Privacy Preserving Pattern Discovery in Distributed Time Series

Josenildo Costa da Silva Matthias Klusch
German Research Center for Artificial Intelligence
Deduction and Multiagent Systems
Stuhlsatzenhausweg 3, 66121 Saarbrücken, Germany
{jcsilva, klusch}@dfki.de

Abstract

The search for unknown frequent pattern is one of the core activities in many time series data mining processes. In this paper we present an extension of the pattern discovery problem in two directions. First, we assume data to be distributed among various participating peers, and require overhead communication to be minimized. Second, we allow the participating peer to be malicious, which means that we have to address privacy issues. We present three problems along with algorithms to solve them. They are presented in increasing order of complexity according to the extensions we are pursuing, i.e. distribution and privacy constraints. As the main result we present our secure multiparty protocol for the privacy preserving pattern discovery problem.

1. Introduction

Mining time series has been an active research area which produced a lot of specialized algorithms (cf. [21, 12, 9, 7] to name a few). Many of these mining algorithms have a pattern problem as core activity, in particular, the discovery of unknown patterns.

In this paper we focus on the pattern discovery in a distributed scenario. Our main motivation in this work is to provide a solution to distributed scenario, which in our understanding has a large number of potential applications. The challenge is to address additional constraints of low communication cost, high scalability, and privacy preservation of data sources. To the best of our knowledge, no solution has been presented to handle the distributed case so far. A scenario where the privacy is an issue is in *Fraud Control*. Suppose a group of banks want to cluster the transaction history of its clients to find out if the behavior has something in common with illegal operations, money laundry for example. Since illegal operations are commonly spread over many institutions, this is a potential scenario for distributed pattern mining. Informally the problem is, *to find all frequent patterns given the union of the individual data owned by the different peers*.

A solution to this problem must fulfill the following requirements: (i) *low communication*:the volume of data exchanged among peers must be kept as low as possible; (ii) *correctness*; (iii)*privacy-preservation*. The rationale for the first requirement is that of reducing communications costs and idle time spent waiting for communication. When it comes to the correctness, the patterns found by a distributed algorithm must be the same found by a traditional algorithm running over a centralized union of the data sets. Finally, no sensitive data owned by a peer is disclosed to other peers. Here sensitive data may be data about individuals, trade secrets, buying patterns, etc.

Distributed data mining problems are commonly solved by transferring all data to a central location. However, in some scenarios transferring local data may not feasible. First, the data sets may be huge. Second, we cannot assume high speed networks. Third, centralized algorithms may not scale well up. Fourth, participants may not want to disclose data to other members of the mining group to protect individual privacy (individual records) in some cases, or to protect business advantage (institution privacy) in other cases.

Building local models at local sites and using some model aggregation approach to build a global model is a suitable alternative. More concretely, we could find locally a set of interesting patterns and compare with the results issued by other sites. However, there are at least two main limitations. Fist, this approach may not be correct, due the fact that some frequent patterns are spread over the data sets and can be discovered only when we consider the whole data set. Second, local model may allow data reconstruction, what is known as the inference problem. Finally, if a subset of the participants forms a collusion group, they may learn sensitive information from non-colluding participants.

Current work on pattern discovery do not address the distributed case and, more specifically, do not address privacy

issues. Privacy preservation is a very important property and has been a hot topic of investigation in databases, and more recently in data mining.

The research question in this work is how to adapt current pattern discovery solutions to address the distributed pattern search problem including the requirement of privacy preservation of local data sets.

Outline. The remaining of this paper is organized as follows. Pattern discovery in time series data is discussed in sections 2 (traditional) and 3 (distributed case). Privacy issues are discussed in section 4. Our algorithm along with an analysis of its privacy preserving properties is presented in section 5. Related works and Conclusion can be found in the sections 6 and 7 respectively.

2. Pattern Discovery in Time Series

A pattern in time series is an "interesting" subsequence of the time series. Here the term "interesting" may have different meanings, depending on the pattern problem one wants to address.

A well investigated problem is to find patterns similar to a given one, which is known as *query-by content* problem. Another class of pattern problem occurs when the pattern is unknown. In this case, if the goal is to identify frequently occurring pattern the problem is called *motif discovery* [16]. On the other hand, the problem of finding non-frequent patterns is defined as *surprise detection*. In this work we focus on the problem of discovery of *unknown frequent patterns* (a.k.a *motif discovery*) or simply *pattern discovery* in time series.

