On Privacy in Time Series Data Mining

Ye Zhu Yongjian Fu
Department of Electrical and Computer Engineering
Cleveland State University
Cleveland OH 44115-2214, USA

September 30, 2007

Abstract

Traditional research on preserving privacy in data mining focuses on *time-invariant* privacy issues. With the emergence of time series data mining, traditional *snapshot-based* privacy issues need to be extended to be multi-dimensional with the addition of *time dimension*. We find current techniques to preserve privacy in data mining is not effective in preserving time-domain privacy. We present data flow separation attack on privacy in time series data mining, which is based on blind source separation techniques from statistical signal processing. Our experiments with real and synthetic data show that this attack is effective. By combining the data flow separation method and the frequency matching method, an attacker can identify data sources and compromise time-domain privacy. We propose possible countermeasures to the data flow separation attack in the paper.

1 Introduction

With the popularity of data mining, privacy issues have been a serious concern. Many approaches have been proposed to preserve privacy in data mining [1, 2, 3, 4, 5, 6]. Most research focuses on the privacy of data. The goal of these research is to mine data while protecting the identity of data owners. Various approaches have been proposed to conduct data mining without breaching of privacy. However, privacy issues studied in previous research are on time-invariant data which do not change over time. In other words, the data can be viewed as a snap-shot of objects.

Time-domain data mining becomes popular recently. The goal of time-domain data mining is to find out pattens contained in time domain data [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. In the context of time-domain data mining, the data to be mined is labeled with timestamps. We call such data time series data. One example is the daily stock price. For time series data, because of the special nature of the data, the privacy goes beyond the protection of data. In this paper, when the meaning of privacy is unclear from context, we call the privacy in time-invariant data mining *snap-shot privacy*, and the privacy in time series data mining *time series privacy*.

We focus on time series privacy issues in this paper. As snap-shot privacy issues arise from snap-shot based data mining, time-domain privacy issues arise from time-domain data mining. Time series privacy issues concern about changes in data over time. We need to protect data, as well as its properties in time and frequency domains. For example, sales data on a car model changes over time, but the manufacturer of the car model will worry about sharing the sales data with data miners because the sales data may indicate changes in financial situation or marketing strategies of the manufacturer over time. Another example is that a store may not be willing to share its sales data because a data miner may find out promotion periods of the store by checking periodicities contained in the data provided by the store. We argue that privacy in time series data involves protection of properties in time domain such as peak, trough, and trend, and properties in frequency domain, such as periodicity. Such properties reveal lots of information, even though they do not reveal data.

Two common approaches have been proposed to preserve snap-shot privacy in data mining. One approach is data perturbation in which data to be mined is modified to protect privacy. The other approach is data partitioning in which data is split among multiple parties and each party only see its share of the data. One method in data perturbation approach is aggregation in which time series data from different sources are aggregated and given to data miners. This can prevent data miners from finding private information about individual sources. For example, auto manufacturers usually do not want to publish daily, monthly or yearly sales data of individual car model because too much sensitive information is contained in the time-series data. Instead, trusted market research companies aggregate sales data of different car models made by different auto manufacturers and publish these aggregated data for data mining or market study. These time series data can be aggregated in different ways such as according to vehicle types or vehicle features for different purposes.

In this research, we found that current techniques to protect snap-shot privacy were largely ineffective under data flow separation attack, which can separate aggregated data and separate noise from original data. The data flow separation attack employs the *blind source separation* model [17], which was originally defined to solve *cocktail party problem*: blind source separation algorithms can extract one person's voice signal given the mixtures of voices in a cocktail party. The blind source separation algorithms solve the problem based on the independence among voices of different persons. Similarly, for time-domain data mining, one can use blind source separation algorithms to separate independent time series data generated from different sources.

The contributions of this paper can be summarized as follows:

- We introduce the concept of privacy in time series data mining. Because of the nature of time series data, privacy issues in time series data mining go beyond these in snap-shot data mining, especially privacy in time and frequency domains. We believe it is important to preserve privacy in time series data as well as in snap-shot data.

- We present *data flow separation attack* and show that aggregation is not always enough to protect time series privacy. We use experiments on real data to show that data flow separation attacks are effective.

- We present *frequency matching attack*, a further attack based on data flow separation attacks, which can fully disclose sensitive information of data sources.

- We analyze and discuss the pros and cons of countermeasures to data flow separation attacks.

The rest of the paper is organized as follows: Section 2 reviews the related work in privacy preserving data mining and time series data mining. We list time series privacy issues in Section 3. Section 4 outlines the threat model. In Section 5, we introduce the data flow separation attack. We will also describe the frequency spectrum matching that can be used as further attacks. In Section 6, we use experiments on real stock data to show the effectiveness of the data flow separation attack. Section 7 discusses the application of data flow separation attack in different settings and countermeasures for data flow separation attack. We conclude this paper in Section 8, with remarks on extensions of this work.

2 Related Work

In this section, previous research on privacy preserving data mining and time series data mining is summarized.

2.1 Privacy Preserving Data Mining

The objective of privacy-preserving data mining is to find interesting patterns without violating privacy, i.e., protecting data privacy in data mining process. On the one hand, we want to protect individual data's identity. On the other hand, we want to find general patterns about data. In other words, privacy concerns individual data, while data mining concerns aggregate data. Though the two do not directly conflict with each other, care must be taken as data mining requires revelation of data. It is important to balance privacy protection and data mining.

The main approaches to privacy-preserving data mining can be categorized into two types: data perturbation and data partitioning. The former modifies data to be mined to protect privacy. The latter splits data among multiple parties which do not share their data. Each of the two approached are explained below.

In data perturbation approaches, original data is modified by data obscuration or by adding random noise. An example of data obscuration is replacing values of a continuous variable with ranges. Random noises are from known distributions, such as even distribution or normal distribution. The modified data is given to data miner for knowledge discovery.

In [1], Agrawal and Srikant proposed a classification algorithm which did not need individual data's identity. They added random noises to original data and built decision trees from the modified data. The noise was either Gaussian or uniform distributed. They proposed an iterative algorithm which could approximate the distribution of original data from the perturbed data. The algorithm might be applied on global data, on each class, or on each node locally. They found that when the algorithm was applied on each class or on each local node, a decision tree with similar accuracy to that from the original data could be built.

