Hiding in the Crowd: Privacy Preservation on Evolving Streams through Correlation Tracking

Feifei Li[‡] Jimeng Sun[§] Spiros Papadimitriou[†] George A. Mihaila[†] Ioana Stanoi[†]
[‡]Boston University, [§]Carnegie Mellon University, [†]IBM T.J.Watson Research Center
lifeifei@cs.bu.edu, jimeng@cs.cmu.edu, {spapadim, mihaila,irs}@us.ibm.com

Abstract

We address the problem of preserving privacy in streams, which has received surprisingly limited attention. For static data, a well-studied and widely used approach is based on random perturbation of the data values. However, streams pose additional challenges. First, analysis of the data has to be performed incrementally, using limited processing time and buffer space, making batch approaches unsuitable. Second, the characteristics of streams evolve over time. Consequently, approaches based on global analysis of the data are not adequate. We show that it is possible to efficiently and effectively track the correlation and autocorrelation structure of multivariate streams and leverage it to add noise which maximally preserves privacy, in the sense that it is very hard to remove. Our techniques achieve much better results than previous static, global approaches, while requiring limited processing time and memory. We provide both a mathematical analysis and experimental evaluation on real data to validate the correctness, efficiency, and effectiveness of our algorithms.

1. Introduction

Recently, there has been an increasing concern regarding privacy breaches, especially those involving sensitive personal data of individuals [15]. As a result, restrictions and regulations in publishing sensitive personal data have been tightened [38]; these address data owned by government organizations [14] as well as corporations [7]. It is therefore not surprising that the data management community has become increasingly focused on ways to guarantee the privacy of sensitive data.

Meanwhile, unprecedented massive data from various sources are providing us with great opportunity for data mining and information integration. Unfortunately, the privacy requirement and data mining applications pose exactly opposite expectations from data publishing [15, 36]. The utility of the published data w.r.t the mining application decreases with increasing levels of privacy guarantees [24]. Previous work has noticed this important tradeoff between privacy and utility and various techniques have been proposed to achieve a desired balance between the two [3, 23, 27, 10, 12, 28, 35, 16].

Prior related work [3, 2, 23, 20] consists of additive random perturbation for the offline, conventional relational data model, where the noise is distributed along the principal components of the original data in order to achieve maximum privacy, given a fixed utility. We show that these offline algorithms are no longer optimal when applied to numerical, non-stationary (or, time-evolving) data streams. The dynamic correlations and autocorrelations, if not carefully considered, may allow for the reconstruction of the original streams.

Guaranteeing data privacy is especially challenging in the case of stream data [6, 29], mainly for two reasons:

1) **Performance requirement:** The continuous arrival of new tuples prohibits storage of the entire stream for analysis, rendering the current offline algorithms inapplicable.

2) **Time evolution:** Data streams are usually evolving, and correlations and autocorrelations [33, 25] can change over time. These characteristics make most offline algorithms for static data inappropriate, as we show later.

To the best of our knowledge, privacy preservation on streams has not yet been addressed in the literature, despite the wide use of data streams in a large range of sensitive applications such as financial, retail, defense, and health care. For example, consider two financial firms that would like to collaboratively monitor clusters over their streaming real-time transactions [39]. However, none of them is willing to publish the original data streams. The best resolution is to find ways to guarantee both the utility and privacy of data in an online fashion. The scheme should general and not restricted to any specific mining operation.

In this paper we fill the gap in the area of data stream privacy, by proposing efficient online streaming algorithms that guarantee the privacy of single or multiple non-stationary data streams. This work focuses on numerical data streams, such as environmental sensor data, performance measurements, or stock trading prices. Our goal is to insert random perturbation that "mirrors" the streams' statistical properties, in an online fashion. A number of impor-

Symbol	Description
$\mathbf{v}$	a vector (lowercase bold)
$\mathbf{v}(i)$	the i-th element of vector $\mathbf{v}$
$\mathbf{X}$	a matrix (uppercase bold)
$\mathbf{X}^\mathsf{T}$	the transpose of $\mathbf{X}$
$\mathbf{X}_i$ or $\mathbf{X}^j$	i-th row or j-th column of $\mathbf{X}$
$\mathbf{X}_i^j$	the entry (i, j) of $\mathbf{X}$
T	the number of timestamps up to now
N	the number of streams
$\mathbf{A}$	original stream collection in $\mathbb{R}^{T\times N}$
$\mathbf{A}^*$	the perturbed stream collection
$\tilde{\mathbf{A}}$	the reconstructed stream collection
$\mathbf{A}^n$	the n-th stream
$\mathbf{A}_t$	the values from all streams at time t
$\mathbf{E}$	the random noise in $\mathbb{R}^{T\times N}$
$\mathcal{D}(\mathbf{A}, \mathbf{A}^*)$	the discrepancy on original and perturbed streams

Table 1. Description of notation.

tant mining operations can still be performed, by controlling perturbation magnitude. However, the original data streams cannot be reconstructed with high confidence.