Pattern discovery problem has been extensively studied in bio-informatics, where the goal is to find frequent patterns in sequences of symbols, e.g. microarray data analysis [11]. Recently this problem was extended to handle real-valued data [16]. More formally the problem is to identify the k-most frequent pattern occurring in the time series.

Definition 1 *(Reoccurring subsequence) Given a real-valued time series X a reoccurring subsequence is one which reoccurs in different parts of the time series without overlap.*

Definition 2 *(k-Frequent Pattern) Given a real-valued time series X and a table with all reoccurring subsequences ordered by number of occurrence, a k-frequent subsequence is a reoccurring subsequence which is in the top k list of reoccurring subsequences.*

PROBLEM 1 (PDTS) *Given a real-valued time series X and k, an integer, the problem is to find the k-most frequent patterns occurring in X.*

A brute force algorithm to solve PDTS uses a sliding window to compare the current subsequence with all other possible subsequences in X. It has time complexity $O(|X|^2)$, where $|X|$ represents the size of X.

Other approaches to PDTS follows a three-step scheme (cf. [25, 16]):

1. Dimension reduction of the original time series X;
2. Discretization of the reduced time series X into a sequence of symbols S;
3. Pattern discovery algorithm using the resulting symbol sequence S.

This general scheme has the advantage of handling the high dimensionality of data, do not assume any knowledge on the structure of patterns and finally, provides a clear separation on the tree different tasks at hand.

We pursue this scheme focusing our attention on the third step. Our solution works as follows: given a sequence S, we compute the density of subsequences of S and output the list with the top k frequent subsequences.

Our approach exploits the fact that a density estimate can be used to find overcrowded regions in a hyperspace. If we take subsequences of time series and represent it as points in a multidimensional space, we can reduce the search for frequent subsequences, to the search for dense regions[1].

Our solution is based on the observation that the search for k-most frequent patterns of size w reduces naturally to the search for k-most dense regions on a $\mathbb{R}^w$ space (cf. algorithm 1).

Algorithm 1 PDTS

Input: k, X, n, w, Σ, r;
Output: $\mathcal{P}$;
1: $\mathcal{H} \leftarrow$ createHashTable(X, n, w, Σ);
2: $\varphi \leftarrow \mathcal{H}$.estimateDensity();
3: $\mathcal{P} \leftarrow \varphi$.getCenters(*k, r*);

In the step 1 the function createHashTable() works as follows. First create an empty hash table $\mathcal{H}$ and for each n-sized non-overlapping subsequence S of X, do the following:

(a) Using eq. (2) compute $\overline{S}$, which is composed of the averages of the subsequence S. This will reduce the dimensionality of the current subsequence S from n to $\frac{n}{w}$, where w is the size of the resulting strings. Actually, this operation (proposed elsewhere [15]) is known as piecewise aggregate approximation (PAA): $\overline{s_j} = \frac{w}{n} \sum_{k=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} s_k$

[1] This approach can solve at least the case where the subsequences have fixed size.

(b) This is the discretization step. Compute the string W from $\overline{S}$ by substituting an element $\overline{s}_j$ of S by a correspondent symbol σ_a in Σ. This procedure is accomplished by choosing break points $\{\beta_a\}$, such that $|\{\beta_a\}| = |\Sigma| + 1$, and such that each occurrence of a given value $\overline{s}_j$ of $\overline{S}$ has the same probability [16] assuming they are normally distributed. Finally, the substitution rule is applied:

$$W[j] = \sigma_a \qquad \text{iff } \beta_{a-1} < \overline{s_j} \leq \beta_a \tag{1}$$

(c) Finally, add an entry in $\mathcal{H}$ for the string W;

In the second step, the density estimation of the strings stored in the hash table, we can use any estimation method, e.g. kernel-based estimation. The requirement is that the density estimate function φ builds a non negative monotonic function over $\mathbb{R}$ and that the local maxima represent the most dense regions in the feature space. In practice it even does not need to be a true estimation, an approximation will do it.