Evfimievski et al. proposed an approach for privacy preserving mining of association rules [2]. They defined and mathematically formulated a form of privacy breaches. They pointed out that previous randomization methods could not totally prevent this kind of privacy breaches. They proposed a randomization method, called select-a-size, which added items to transactions to make up false frequent itemsets to be mixed with true frequent itemsets. This would prevent one from finding out true frequent itemsets, and therefore items in transactions. The method let users control the balance between meaningful association rules and privacy breaches. Using partial support, they developed an algorithm to find frequent itemsets in randomized data. Results from their experiments showed that the algorithm could find most true frequent itemsets within the limits of privacy breaches.

Du and Zhan proposed to use randomized response technique for protecting privacy of data [3]. They generalized the classical random response technique by allowing the attribute to be a logical expression. Using the proposed technique, decision trees can be built from randomized data with precisions similar to these from original data. One limitation is that the data must be binary. They also showed the relationship between randomness and recovered precision. Not surprisingly, the precision of decision trees decreased when the randomness level increased.

In [18], Huang et al. argued that even with random noise, data privacy might be compromised when data correlation was exploited. They developed a data reconstruction method based on Principal Component Analysis (PCA). When there were high correlations among attributes, their method could apply PCA to find more significant components among the attributes and reconstruct data from these components. The insignificant components were thrown away. Since noises were independently added to attributes, throwing away insignificant components will reduce a lot of noises without losing much original data. They also proposed a Bayesian based method for data reconstruction, which worked even when attribute correlations were low. Based on their analysis and experiments, they proposed a randomization approach where noises had correlations similar to these of data.

Zhu and Liu proposed a generalized approach to randomization [19]. They pointed out that the randomization problem is the same as the mixture model problem in statistics. They showed that the randomization in [2, 1] were special cases of the mixture model. Using the results from the mixture model, they presented two methods to reconstruct the density distribution of original data. They also presented an interval based and an entropy based metrics for privacy. Finally, they presented two problems that must be solved to obtain optimal randomization.

In data partitioning approaches to privacy preserving data mining, the original data is distributed among multiple sites, either by the partitioning of centralized data or by the nature of data collection. The data mining process is split into local computation at individual sites and global computation. During the process, each party does not see other party's data, but cooperates to find global patterns. Data is usually partitioned vertically or horizontally. In vertical partitioning, each party holds all tuples, but only a subset of attributes. In horizontal partitioning, each party holds a subset of tuples with all attributes. In many cases, secure multi-party computation [20] is employed. A lot of them use encryptions too.

Lindell and Pinkas first introduced secure multiparty computation as a technique for privacy preserving in data mining [4]. The data was horizontally partitioned between two parties. A privacy preserving algorithm for decision tree construction was proposed, which was based on the ID3 algorithm [21]. They introduced three protocols for privately computing the global information gain of any attribute. In addition, they presented how to privately handle terminal conditions, i.e., when all tuples in a node belonged to a class and when all attributes had been used in nodes. Based on these, they proposed a private secure ID3 algorithm.

Vaidya and Clifton proposed a method for mining association rules in vertically partitioned data [5]. Their method modified the Apriori algorithm [22]. Assuming two parties, the method used a scalar product protocol to compute the global supports of itemsets, without revealing supports of itemsets at each party.

Kantarcioglu and Clifton proposed a privacy preserving algorithm for mining horizontally partition data [6]. They assumed three or more parties, each of which held a subset of the data. The parties wanted to learn global association rules without revealing their data or local associations. The proposed algorithm extended a distributed association rule mining algorithm [23]. Two secure protocols were introduced, one for computing union of globally frequent itemsets, another for computing the global supports of itemsets. The main ideas in the first protocol were to encrypt locally frequent itemsets and to alternate itemsets exchange. In the second protocol, the difference between actual support and minimum support at each site was accumulated with a random number introduced by the first party.

Vaidya and Clifton proposed a privacy-preserving k-means clustering algorithm [24]. They assumed the data was vertically partitioned and there were at least three parties. They developed a secure algorithm to find closest cluster, which employed the secure permutation algorithm [20]. Also, a secure algorithm for checking terminal condition, when the means were stable, was developed.

Jagannathan and Wright proposed a privacy-preserving k-means clustering algorithm for arbitrarily partitioned data [25]. They assumed there were two parties. In their model, data tuples and attributes were split arbitrarily between the two partied. In other words, data was viewed as a two dimensional (tuple, attribute) table. Every cell in the table might belong to any one of the parties. In addition to secure algorithms for finding closest cluster and for checking terminal condition, they also developed a secure algorithm for computing means after each iteration. The algorithms used random share of variables, secure scalar product [20], and Yao's protocol [26].

Wright and Yang proposed a secure algorithm for learning Bayesian network structures [27]. The data was partitioned vertically between two parties. Based on the K2 algorithm [28], their algorithm included secure computing for the scoring function, which decided whether a link should be added to the Bayesian network structure, and if so, which link to add.

In the above data partitioning algorithms, all parties were assumed to be semi- honest. That is, every party would faithfully follow the protocol or algorithm, but tried to learn as much as possible about others.

As discussed above, past research in privacy preserving data mining focuses on privacy of raw data. Though privacy of derived information has been mentioned [18], we are not aware of any research in time series privacy. We hope to raise the awareness of time series privacy issues by this paper.

2.2 Time Series Data Mining

Research in time series data mining mostly focuses on data preprocessing techniques, such as transformation, indexing, feature extraction, and feature reduction. Work has been also been done in related techniques such as representation and similarity metric. Because time series data is usually large and noisy, direct application of data mining algorithms on raw data is time-consuming and gives unreliable results. Therefore a lot of attentions have been paid on preprocessing techniques that facilitate data mining tasks. The data mining tasks studied by researchers include subsequence matching, classification, clustering, and association rule mining.

Agrawal et al. pioneered the work on sequence matching in their seminal paper [7]. They tackled the problem of finding sequences that match a query sequence. To efficiently find matching sequences, they first applied Discrete Fourier Transform on sequences to get their coefficients. The first few significant coefficients were extracted and saved in an R*-tree [29]. The search for matching sequences was then done by searching in the R* tree. This enabled faster search without missing any potential matching. The false positives were eliminated in the post processing by comparing the actual sequences.

Faloutsos et al. extended the idea in [7] to allow query sequence shorter than the sequences in database[8]. The initial steps were similar. Each sequence was mapped into a series of subsequences using a sliding window. The subsequences were processed by Discrete Fourier Transform. After significant coefficients were extracted, instead of indexing them directly, the minimum bounding regions of the coefficients were indexed. This improved search because consecutive subsequences were usually similar and shared minimum bounding regions.