To the best of our knowledge, our work is the first to provide the basic building blocks sufficient for a general solution for privacy of numerical streams. More specifically, we focus on the fundamental cases of correlation across multiple streams and of autocorrelation within one stream.

Our contributions are: 1) define the notion of utility and privacy for perturbed data streams, 2) explore the effect of evolving correlations and autocorrelation in data streams, and their implications in designing additive random perturbation techniques, 3) design efficient online algorithms under the additive random perturbation framework, which maximally preserve the privacy of data streams given a fixed utility while, additionally, better preserving the statistical properties of the data, and 4) provide both theoretical arguments and experimental evaluation to validate our ideas.

The rest of the paper is organized as follows: Section 2 introduces definitions and problem formulation and Section 3 discusses the related work. Section 4 studies privacy preservation for multiple streams through correlation tracking. Section 5 further exploits the autocorrelation property to preserve privacy. Finally, the experimental evaluation on real data streams is performed in Section 6.

2. Preliminaries

2.1 Data Stream Model

Our model assumes that the input consists of multiple continuous streams. Without loss of generality, we may assume that each tuple consists of a single attribute. Furthermore, we assume that all streams are resampled to a common rate, which is between the arrival rate of the fastest and the slowest stream. The common sampling rate can be chosen based on arrival rate, data characteristics and available processing capacity—details are beyond the scope of this paper. Subsequently, any standard resampling technique [30, 19] can be applied such as, for example, linear interpolation (for upsampling) or decimated moving average (aka. tumbling average, for downsampling). We will thus assume a time granularity such that, during each time interval, there is exactly one recorded incoming value from each stream.

Therefore, for the purposes of our analysis and without loss of generality, the input consist of N data streams, denoted as $\mathbf{A}^1, \ldots, \mathbf{A}^N$. For any i-th data stream $\mathbf{A}^i$, its value at time t is $\mathbf{A}_t^i$. The stream collection is written as $\mathbf{A} = [\mathbf{A}^i$ for $1 \le i \le N]$. Formally, the stream collection $\mathbf{A}$ can be considered as a $T \times N$ matrix where N is the number of streams and T is the current timestamp, which grows indefinitely. The values from all streams at time t are $\mathbf{A}_t \in \mathbb{R}^N$, i.e., t-th row of $\mathbf{A}$.

2.2 Discrepancy, Utility and Privacy

To ensure privacy of streaming data, the values of incoming tuples are modified by adding noise. We denote the random noise as $\mathbf{E} \in \mathbb{R}^{T\times N}$ where each entry $\mathbf{E}_t^i$ is the noise added to the i-th stream at time t. Therefore, the perturbed streams are $\mathbf{A}^* = \mathbf{A} + \mathbf{E}$. Without loss of generality, we assume the noise has zero mean.

Discrepancy: To facilitate the discussion on utility and privacy, we define the concept of *discrepancy* $\mathcal{D}$ between two versions of the data, $\mathbf{A}$ and $\mathbf{B}$, as the normalized squared Frobenius norm[1],

$$\mathcal{D}(\mathbf{A}, \mathbf{B}) := \frac{1}{T}\|\mathbf{A} - \mathbf{B}\|_F^2, \quad \text{where } \mathbf{A}, \mathbf{B} \in \mathbb{R}^{T\times N}.$$

Utlity: Considering the perturbed versus the original data, the larger the amplitude of the perturbation (i.e., the variance of the added noise), the larger the distortion of the original values. However, as the distortion increases, the usefulness of the data decreases: a larger distortion hides the original values better but it also hides more information about their relationships. The discrepancy $\mathcal{D}(\mathbf{A}, \mathbf{A}^*)$ between original and perturbed data measures precisely the squared distortion. We naturally define the utility to be the inverse of this discrepancy. However, throughout the paper, we typically use discrepancy, since the two are essentially interchangeable.

Privacy: Distorting the original values is only part of the story. We also have to make sure that this distortion cannot be filtered out. Thus, to measure the privacy, we have to consider the power of an adversary in reconstructing the original data. Specifically, suppose that $\tilde{\mathbf{A}}$ are the reconstructed data streams obtained by the adversary, in a way that will be formalized shortly. Then the privacy is the discrepancy between the original and the reconstructed streams, i.e., $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}})$.

[1] The squared Frobenius norm is defined as $\|\mathbf{A}\|_F^2 := \sum_{i,j}(A_i^j)^2$

2.3 Problem Formulation

We formulate two problems: data reconstruction and data perturbation. From his side, the adversary wants to recover the original streams from the perturbed data.