The final step. Choose a set of strings, each of them representing a center of the k-most dense regions and call it $\mathcal{P}$. The regions are constrained to define a ball radius r. More formally, for a given density estimate φ we have:

$$\mathcal{P} = \{\sigma \mid \forall \sigma' \in \Sigma^w \, (\mid \sigma - \sigma' \mid \leq r \rightarrow \varphi(\sigma) > \varphi(\sigma'))\} \tag{2}$$

Theorem 1 *Algorithm PDTS produces no false positives.*

Proof Sketch. This is a result from the following facts. First, patterns correspond to local maxima in the density estimate and maxima correspond only to highly populated regions. Therefore a frequent pattern represents a frequently reoccurring subsequence (within a given range of dissimilarity). Second, given two different local maxima the one with higher density represent the pattern with higher frequency rate. Therefore the ordering of the k-most dense regions corresponds to the ordering of the k-most frequent subsequences. Finally, since we assume non-overlapping subsequences, each subsequence contributes only once to a given point in the density space.

□

3. Pattern Discovery in Distributed Time Series

Now we extend the pattern discovery problem to the distributed case including the assumption of split datasets and two constraints (*communication* and *correctness*).

PROBLEM 2 (DPDTS) *Given an integer k, and a set of sites $\mathcal{L} = \{L_i\}_{1 \leq i \leq P}$, each of them with a local time series X_i, the problem is to find the set $\mathcal{P}$ of the k-most frequent patterns occurring in $X = \bigcup_{i=1}^{P} X_i$, such that:*

1. *The total communication cost is minimized*
2. *The result using the distributed data X_i is the same if the algorithm runs using $X = \bigcup_{i=1}^{P} X_i$*

3.1. Underlying Assumptions

Time Series Semantics. An important point to mention here is that a time series X can be instantiated in the distributed setting in two different ways: (a) the time series at each local site measure the same variable; or (b) each time series measures a different variable. In this work we focus on the first case: same variable across the sites.

Architecture. We have at least (but not limited to) two choices of distributed architecture. First, we can assume a central entity. But it is highly dependent on the central point. Particularly in an open environment we cannot assume that it is not going to fail. Second, we can use a pure P2P network. This architecture is not dependent on a central entity, which means no central-point-fail problem. Since we want to avoid the dependence on a central entity, P2P seems the natural choice as the underling architecture of our solution.

3.2. A Density Based Solution to the DPDTS Problem

We exploit the fact that the density estimate is additive. Therefore we can compute it locally and cooperatively produce the global result.

Algorithm 2 DPDTS

Input: k, X_i, n, w, Σ, $\mathcal{L}$, r;
Output: $\mathcal{P}$;

```
1: H ← createHashTable(X, n, w, Σ);
2: φ_i ← H_i.estimateDensity();
3: φ ←cooperativeSum(φ_i, L);
4: P ← φ.getCenters(k, r);
5: broadcast(P);
```

DPDTS(k, X_i, n, w, Σ, $\mathcal{L}$, r) computes a set of k-frequent patterns occurring in the union of local time series X_i and time series owned by other peers in the group $\mathcal{L}$. X_i is the local dataset, n is the size used to generate subsequences, w is the number of symbols per string, i.e. the string size, Σ is the alphabet used to generate strings, and $\mathcal{L}$ is the set of peers forming the mining group. The parameter r is define the radius of the density ball to be used in the second step. As output DPDTS returns a set $\mathcal{P}$ with the *globally* k-most globally frequent patterns.

Step 1 (Local hash tables). Transform local time series in strings of a common alphabet Σ, and build a local hash table of strings.

Step 2 (Cooperative Sum). Cooperatively compute the set of globally k-most frequent patterns by summing up all local tables.

(a) compute the local density estimate $\varphi_i : \Sigma^w \rightarrow \mathbb{R}$ of the items in $\mathcal{H}_i$;

(b) cooperatively compute $\varphi = \sum_{i=1}^{P} \varphi_i$; The specific way of computing is dependent on the underlying architecture. In this case, using the architecture described in section 3.1, we have each site receiving the density estimates from its neighbor, adding its local density and sending it to the next neighbor in the mining group formed in the beginning of the protocol.