Agrawal et al. proposed an approach to deal with irregularities in sequence matching [30]. When matching two sequences, their approach allowed gaps, scaling, and translation of sequences. They defined the shortest subsequence without gap as atomic windows. First, matching atomic windows were found between the two sequences. Second, consecutive matching atomic windows were concatenated to form longer matching subsequences, as long as gaps were within a threshold and scaling was similar for all matching pairs. Last, their algorithm found the best overall matching among the matching subsequences from the previous step.

Das at el proposed a method for mining association rules from time series data [9]. First, they used a sliding window to transform a sequence into subsequences. Second, they clustered these subsequences into groups. Third, the subsequences were replaced by their corresponding cluster number. Finally, they ran an association rule mining algorithm to find rules in the discretized data.

Geurts proposed an approach for classification of time series data [10]. Given a set of series, he used a greedy algorithm to divide each series into segments. The algorithm found cutoff points for segments by minimizing variances in the resulting segments. Each segment was discretized by its mean. A decision tree classifier was built using the segments. One series from each class was chosen at each node splitting. The subsequences in the chosen series were tested as splitting conditions, and the one which minimized classification error was selected for splitting.

Keogh and Lin argued that clustering of subsequences of a time series would not give meaningful clusters [11]. They demonstrated that subsequence clustering resulted in clusters no better than clusters from random data. They revealed the hidden constraint on subsequence clustering, i.e., the weighted average of cluster means must equal the global mean. Instead of clustering subsequences, they proposed to find frequent subsequences, which they called motif.

Ihler et al. built a framework for modeling count data where the time series data was the count of some human activity [12]. They argued that the total count could be explained by normal activity and special events. The normal activity could be modeled by a Poisson distribution. The special events could be modeled as a Marcov process.

Morchen and Ultsch proposed a persistence based approach to segmentation of time series data [13]. They assumed that each segment was representing an underlying state. The segmentation was to find a set of most persistent states. A states persistence was measured by the difference between its prior probability and its self transition probability. An algorithm iteratively found breakpoints that maximized the persistence measure.

Bagnall and Janacek proposed to use clipping to discretize date [31]. In clipping, a data point was assigned 1 if its value was greater than the mean; and 0 otherwise. Therefore, clipping would convert the raw date into a binary series for faster processing and easy storage. For time series generated by autoregressive moving average models, clustering of clipped data was as good as that of original data, and was better when there were outliers.

Keogh and Pazzani introduced a representation of time series based on piecewise linear segmentation [14]. Each linear segment was represented by its end points and its weight, which was the relative significance of the segment. They found the representation suitable for clustering and classification. Another representation based on piecewise linear segmentation was introduced in [32] for feature reduction. The representation allowed variable size segments and each segment was by its mean. Two distance measures were proposed as well as an index structure.

In [15], Cole et al. dealt with the problem of feature reduction when energy was spread across spectrum and thus could not by approximated by a few coefficients in other feature reduction methods such as Fourier transformation, wavelet transformation, and principal component analysis. They proposed a method called sketch. Subsequences of a sequence were transformed into a vector in sketch space by taking inner product with a set of random vectors. To deal with the dimensionality of the resulting vector, it was split into small subsets for indexing and searching.

Mielikainen et al. studied the problem of merging various segmentations of a time series [16]. Given a set of segmentations over the same time series, they tried to find the best segmentation, in the sense that it agreed the most with the given segmentations. An algorithm was proposed which used dynamic programming to find the best segmentation. A greedy algorithm was also proposed which was faster and in practice found near optimal solution.

Patel et al. proposed to find unknown patterns in time series data [33]. Instead of looking for patterns in a time series database, they tried to identify frequent patterns in the database, which they called *motif.* The time series was first divided into a set of segments. Each segment was reduced to its mean. The segments were then normalized and discretized using equal area under a Gaussian distribution. An algorithm was proposed which searched motifs in the discretized data.

A probabilistic model for subsequence matching was proposed by Keogh and Smyth [34]. After piecewise linear segmentation of sequences, features were extracted and used to model the sequences. A similarity measure for comparing two sequences was proposed which was composed of two parts, distance and deformation. Both were modeled with a known distribution. In [35], Ge and Smyth also used the piecewise linear segmentation. However, they proposed a Markov model for sequences. Each segment was a state in the Markov model. The parameters of the Markov model were learned from sequences in database. The query sequence was checked against the model to find the likelihood that it was generated from the model. If so, a matching was found.

Most time series data mining research used Euclidean distance to measure similarity between two subsequences. However, Euclidean distance is sensitive to noise and temporal variation. Dynamic Time Warp (DTW) was introduced in [36] to overcome the shortcomings of the Euclidean distance. DTW allows mapping of a point in one sequence to one or more points in the other sequence. Using dynamic programming, the DTW similarity between two sequences could be calculated efficiently. Keogh and Pazzani [37] introduced an approximate representation of sequences to speed up DTW calculation for long sequences. In their method, a sequence was split into segments, each of which was represented by its mean, thus reducing the size of the sequence.

It is clear that most research in time series data mining does not address privacy issues, let alone time series privacy issues. While current privacy preserving techniques can be applied to preserve snap-shot privacy in time series data, they are inadequate for protecting time series privacy.

3 Time Series Privacy Issues

we identify privacy issues for time series data in addition to traditional privacy issues in data mining. Time series data from a data source can be regarded as a time-domain signal. All the characteristics of a time-domain signal can be potentially regarded as private information by the data provider. Below, we list common characteristics in time series data that a data provider may need to keep confidential.

- Amplitude: Amplitude indicates the strength of a signal. This is the same as in traditional privacy research.
- Average: The average signal strength over time. For example, for a series of sales data, average amplitude indicates the average sales.
- Peak and trough: Peak and trough indicate extreme situations. The information is usually considered confidential as it may disclose extreme changes in underlying causes such as difficulties in money flow.
- Trend: By observing trends of time series data, an adversary may predict future changes of time series data. Thus trend information should be protected from competitors as well.
- Periodicity: Periodical changes in time series data indicate existence of periodically changing factors. For sales data of a store, the factor can be periodical changes in marketing strategies such as promotion which are usually regarded as confidential information for stores. Unlike the previous characteristics which are in time domain, periodicity is in frequency domain.

There are other characteristics which may be regarded as confidential by some data providers. However, as an initial study on time series privacy, we focus on the common characteristics listed above. Since data flow separation attack aims to recover original signal, the attack may be effective to disclose these common characteristics.

