang and U. Yun, "High utility pattern mining over data streams with sliding window technique," *Expert Syst. Appl.*, vol. 57, pp. 214–231, Sep. 2016.

[18] J. Cao, P. Karras, C. Raïssi, and K. L. Tan, "ρ-uncertainty: Inference-proof transaction anonymization," in *Proc. VLDB*, Singapore, 2010, pp. 1033–1044.

[19] M. Terrovitis, N. Mamoulis, and P. Kalnis, "Privacy-preserving anonymization of set-valued data," in *Proc. VLDB*, Auckland, New Zealand, 2008, pp. 115–125.

[20] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 6, pp. 1010–1027, Nov. 2001.

[21] Y. He and J. F. Naughton, "Anonymization of set-valued data via top-Down, local generalization," in *Proc. VLDB*, Lyon, France, 2009, pp. 934–945.

[22] G. Ghinita, Y. Tao, and P. Kalnis, "On the anonymization of sparse high-dimensional data," in *Proc. ICDE*, Cancun, Mexico, 2008, pp. 715–724.

[23] M. Xue, P. Karras, C. Raïssi, J. Vaidya, and K.-L. Tan, "Anonymizing set-valued data by nonreciprocal recoding," in *Proc. KDD*, Beijing, China, 2012, pp. 1050–1058.

[24] Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu, "Anonymizing transaction databases for publication," in *Proc. KDD*, Las Vegas, NV, USA, 2008, pp. 767–775.

[25] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan, "CASTLE: Continuously anonymizing data streams," *IEEE Trans. Depend. Sec. Comput.*, vol. 8, no. 3, pp. 337–352, May/Jun. 2011.

[26] J. Cao, B. Carminati, E. Ferrari, and K. Tan, "CASTLE: A delay-constrained scheme for k_s-anonymizing data streams," in *Proc. ICDE*, Cancun, Mexico, 2008, pp. 1376–1378.

[27] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: Privacy beyond K-anonymity," in *Proc. ICDE*, Atlanta, GA, USA, 2006, Art. no. 24.

[28] K. Guo and Q. Zhang, "Fast clustering-based anonymization approaches with time constraints for data streams," *Knowl.-Based Syst.*, vol. 46, pp. 95–108, Jul. 2013.

[29] J. Li, B. C. Ooi, and W. Wang, "Anonymizing streaming data for privacy protection," in *Proc. ICDE*, Cancun, Mexico, 2008, pp. 1367–1369.

[30] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia, "Continuous privacy preserving publishing of data streams," in *Proc. EDBT*, Saint-Petersburg, Russia, 2009, pp. 648–659.

[31] S. Kim, M. K. Sung, and Y. D. Chung, "A framework to preserve the privacy of electronic health data streams," *J. Biomed. Inf.*, vol. 50, pp. 95–106, Aug. 2014.

[32] W. Wang, J. Li, C. Ai, and Y. Li, "Privacy protection on sliding window of data streams," in *Proc. CollaborateCom*, New York, NY, USA, 2007, pp. 213–221.

[33] K. Al-Hussaeni, B. C. M. Fung, and W. K. Cheung, "Privacy-preserving trajectory stream publishing," *Data Knowl. Eng.*, vol. 94, pp. 89–109, Nov. 2014.

[34] N. Mohammed, B. C. M. Fung, and M. Debbabi, "Walking in the crowd: Anonymizing trajectory data for pattern analysis," in *Proc. CIKM*, Hong Kong, 2009, pp. 1441–1444.

[35] D. Molodtsov, "Soft set theory—First results," *Comput. Math. Appl.*, vol. 37, nos. 4–5, pp. 19–31, Feb./Mar. 1999.

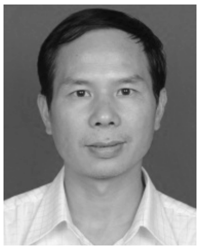
[36] T. Herawan and D. M. Mat, "A soft set approach for association rules mining," *Knowl. Based Syst.*, vol. 24, no. 1, pp. 186–195, 2011.

[37] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association rule hiding," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 4, pp. 434–447, Apr. 2004.

[38] Z. Zheng, R. Kohavi, and L. Mason, "Real world performance of association rule algorithms," in *Proc. KDD*, San Francisco, CA, USA, 2001, pp. 401–406.

JINYAN WANG received the B.Sc., M.Sc., and Ph.D. degrees from the School of Computer Science and Information Technology, Northeast Normal University, Changchun, China, in 2005, 2008, and 2011, respectively. She is currently an Associate Professor with the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China. Her research has been supported by the National Science Foundation of China. Her research interests include information security and automated reasoning.

CHAOJI DENG received the M.Sc. degree from the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China, in 2017. He is currently a Programmer with RaySharp, Zhuhai, China. His research interests include information security.



XIANXIAN LI received the Ph.D. degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2002. He was a Professor at Beihang University from 2003 to 2010. He is currently a Professor with the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China. His research interests include information security.