Step 3. (Choosing k-most frequent patterns) Choose a set of strings, each of them representing a center of k-most dense regions and call it $\mathcal{P}$. The regions are constrained to define a ball radius r:

$$\mathcal{P} = \{\sigma \mid \forall \sigma' \in \Sigma^w \, (\mid \sigma - \sigma' \mid \leq r \rightarrow \varphi(\sigma) > \varphi(\sigma'))\} \quad (3)$$

3.3. Performance Analysis of DPDTS

Time. The time complexity of DPDTS at a local peer is $\mathcal{O}(|X_i|)$, where $|X_i|$ means the size of the time series at peer L_i. There are $\lfloor \frac{|X_i|}{n} \rfloor$ subsequences in X_i. For each subsequence w arithmetical means are computed summing $\lfloor \frac{n}{w} \rfloor$ points for each mean, i.e. n steps. Additionally each arithmetical mean is substituted for a symbol, which takes w steps. The overall time cost is $\frac{|X_i|}{n}(n+w) = X_i(\frac{w}{n}+1)$ steps. Note that normally $w < n \ll |X_i|$. The discovery step, which is the search for the k-most dense regions, is independent of the size of X_i and is $\mathcal{O}(|\Sigma|^w)$.

Communication. Each peer sends 1 message to a neighbor peer and receives 1 message from another neighbor. There are only 2 rounds of messages, one of which informs the mining results. Each message has size $\mathcal{O}(|\Sigma|^w)$, for given global w and Σ.

4. Privacy Issues

There are scenarios where the local data cannot be exchanged to other parties. In this case we cannot use the solution presented in the previous section. To put this concern more clearly we define here the problem and discuss a privacy measure we can employ to quantify privacy in time series.

PROBLEM 3 (PP-DPDTS) *Given a set of sites $\mathcal{L}$ as in problem 2, with the same goal, we have to consider the following additional constraint: after the computation, for all $L_i, L_q \in \mathcal{L}$, it can be shown that L_i learns nothing about the data owned by L_q, with $i \neq q$.*

4.1. Privacy Measure

The question now is how to quantify the intuitive notion of privacy. A lot of different metric have been proposed in the privacy-preserving data mining literature (e.g. [2, 20, 13]). Since none of the proposed metrics are developed for the time series the question is whether we can pick one of the proposed metrics, or if we need a new metric. In some sense each metric is focusing on a specific aspect of privacy. In information theoretical the aspect is how much confident the attacker can be, given the entropy of the variable, using basically the privacy as the reciprocal from entropy. In more database oriented metrics, like the ideas used in k-anonymization, the focus in on how many points can be identified. In time series there is no identification issue, therefore we need to provide some probability bound on the reconstruction quality. So, we need an information theoretical based metric.

We follow the metric proposed by Agrawal & Aggarwal [1] to address time series data. The starting point is the entropy, which is can be interpreted as the size of an interval where our interest variable X and an uniformly distributed variable Y have the same amount of uncertainty, which is given by the differential entropy $H(X) = -\int_{\Omega} p(x) log_2(p(x))$ where Ω is the domain of X and $p(x)$ is its probability density function. The privacy of the time series X at time t, denoted by $PR(x[t])$, is the size of the interval of values X may assume, and it is given by:

$$\mathbf{PR}(x[t]) = 2^{H(X)} \quad (4)$$

The previous equation assumes that each point in the time series does not dependent from the previous points. When it is not the case, only the entropy $H(X)$ should take that in account and the main equation remains the same.

4.2 Measuring the Information Loss

The trade-off between privacy and utility is an important concern. Clearly, there is no use in running a privacy preserving data mining algorithm which produces no useful results due to the privacy constraints. On the other hand if the risk of disclosure is too high one may do not take part in a distributed data mining group. In the last section we discussed how to measure privacy. In the following we discuss how to measure utility.

There are basically two approaches to capture the utility of data. The first approach is to use some information loss function quantifying the amount of distortion due to the privacy preserving transformation imposed to the data. The second approach is to use some measure of quality on the mining results produced by the privacy preserving algorithm. This approach is better suited to classification or regression, where makes sense to talk about training dataset

and test data set. In our case, we are looking for unknown patterns. Therefore we pursue the information loss approach.

We use the *mean absolute error* (MAE) as information loss function. It is a direct comparison between the original dataset and the public data. MAE does a great job indicating the loss of information, in particular, when periodic patterns are destroyed, which leave us with larger information loss values. This will help the user to choose the trade-off between privacy and utility.

$$IL_{MAE} = \frac{\sum_{i=1}^{n} \mid x_i - x'_i \mid}{n} \quad (5)$$

where X is the original time series, n is the length of the original time series and X' is the result of transformations by the algorithm under analysis.