4 Threat Model

In this paper we assume that data providers care about the sensitive information contained in their time series data. To protect their privacy, data providers will only supply their data to trusted research companies. Research companies will aggregate time series data provided by different data providers according to different criteria. An example of aggregating sales data provided by auto manufacturers is shown in Figure 1. In Figure 1, there is only one aggregation layer. In practice there can be many layers of aggregation because some research companies may aggregate data provided by other research companies or aggregate data provided by both original data providers and research companies.

We assume research companies will publish aggregated data for profit or for public usage. The research companies will disclose criteria used in aggregation. But research companies will not disclose information of data sources, specifically identities of data providers to protect privacy of data providers.

We assume adversaries to have capability summarized as follows:

- Adversaries can obtain aggregated data from research companies for a small amount of fee or for free.
- Adversaries can not obtain data generated from original data sources because of lack of trust with original data sources. This assumption excludes the possibility of an original data provider being a privacy attacker. We do not study the case of compromised data provider in this paper. But obviously the data flow separation attack will be more effective if an adversary, being a provider of original data, can know part of original data aggregated by research companies.
- Adversaries can obtain data aggregated according to different criteria.
- Research companies have various data providers as their data sources and research companies do not want to disclose the composition of data sources. It is similar as risk company does not want to disclose the composition of stocks in possess.

The model assumed in our paper is realistic. Many research company compiles weekly or monthly sales of large items, such as cars, TVs, computers, etc, from retailers or manufacturers. Each research company has its own sources and publishes its reports with aggregated totals. Since these reports are available with a small fee, someone can collect all these reports and try to separate data to recover original data. The manufacturers do not want to share data with parties other than trusted research companies, but would like to see the aggregated data to understand industry.

5 Data Flow Separation Attack

In this section, we will first define the problem in the context of blind source separation and then describe how to apply the data flow separation attack in

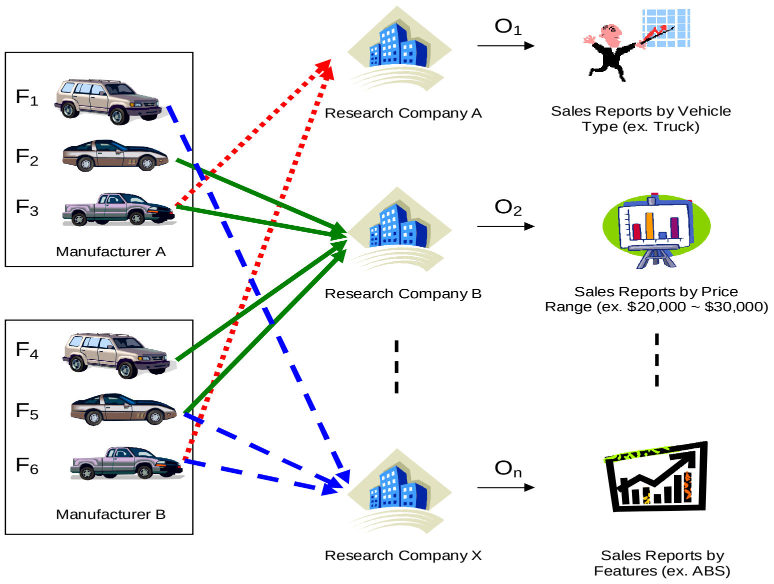

F1
F2
F3
Manufacturer A
F4
F5
F6
Manufacturer B
Research Company A
Research Company B
Research Company X
O1
O2
On
Sales Reports by Vehicle Type (ex. Truck)
Sales Reports by Price Range (ex. $20,000 ~ $30,000)
Sales Reports by Features (ex. ABS)

Figure 1: An Example for Data Flow Model

practice.

5.1 Blind Source Separation

Blind source separation is a methodology in statistical signal processing to recover unobserved "source" signals from a set of observed mixtures of the signals. The separation is called "blind" to emphasize that the source signals are not observed and that the mixture is a black box to the observer. While no knowledge is available about the mixture, in many cases it can be safely assumed that source signals are independent. In its simplest form [38], the blind source separation model assumes n independent signals $F_1(t), \cdots, F_n(t)$ and n observations of mixture $O_1(t), \cdots, O_n(t)$ where $O_i(t) = \sum_{j=1}^{n} a_{ij} F_j(t)$. The goal of blind source separation is to reconstruct the source signals $F_j(t)$ using only the observed data $O_i(t)$, with the assumption of independence among the signals $F_j(t)$. A very nice introduction to the statistical principles behind blind source separation is given in [38]. The common methods employed in blind source separation are minimization of mutual information [39, 40], maximization of nongaussianity [41, 42] and maximization of likelihood [43, 44].

5.2 Data Flow Separation as a Blind Source Separation Problem

In this paper, we define an *individual data flow* as a series of time-stamped data generated by an original data source. *An aggregate data flow* is defined as the aggregate of individual data flows. Aggregate data flows are generated by research companies. If not specified, the phase *data flow* in the remaining of this paper means the individual data flow for brevity.

For the attacker who is interested in sensitive information contained in individual data flow, it will be very helpful to separate the individual data flows based on the aggregate data flows. Because the separation of the data flows can recover the pattern of data flows, they can be use for further attack such as frequency spectrum matching attack described in Section 5.3.

In this paper, we are interested in patterns carried in the time series data. For example, in Figure 1, the attacker can get a time series $O_1 = [o_1^1, o_2^1, \cdots, o_n^1]$ of aggregate data flow from Research Company A. We call n as the sample size in this paper. The attacker's objective is to recover the time series $F_i = [f_1^i, f_2^i, \cdots, f_n^i]$ for each individual data flow. The time series F_3 is contained in both time series O_1 and O_2, i.e., $O_1 = F_3 + F_6$, $O_2 = F_2 + F_3 + F_4 + F_5$. For the scenario with l research companies and m individual data flows, we can rewrite the problem in vector-matrix notation,

$$\begin{pmatrix} O_1 \\ O_2 \\ \vdots \\ O_l \end{pmatrix} = \mathbf{A}_{l\times m} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{pmatrix} \tag{1}$$

where $\mathbf{A_{l\times m}}$ is called mixing matrix in blind source separation problem.

Data flow separation can be achieved using blind source separation techniques. The individual data flows are independent from each other since the individual data flows are from different sources. Given the observations $O_1, O_2, \cdots, O_l$, blind source separation techniques can be used to estimate the independent aggregate flows $F_1, F_2, \cdots, F_m$ by maximizing the independence between estimated individual data flows.

The issues about blind source separation method are summarized as follows.

- Basic blind source separation algorithms requires the number of observations to be greater than or equal to the number of independent components. For data flow separation, it means $l \geq m$. Advanced blind source separation algorithms [45, 46] can be use for cases where $m > l$. But they usually require that m, the number of independent flows, to be known. Since it is hard for the attacker to get m, we assume $l \geq m$. The cost of the assumption is that some independent flows are still mixed when $m > l$. But by using frequency spectrum matching method, the shortcoming can be overlooked. Furthermore, we assume $m = l$ in this paper since it is fairly straightforward to extend our idea to cases where $l > m$.

- The l observations may have redundancy. In other words, the row vectors of the mixing matrix may be linearly dependent. Again, the cost of the redundancy will be that some independent data flows are not separated.

- The data flow estimation by blind source separation algorithms are usually a lifted, scaled version of the actual data flow. Sometimes, the estimated data flow may be of different sign than the actual data flow. This is because the mixing matrix got from the blind source separation can contain arbitrary numbers[1]. However, the attacker can still find characteristics of the actual data flow from the estimated data flow. Also, heuristic approaches can be used to fine tune the estimation, which is an interesting topic for further research.

5.3 Frequency Matching Attack

After the data flows have been separated, a number of data flows, each with a given time series, has been determined to be included in the aggregate.

We choose the frequency spectrum matching to do further attack. Frequency spectrum can be generated by applying discrete Fourier transform on time series data and then calculating the magnitude of transformed data. We match frequency spectrum by correlation.

The rationale for the use of frequency matching is two-fold: First, the dynamics of many data flows, such as sales, stock price, and weather, is characterized by their periodicities. By matching the frequency spectrum of a known data flow with the frequency spectrum of estimated data flows obtained by blind source separation techniques, we can identify the known flow with high accuracy. Second, frequency matching can easily remove the ambiguities introduced by the lifting and scaling in the estimated time series by removing the zero-frequency component. In summary, frequency spectrum analysis has excellent prerequisites to be highly effective.

Frequency spectrum matching can be applied to match data flows separated from different attacks. After collecting a set of aggregate data flows according to different criteria, the attacker can select arbitrary subsets of aggregate data flows as groups and apply data flow separation techniques on the groups to recover individual data flows. If a data flow separated from a group matches with a data flow separated from another group, then these two data flows should be generated from the same source. Moreover, the source generating these two data flows should satisfy at least one aggregation criteria in each group. If the attacker can match a data flow with data flows separated from several groups, the attacker can largely reduce the anonymity or possibly determine the identify of the source generating the data flow since the source should satisfy at least one criteria in each of these groups of aggregate flows. To better utilize the data, the attacker can try all the possible combinations to group available aggregate data flows and then match the data flows separated from these groups. Of course, when the number of aggregate flows in a group is too small, the data flow separation technique can not separate all data flows because the number of observations is smaller than the number independent components. We will study the limit of this grouping and matching method in our future research.

[1]For some cases, the mixing matrix is known to be binary. This kind of *a priori* information can eliminate the ambiguities.

6 Evaluation

In this section, we will evaluate the performance of the data flow separation. We use the blind source separation algorithm proposed in [47] to separate the data flows. The accuracy of separation will be measured using correlation with actual flows. In our experiments, real stock market data [48] is used.

6.1 Performance Metrics

In the following, we will adopt two metrics to evaluate the accuracy of the data flow separation. Both metrics are based on a comparison of the separated data flows with the actual data flows.

As first performance metric, we use *mean square error (MSE)*, a widely used performance criterion in blind source separation research. Let $F_A = [f_1^A, f_2^A, \cdots, f_n^A]$ represent the time series of the actual data flow and $F_B = [f_1^B, f_2^B, \cdots, f_n^B]$ represent the time series estimated by the blind source separation algorithm. To match the time series F_A with F_B, we first need to scale and lift F_B so that they have the same mean and variance.

$$F'_B = \frac{std(F_A)}{std(F_B)} \cdot (F_B - mean(F_B) \cdot [1, 1, \cdots, 1]) + mean(F_A) \cdot [1, 1, \cdots, 1] \ , \quad (2)$$

where $std(F)$ and $mean(F)$ denote the standard deviation and average of the time series F, respectively. The *mean square error* is defined as follows:

$$\varepsilon_{A,B} = \frac{\|F_A - F'_B\|^2}{n} \quad . \quad (3)$$

Since the times series F_B can also be a flipped version of F_A, we also need to match F_A with $-F_B$.

As the second metric, we use *correlation* between a separated flow F_B and the corresponding actual flow F_A defined as follow:

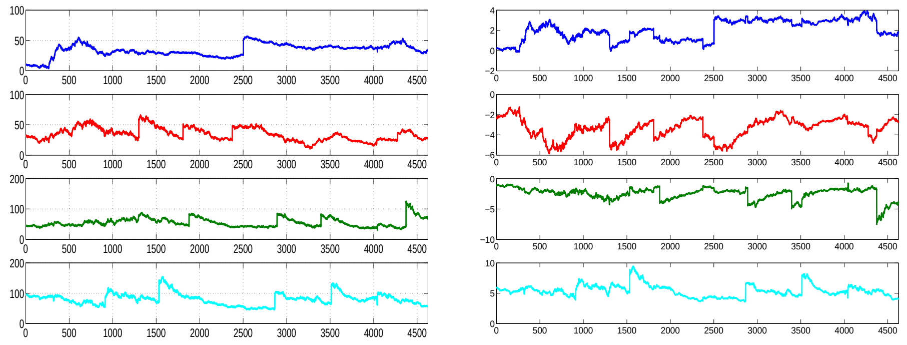
$$R_{F_A, F_B} = \frac{\sum_i (f_i^A - mean(F_A))(f_i^B - mean(F_B))}{std(F_A) std(F_B)} \quad (4)$$

6.2 A Small Example

In this experiment, four time series of stock price selected from [48] are mixed into four aggregates. Figure 2(a) and Figure 2(b) show the actual data flows and separated data flows from the aggregates.

We can observe for data flow 1, 2, and 3, the separated data flows are flipped, scaled and lifted versions of the corresponding actual data flows. We can also observe the resemblance between separated flow and the corresponding actual flow for data flow 4.

Figure 3 shows the performance of data flow separation in terms of metrics introduced in Section 6.1. According to the correlation metric, the separated data flows are highly correlated to actual data flows as shown in Figure 3(a).