Other information loss functions are possible, e.g. comparing statistics (sample mean, variance, etc.) of the original and transformed data. Yet another approach is to use *mean squared error* or other distance measure. In our preliminary tests the *mean absolute error* produces a good indication of how many patterns have been wiped out by the transformation being equivalent to its alternatives.

5. Privacy Preserving Pattern Discovery in Distributed Time Series

5.1. Preliminaries

Homomorphic Encryption (HE) scheme allows for parties to perform arithmetical operation directly without decryption. Here we use Paillier scheme [22] which is an additive homomorphic. So, given two messages m_1 and m_2 the following holds: $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$.

The main steps of this scheme are:

Key Generation Let $N = pq$ be a RSA modulus and g be an integer of order αN mod N^2, for some integer α. The public key is (N, g) and the private key is $\lambda(N) = lcm((p-1),(q-1))$.

Encryption The encryption of message $m \in \mathbb{Z}_N$ is $E(m) = g^m r^N \ mod\ N^2$, with r randomly selected from $\mathbb{Z}_N$

Decryption Given a cipher text c the message is computed as follows:

$$m = \frac{L(c^{\lambda(N)} \ mod\ N^2)}{L(g^{\lambda(N)} \ mod\ N^2)}$$

where $L(u) = \frac{u-1}{N}$.

5.2. DPD-HE: Detailed Description

We provide the DPD-HE algorithm as a solution to the PP-DPDTS problem.

By representing subsequences of a time series as n-dimensional points in $\mathbb{R}^n$ we are able to reduce the search to frequent subsequences to the search for density populated regions on the given multidimensional space. The main idea is to exploit the linearity of the local density estimate to find patterns in a distributed fashion. Since density estimates have the additive property, we can combine local densities estimates and search for dense regions on a global density estimate. Of course the privacy of local data needs to be preserved to the maximum and therefore no data transfer is allowed. Moreover, even models are being protected by the use of secure multi-party computation, as the following sections describes.

Algorithm 3 Initiator

Input: k, X_i, n, w, Σ, $\mathcal{L}$, r;
Output: $\mathcal{P}$;

At the initiator do:
1: $(n, w, \Sigma) \leftarrow$ negotiateParameters($\mathcal{L}$);
2: $\mathcal{H}_1 \leftarrow$ createsHashTable(X_1, n, w, Σ);
3: $LDE_1 \leftarrow \mathcal{H}_1$.density($\theta, X_1$);
4: $(PK, SK) \leftarrow$ generateKeyPairs();
5: broadcast($\mathcal{L}$, PK);
6: $EGDE_{|\mathcal{L}|-1} \leftarrow$receive($L_{|\mathcal{L}|-1}$, *Encr*($PK$, $\sum_{j=1}^{|\mathcal{L}|-1} LDE_j$));
7: $GDE \leftarrow$Decr($SK, EGDE_{|\mathcal{L}|-1}$) + LDE_1;
8: $\mathcal{P} \leftarrow GDE$.findDensityCenters($k$,$r$);
9: **for** $i = 1; i < |\mathcal{L}|; i++$ **do**
10: broadcast($\mathcal{L}$,*Encr*($PK_i, \mathcal{P}$));
11: **end for**

Algorithm 4 Arbitrary Party

Input: k, X_i, n, w, Σ, $\mathcal{L}$, r;
Output: $\mathcal{P}$;

At an arbitrary party j do:
1: $(n, w, \Sigma) \leftarrow$ negotiateParameters($\mathcal{L}$);
2: $\mathcal{H}_j \leftarrow$ createsHashTable(X_j, n, w, Σ);
3: $LDE_j \leftarrow \mathcal{H}_j$.density($\theta, X_j$);
4: $PK \leftarrow$ receive(L_1);
5: $EGDE_{j-1} \leftarrow$receive(L_{j-1});
6: send(L_{j+1}, *Encr*(PK, LDE_j) + $EGDE_{j-1}$);
7: $\mathcal{P} \leftarrow$Decr(SK_j, *receive*(L_1));

Our algorithm has three main phases as described bellow.

Phase 1: preparation. The mining group $\mathcal{L}$ agrees on the parameters and each party computes the local density of the strings generated from the local time series data, just like in

the first step of algorithm 1. The initiator, which is the peer that proposes and coordinates the mining session, create a key pair and publicize its public key.

Phase 2: computation. Each party encrypts its local density estimate using the public key from the initiator. Then after it receives the encrypted partial sum from its neighbor, sums its local encrypted density estimate, and sends to the next neighbor in the sequence.