(a) Actual Data Flows **(b) Estimated Data Flows**

Figure 2: Example of Data Flow Separation

In Figure 3(b) both the separated data flow and its flipped time series are compared against the actual flows and the mean square error for each data flow shown in the figure is the smaller one. According to the mean square error metrics, we can observe that the reconstructed data flows are off by around 10% in comparison with the actual data flows. Both metrics indicate that the data flow separation is successful. In the following we will use correlation only to evaluate performance because the lifting and scaling in the mean square error metrics may introduce error.

6.3 Mixing Degree

In this set of experiments, we would like to study the effect of mixing degree on the performance of data flow separation. We define mixing degree as follows:

$$D_{mix} = \frac{\text{average number of individual data flows mixed in aggregates}}{\text{number of individual data flows}} \quad . \tag{5}$$

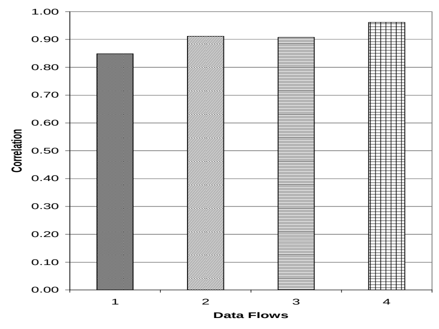
It is equivalent as

$$D_{mix} = \frac{\text{number of non-zero entries in } A_{l\times m}}{l \times m} \quad . \tag{6}$$

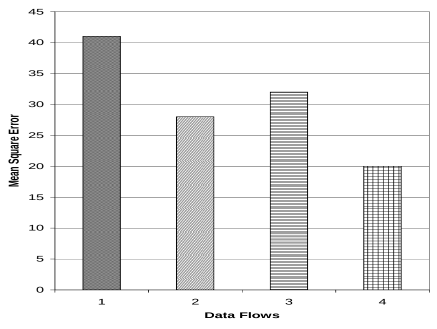
Ten time series selected from stock data [48] are mixed randomly in this experiment to create ten aggregates. Totally 10000 randomly-generated *full-rank*[2] mixing matrices were used in this experiment[3].

[2] Experiments on rank deficient mixing matrices are described in Section 6.4.

[3] In this experiment, we focus on binary mixing matrices since most applications use binary mixing matrices. There are 2^{100} different mixing matrices of size 10×10. We randomly generated 10000 mixing matrices for this experiment.



(a) Correlation



(b) Mean Square Error

Figure 3: Performance of Data Flow Separation on a Small Example

Figure 4 shows the effect of mixing degree on the performance of data flow separation. We plot statistics of both average correlation and worst case correlation. In this paper average correlation is defined as mean of correlation between separated data flows and actual data flows for each trial. We use worst case to refer to the most accurately separated data flow in each trial. It corresponds to worst privacy compromising in each trial.

From Figure 4, we can observe that data flow separation is effective since the separated flows are highly correlated to actual flows especially for the worst case. We can also observe that the performance of data flow separation is not sensitive to mixing degree for full-rank mixing matrices. This experiment indicates that countermeasure to data flow separation attack by simply increasing mix degree is not effective.

6.4 Redundant Aggregate Data Flows

In this set of experiments, we focus on the cases with redundant aggregate data flows. In our setting, redundant aggregate data flows mean that some aggregate data flows are linear combinations of other aggregate data flows. Redundant aggregate data flows will reduce the number of effective aggregate data flows. Redundant aggregate data flows are caused by rank deficient mixing matrices.

To study the effect of redundant observations, we randomly generate 1000 mixing matrices for each possible rank. Ten data flows randomly selected from the stock data are mixed using the randomly-generated mixing metrics of different ranks.

Figure 5 shows the performance of data flow separation with redundant observations. We can observe that the performance of data flow separation decreases as the number of redundant observations increases. The performance

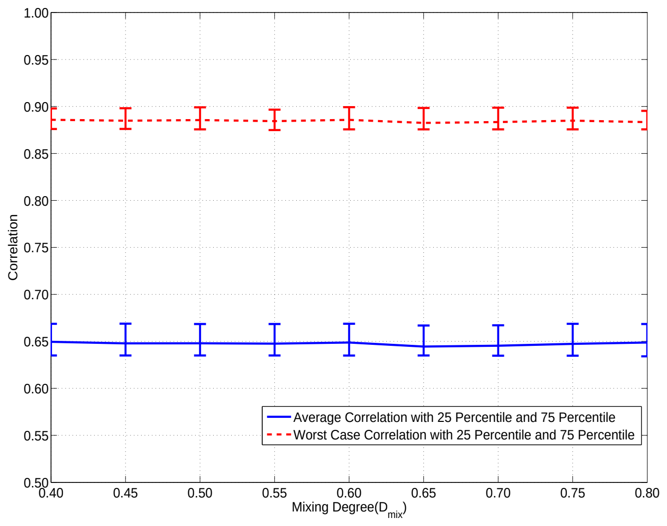


Figure 4: Effect of Mixing Degree (Lower bar: 25 percentile, Upper bar: 75 percentile)

degradation is because the number of knowns decreases. When the number of aggregate data flows is larger than the number of individual data flows, the data flow separation problem becomes an over-complete base problem in blind source separation literature. In general an over-complete base problem is harder to solve.

6.5 Dependence between Individual Data Flows

In this set of experiments, we study the effect of dependence between individual data flows on data flow separation performance. We did this series of experiments because of the fact that most blind source separation algorithms assume relative independence between actual signals.

Groups of ten data flows are randomly picked from the stock data [48]. These groups of data flows have different average correlation between data flows in a same group. The time series in each group are mixed randomly and we apply data flow separation technique on the generated aggregates.

Figure 6 shows that the performance of data flow separation technique decreases when the dependence between individual data flows increases. It is because blind source separation algorithms used in data flow separation assume independence between underlying components. Even for the blind source separation algorithm [47] which takes advantage of both independence and timing structure of underlying signals, the dependence between individual data flows can still degrade the performance of data flow separation attack. We can also

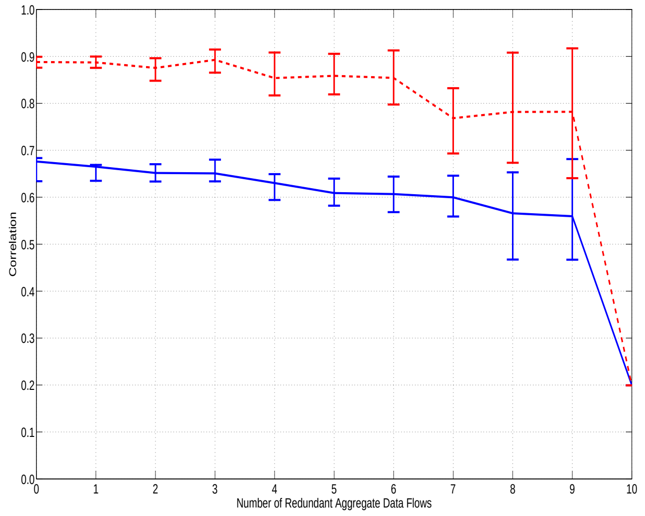


Figure 5: Effect of Dependence between Individual Data Flows (Lower bar: 25 percentile, Upper bar: 75 percentile)

observe that worst case correlation is not sensitive to the dependence between individual data flows.