Phase 3: termination. When all parties added its local encrypted density estimate it is sent back to the initiator. The initiator decrypts it and performs an algorithm searching for the local maxima in the global density estimate. The result is sent back to the mining group.

Example. Consider the following (rather artificial) example. Let a group of p retailers each of them with a local time series data describing the weekly sales volumes for a period of one year. Assume this 52 point long time series to be basically a random walk with mean $\mu_p = 0$. Assume a given pattern of length 12 occurs at each site, e.g. $0.1 * sin(x)$ with $x \in (1, 6)$ sampled at 0.5 points. It means a period of 3 months of relative stable sales volume. Now, every peer in this group wants to discover patterns in the union of the data but is not willing to disclose local sales volume to its trade rivals.

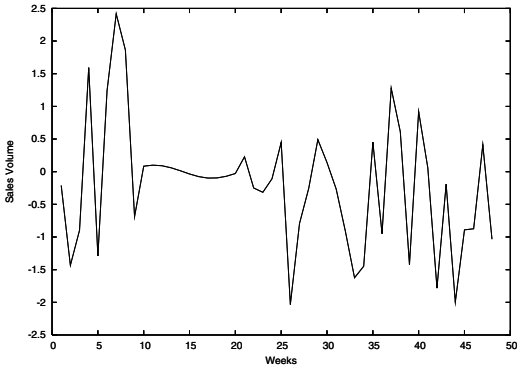

Figure 1. Weekly sales report represented as time series data

Using DPD-HE the group chooses the following parameters: n (the subsequence size) is set to 12, w (the string size) is set to 3 and the alphabet is chosen to be a 4 symbol set $\Sigma = \{a, b, c, d\}$. Firstly, the local sites generate local strings corresponding to the discretization of its local data. After that, the sites compute the local density of the local strings and securely compute the global density. In this example the values of the planted pattern will be translated to the subsequence “bba”. Since this string occurs

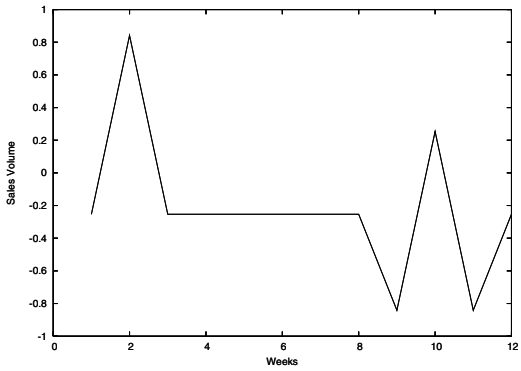

Figure 2. Local data after discretization step. Each point will be further converted to a symbol which lead us with the string ”bdbbbbbbacab”

more frequently than its neighbors in Σ^w (the set of strings with length 3), its density value will form a local maxima in the global density, the same happening to the other frequent strings.

By Using DPD-HE the group can rely on the fact that the sales volume will remain undisclosed since the protocol does not transfer any data and the cryptography ensures that no peer outside the group will be able to reconstruct the info exchanged, namely, the partial sum of local densities and the final results

Complexity Analysis The computational complexity of either algorithms 3 and 4 is $\mathcal{O}(|X_i|)$ because the computation of the density dominates over the other steps.

There are two rounds of communication: the cooperative computation phase and the broadcast of the output. Therefore the round complexity is $\mathcal{O}(1)$.

The message size in each round is $\mathcal{O}(\Sigma^w)$ which is independent of the input size.

5.3. Privacy and Information Loss

Raw Data By construction the DPD-HE does not transmit raw data. The only information transmitted is the encrypted density estimate.

Disclosure Given a set of global patterns $\mathcal{P}$ the members of the mining group can try to infer the original values X which originated the set $\mathcal{P}$. Since the discretization step is the main responsible for the privacy, we intuitively know that the more symbols in the alphabet Σ, the less privacy we get, for the discretized version tends to get the "shape" of the original data. The next result shows how the size of Σ influences the privacy of a single point in X.

Theorem 2 *Let Σ be an alphabet of symbols used by the DPD-HE protocol. Let $\{\beta_j \in \mathbb{R}\}_{j=1}^{|\Sigma+1|}$ be a set of breakpoints which divides the normal curve in $\mid \Sigma \mid$ equiprobable regions. Let X be a time series and $X' \in \Sigma^w$ be its transform according to the discretization step of algorithm 1. For a given point $x[t]$ if its transformed counterpart $x'[u]$ is known, than its privacy level is given by:*

$$PR(x[t]) = \mid \beta_{j+1} - \beta_j \mid \tag{6}$$

Proof Sketch. This is a consequence of the discretization step.

Let $\sigma_j \in \Sigma$ be the symbol at point $x'[u]$. Since we know that the symbol σ_j comes from the substitution rule (cf. step (b) of algorithm 1), we know that the subsequence $x[t], \ldots, x[t+n]$ lies in the interval (β_j, β_{j+1}). In the absence of further information, the only suitable option is to model $x[t]$ as a random variable uniformly distributed in the given interval, i.e. $x[t] \sim U(\beta_j, \beta_{j+1})$. Now, using the eq. 4 we have

$$\begin{aligned}\mathbf{PR}(x[t]) &= 2^{H(x[t])} = 2^{\int_{\beta_j}^{\beta_{j+1}} p(x) log_2 p(x) dx} \\ &= 2^{log_2(\beta_{j+1} - \beta_j)} \\ &= \mid \beta_{j+1} - \beta_j \mid\end{aligned}$$

□

In other words, using the above theorem one party in the mining group may define the minimum amount of privacy it wants to have on its local data, by setting the maximal size of the alphabet Σ, and consequently the size of the intervals defined by the breakpoints $\{\beta_j\}$. Otherwise the peer doesn't take part in the mining group because the privacy level is below its local restrictions. Recall that the patterns in $\mathcal{P}$ represent only subsequences the true data X. Therefore, a malicious peer cannot reconstruct all the points in X but only the points represented in the patterns set.

Eavesdropper Note that an eavesdropper cannot learn anything from the data owned by the group members from the data exchanged during the protocol.

The overall security is a consequence of the security of the Paillier encryption scheme, which was shown to be semantically secure elsewhere [22]. Therefore an eavesdropper cannot reconstruct the global density estimate or even a partial estimate sent among the parties. As a consequence, an eavesdropper cannot reconstruct the original data, nor estimate confidence intervals nor get information a specific site.

Ownership Finally, when it comes to ownership the protocol does not reveal the ownership of any X_i's by construction, provided there is a majority of *non-colluding* peers in the mining group. We leave, however, the study of this issue to further research.

Information Loss The information loss of the DPD-HE is given by

$$IL_{MAE}(X, X') = \frac{\sum_{i=1}^{n} \mid x_i - x'_{\lfloor \frac{w}{n}(i-1) \rfloor + 1} \mid}{n} \tag{7}$$

where X is the original time series, X' is the result of substituting the symbols in the resulting string by its corresponding breakpoint (see algorithm 1, step 1 (b)), and n is the size of the PAA sequence and w is the size of the resulting string. The involved index expression assures that each point in the original time series X is compared with the correct element in X', which is smaller than X.

6. Related Work

Works on privacy preserving data mining follows three main approaches. *Sanitation*, aims to modify the dataset such that sensitive patterns cannot be inferred. It was developed primarily to association rule mining (cf. [3, 24]). The second approach is *data distortion*, in which the true value of any individual record is modified while keeping "global" properties of the data (cf. [8, 2] among others). Finally, *SMC-based* approaches apply techniques from secure multi-party computation (SMC), which offers an assortment of basic tools for allowing multiple parties to jointly compute a function on their inputs while learning nothing except the result of the function. In a SMC problem we are given a distributed network with each party holding secret inputs. The objective is to compute a function with the secret inputs ensuring that no party learns anything but the output. The general SMC problem was investigated by Goldreich et. al [10]. Latter Lindell and Pinkas showed that privacy-preserving data mining problems could be solved using techniques of SMC [18, 23]. Many applications of SMC to data mining have been proposed so far (cf. [6], [14], to name a few).

When it comes to privacy metrics there are many approaches. The information theoretical approach defines privacy as the length of the interval which a random variable is generated from [1]. The metric is based on the differential entropy of the random variable. Computational approach involves proving the impossibility of reconstruction of sensitive data from a set of queries posed by an adversary in a given adversary model [4]. Research on the trade-off between data quality and privacy has been investigated among others by Domingo-Ferrer in the context of statistical disclosure control for microdata [5].