6.6 Frequency Matching

In this subsection, we show the performance of frequency matching attack proposed in Section 5.3. In this experiment, two groups of ten data flows are formed by selecting data flows from the stock dataset. Three data flows in both groups are the same. These two groups of data flows are mixed randomly to form two groups of aggregate data flows. Data flow separation is performed on the two groups of aggregate data flows. We identify common flows in both groups by matching spectrum of separated data flows in two different groups.

Figure 7 shows the correlation between three identified separated data flows in one group and the ten separated data flows in the other group. As shown in Figure 7 we can easily find out the data flows common to both groups.

7 Discussion

From the experiments in Section 6, it is apparent that aggregation methods are not sufficient to effectively counter data flow separation attacks. Additional measures are needed.

One naive countermeasure can be adding different noise to a data flow to be

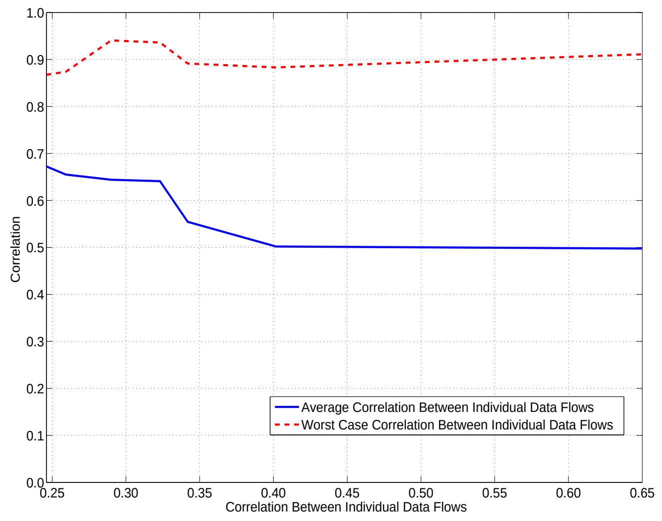


Figure 6: Effect of Dependence between Individual Data Flows

supplied to research companies so that research companies will receive different copies of the data flow. This approach may not work because the noise and the original data flow are regarded as independent components and thus they can be separated by blind source separation algorithm.

According to our experiments, following countermeasures will be effective against data flow separation attacks:

- Increase the dependence among data flows by adding dependant noise to the data flows. Further research is needed to investigate how to optimally add noise so that privacy can be preserved and the performance of time-domain data mining will not be significantly affected.

- Limit the number aggregate data flows that can be obtained by an adversaries so that the number of observations is much less than the number of independent components. This countermeasure requires cooperation among research companies and it is hard to be enforced.

- Data sources should know from research companies about how the supplied data to be aggregated and restricted the usage of supplied data.

Also, research in blind source separation shows most blind source separation algorithms fail when the signals mixed are Gaussian distributed. Therefore, another countermeasure against data flow separation attack is padding each aggregate data flow so that the distribution of the aggregated data is Gaussian. But different classes of blind source separation algorithm that make use of the time structure of the signals can still separate the data flows e.g., [49, 50].

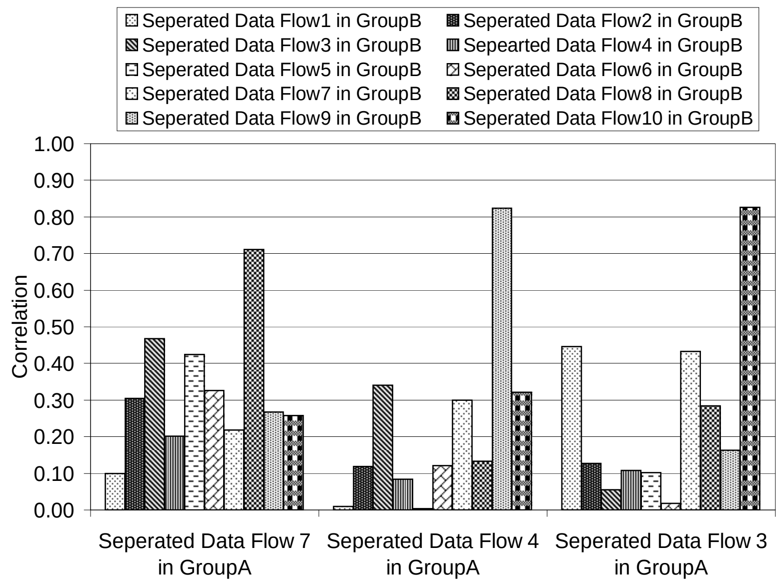


Figure 7: Performance of Frequency Matching

Data flow separation attack can be launched to break privacy of a wide range of privacy-sensitive systems. In this paper, we use auto industry and research company as our example. Our threat model can be easily extended to other systems such as mutual funds. The composition of stocks held by a mutual fund is regarded as commercial secret. But commercial companies publish their performance index routinely. The published performance index is essentially an aggregation of stock prices. An attacker can run data flow separation attack to separate data flows and compare separated data flows with public-known stock prices to estimate the composition of stocks held by mutual fund companies.

As mentioned in [51], aggregation is a major technique used to preserve privacy in data mining. Since data flow separation attack can separate individual data flows from aggregates, aggregation technique based privacy-preserving data mining systems are potentially vulnerable to data flow separation attacks.

8 Conclusion

We presented a new attack against privacy in time series data mining, called data flow separation attack, which can be used either alone or in conjunctions with other attacks to significantly reduce the effectiveness of privacy-preserving techniques in data mining. Data flow separation attack is based on the blind source separation algorithms widely used to recover individual signals from mixtures of signals. Our experiments show that the attack is effective. With the aid of further attack such as frequency spectrum matching attack, data flow separation attack can be used to determine data sources of separate data flows.

We discuss countermeasures against data flow separation attack. Our future work will focus on countermeasures to balance privacy-preserving and performance of data mining. We also plan to analytically model the effectiveness of the attack.