Many approaches to PDTS problem have been proposed. Mörchen and Ultsch [21], for example, proposes using a

grammar-based approach and Kadous [12] suggests clustering subsequences of the original time series to find prototypical shapes. These approaches, however, have some disadvantages. The grammar-based approach is too dependent on the knowledge of the possible patterns to be discovered and the clustering-based approach has been shown to be very problematic [17]. Liu et al. [19] proposed effective heuristics to solve PDTS problem defining patterns size m as multidimensional points in $\mathbb{R}^m$.

7. Conclusion and Outlook

We presented here the problem of finding frequent patterns in distributed time series data. We also presented an approach to solve this problem by means of the DPD-HE algorithm, which is linear in the size of datasets and has communication complexity independent of the size of dataset. We pursue a density-based approach, that uses a discretized version of the original data to compute a density estimate of the candidate patterns, and proposed a secure multiparty-computation protocol to ensure the data secrecy. Additionally no peer receives original data, which ensures a provable level of privacy to each participant of the mining group against a potential malicious peer.

As future research we want to investigate how to extend the current technique to handle patterns with variable length and multivariate time series.

References

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *20th ACM PODS*, pages 247–255, Santa Barbara, Califonia, May 2001.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proc. of 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 45–52, Chicago,IL, November 1999.

[4] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *ACM PODS '03*, pages 202–210, New York, NY, USA, 2003. ACM Press.

[5] J. Domingo-Ferrer, A. Oganian, and V. Torra. Information-theoretic disclosure risk measures in statistical disclosure control of tabular data. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 227–231, Washington, DC, USA, 2002. IEEE Computer Society.

[6] W. Du and Z. Zhan. Building Decision Tree Classifier on Private Data. In *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14 of *CRPIT*, pages 1–8, Maebashi City, Japan, 2002. ACS.

[7] M. Elfeky, W. Aref, and A. K. Elmagarmid. Using convolution to mine obscure periodic patterns in one pass. In *9th EDBT*, volume 2992 of *LNCS*, pages 605–620, Heraklion, Crete, Greece, March 14-18 2004. Springer.

[8] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of 8th ACM KDD*, Edomonton, Alberta, Canada, 2002.

[9] P. Geurts. Pattern extraction for time series classification. *LNCS*, 2168:115–127, 2001.

[10] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *In Proc. of the 19th annual ACM conference on Theory of computing*, pages 218–229. ACM Press, 1987.

[11] K. L. Jensen, M. P. Styczynski, I. Rigoutsos, and G. N. Stephanopoulos. A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22:21–28, 2006.

[12] M. W. Kadous. Learning comprehensible descriptions of multivariate time series. In *Proc. 16th International Conf. on Machine Learning*, pages 454–463. Morgan Kaufmann, San Francisco, CA, 1999.

[13] M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *KDD '04*, pages 599–604, New York, NY, USA, 2004. ACM Press.

[14] M. Kantarcioglu and J. Vaidya. Privacy preserving naive bayes classifier for horizontally pertitioned data. In *IEEE ICDM Workshop on Privacy Preserving Data Mining*, pages 3–9, November 2003.

[15] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2000.

[16] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. In *Proc. of the Second Workshop on Temporal Data Mining*, Edmonton, Alberta, Canada, July 2002.

[17] J. Lin, E. Keogh, and W. Truppel. Clustering of streaming time series is meaningless. In *Proc. of the 8th ACM DMKD*, pages 56–65, New York, NY, USA, 2003. ACM Press.

[18] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Lecture Notes in Computer Science*, 1880:36–54, 2000.

[19] Z. Liu, J. X. Yu, X. Lin, H. Lu, and W. Wang. Locating motifs in time-series data. In *9th PAKDD*, volume 3518 of *LNCS*, pages 343–353, Hanoi, Vietnam, May 18–20 2005. Springer.

[20] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *Proc. of the 3rd IEEE ICDM*, Florida, 2003. IEEE CS.

[21] F. Moerchen and A. Ultsch. Discovering temporal knowledge in multivariate time series. In *Proc. GfKl 04*, Dortmund, Germany, 2004.

[22] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592:223–??, 1999.

[23] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.

[24] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid. Privacy preserving association rule mining. In *Reseach Issues in Data Engineering (RIDE)*, 2002.

[25] Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of time-series motif from multi-dimensional data based on mdl principle. *Machine Learning*, 58:269–300, 2005.