9 Acknowledgement

We thank Professor Keogh for providing us the data sets used in our experiments.

References

[1] Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD Conference. (2000) 439–450

[2] Evfimievski, A.V., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: SIGKDD. (2002) 217–228

[3] Du, W., Zhan, Z.: Using randomized response techniques for privacy-preserving data mining. In: SIGKDD. (2003) 505–510

[4] Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: CRYPTO. (2000) 36–54

[5] Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: SIGKDD. (2002) 639–644

[6] Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans. Knowl. Data Eng. **16**(9) (2004)

[7] Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: FODO. (1993) 69–84

[8] Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD Conference. (1994) 419–429

[9] Das, G., Lin, K.I., Mannila, H., Renganathan, G., Smyth, P.: Rule discovery from time series. In: SIGKDD. (1998) 16–22

[10] Geurts, P.: Pattern extraction for time series classification. In: PKDD. (2001) 115–127

[11] Keogh, E.J., Lin, J.: Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowl. Inf. Syst. **8**(2) (2005) 154–177

[12] Ihler, A.T., Hutchins, J., Smyth, P.: Adaptive event detection with time-varying poisson processes. In: SIGKDD. (2006) 207–216

[13] Mörchen, F., Ultsch, A.: Optimizing time series discretization for knowledge discovery. In: SIGKDD. (2005) 660–665

[14] Keogh, E.J., Pazzani, M.J.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: SIGKDD. (1998) 239–243

[15] Cole, R., Shasha, D., Zhao, X.: Fast window correlations over uncooperative time series. In: SIGKDD. (2005) 743–749

[16] Mielikäinen, T., Terzi, E., Tsaparas, P.: Aggregating time partitions. In: SIGKDD. (2006) 347–356

[17] Jutten, C., Herault, J.: Blind separation of sources, part 1: an adaptive algorithm based on neuromimetic architecture. Signal Process. **24**(1) (1991) 1–10

[18] Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: SIGMOD Conference. (2005) 37–48

[19] Zhu, Y., Liu, L.: Optimal randomization for privacy preserving data mining. In: SIGKDD. (2004) 761–766

[20] Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: a review and open problems. In: New Security Paradigms Workshop 2001, Cloudcroft, New Mexico, USA, September 10-13,. (2001) 13–22

[21] Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1) (1986) 81–106

[22] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB. (1994) 487–499

[23] Cheung, D.W.L., Han, J., Ng, V.T.Y., Fu, A.W.C., Fu, Y.: A fast distributed algorithm for mining association rules. In: PDIS. (1996) 31–42

[24] Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: SIGKDD. (2003) 206–215

[25] Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: SIGKDD. (2005) 593–599

[26] Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS. (1986) 162–167

[27] Wright, R.N., Yang, Z.: Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In: SIGKDD. (2004) 713–718

[28] Cooper, G.F., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning **9** (1992) 309–347

[29] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The r*-tree: An efficient and robust access method for points and rectangles. In: SIGMOD Conference. (1990) 322–331

[30] Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K.: Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In: VLDB. (1995) 490–501

[31] Bagnall, A.J., Janacek, G.J.: Clustering time series from arma models with clipped data. In: SIGKDD. (2004) 49–58

[32] Keogh, E.J., Chakrabarti, K., Mehrotra, S., Pazzani, M.J.: Locally adaptive dimensionality reduction for indexing large time series databases. In: SIGMOD Conference. (2001) 151–162

[33] Patel, P., Keogh, E.J., Lin, J., Lonardi, S.: Mining motifs in massive time series databases. In: ICDM. (2002) 370–377

[34] Keogh, E.J., Smyth, P.: A probabilistic approach to fast pattern matching in time series databases. In: SIGKDD. (1997) 24–30

[35] Ge, X., Smyth, P.: Deformable markov model templates for time-series pattern matching. In: SIGKDD. (2000) 81–90

[36] Berndt, D., Clifford, J.: Using dynamic time warping to find patterns in time series. In: AAAI-94 Workshop on Knowledge Discovery in Databases. (1994)

[37] Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: SIGKDD. (2000) 285–289

[38] Cardoso, J.: Blind signal separation: statistical principles. **9**(10) (1998) 2009–2025 Special issue on blind identification and estimation.

[39] Comon, P.: Independent component analysis, a new concept? Signal Process. **36**(3) (1994) 287–314

[40] He, Z., Yang, L., Liu, J., Lu, Z., He, C., Shi, Y.: Blind source separation using clustering-based multivariate density estimation algorithm. IEEE Trans. on Signal Processing **48**(2) (2000) 575–579

[41] Hyvärinen, A.: Fast and robust fixed-point algorithms for independent component analysis. IEEE Transactions on Neural Networks **10**(3) (1999) 626–634

[42] Hyvärinen, A., Oja, E.: A fast fixed-point algorithm for independent component analysis. Neural Comput. **9**(7) (1997) 1483–1492

[43] Gaeta, M., Lacoume, J.L.: Source separation without prior knowledge: the maximum likelihood solution. In: Proc. EUSIPCO'90. (1990) 621–624

[44] Pham, D.T., Garrat, P., Jutten, C.: Separation of a mixture of independent sources through a maximum likelihood approach. In: Proc. EUSIPCO. (1992) 771–774

[45] Hyvärinen, A., Inki, M.: Estimating overcomplete independent component bases for image windows. J. Math. Imaging Vis. **17**(2) (2002) 139–152

[46] Hyvärinen, A., Cristescu, R., Oja, E.: A fast algorithm for estimating overcomplete ICA bases for image windows. In: Proc. Int. Joint Conf. on Neural Networks, Washington, D.C. (1999) 894–899

[47] Cruces-Alvarez, S.A., Cichocki, A.: Combining blind source extraction with joint approximate diagonalization: Thin algorithms for ICA. In: Proc. of the Fourth Symposium on Independent Component Analysis and Blind Signal Separation, Nara, Japan (apr 2003) 463–468

[48] Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: The ucr time series classification/clustering homepage. `http://www.cs.ucr.edu/~eamonn/time_series_data/` (2006)

[49] Tong, L., Liu, R.W., Soon, V.C., Huang, Y.F.: Indeterminacy and identifiability of blind identification. Circuits and Systems, IEEE Transactions on **38**(5) (1991) 499–509

[50] Molgedey, L., Schuster, H.G.: Separation of a mixture of independent signals using time delayed correlations. Physical Review Letters **72**(23) (June 1994) 3634–3637

[51] Zhang, N., Zhao, W.: Privacy-preserving data mining systems. Computer **40**(4) (2007) 52–58