Problem 1 (Reconstruction)**.** *Given the perturbed streams* $\mathbf{A}^*$*, how to compute the reconstruction streams* $\tilde{\mathbf{A}}$ *such that* $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}})$ *is minimized?*

In this paper, we focus on linear reconstruction methods which have been used by many existing works [23, 20, 27, 10]. Intuitively, the adversary can only use linear transformations on the perturbed data, such as projections and rotations, in the reconstruction step.

Definition 1 (Linear reconstruction)**.** *Given the perturbed streams* $\mathbf{A}^*$*, the linear reconstruction is* $\tilde{\mathbf{A}} = \mathbf{A}^*\mathbf{R}$*, such that* $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}})$ *is minimized*

If both the perturbed streams $\mathbf{A}^*$ and the original streams $\mathbf{A}$ are available, the solution $\tilde{\mathbf{A}}$ can be easily identified using linear regression. However, $\mathbf{A}$ is not available. Therefore, in order to estimate $\tilde{\mathbf{A}}$, some additional constraints or assumptions must be imposed to make the problem solvable. A widely adopted assumption [22] is that the data lie in a static low dimensional subspace (i.e, global correlation exists). This is reasonable, since if no correlations are present, then i.i.d. perturbations are already sufficient to effectively hide the data. However, real data typically exhibit such correlations. In this paper, as we will formally show later, we rely on the dynamic (rather than static) correlations among streams, as well as on dynamic autocorrelations.

From their side, data owners want to prevent the reconstruction from happening.

Problem 2 (Perturbation)**.** *Given the original streams* $\mathbf{A}$ *and the desirable discrepancy threshold* σ^2*, how to obtain the perturbed streams* $\mathbf{A}^*$ *such that 1)* $\mathcal{D}(\mathbf{A}, \mathbf{A}^*) = \sigma^2$ *and 2) for any linear reconstruction* $\tilde{\mathbf{A}}$*,* $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}}) \geq \sigma^2$*.*

Perturbation has exactly the opposite goal from the *reconstruction*. However, the correlation and autocorrelation properties of the streams are still the keys in the solution of both problems, as shown later.

3. Related Work

Privacy preserving data mining was first proposed in [3] and [2]. This work paved the road for an expanding field, and various privacy preservation techniques have been proposed since. These methods apply to the traditional relational data model, and can be classified as data perturbation [3, 2, 27, 10, 16, 4], k-anonymity [24, 35, 1, 28, 38] and secure multiparty computation [26, 37]. Our work focuses on privacy preservation in the context of the randomized data perturbation approach and we will focus on discussing related work in this area.

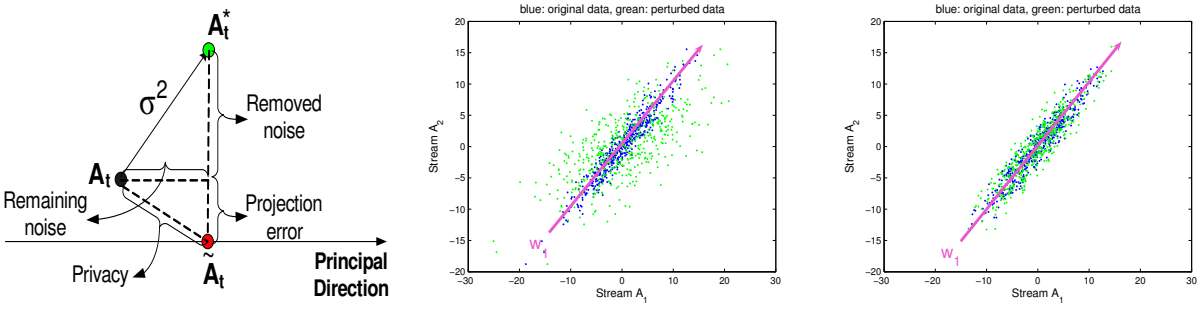
Data perturbation can be further classified in two groups: retention replacement perturbation [4, 16] and data value perturbation [3, 27, 10]. For each element in a column j, the retention replacement perturbation retains this element with probability p_j and with probability $1 - p_j$ replaces it with an item generated from the replacing p.d.f. on this column. This approach works for categorical data as well, and it has been applied to privacy preserving association mining [16]. Our work focuses on numerical data value perturbation. Initial solutions in this category, [3, 2], proposed adding random i.i.d. noise to the original data and showed that, with knowledge of the noise distribution, the distribution of the original data can be estimated from the perturbed data, and aggregate values are preserved. In [23, 20] the authors pointed out that adding random i.i.d. noise is not optimal for privacy preservation. They showed how to reconstruct the original data (individual data values) using Spectral Filtering (SF) or the equivalent PCA method. The main conclusion is that random noise should be distributed along the principal components of the original data, so that linear reconstruction methods cannot separate the noise from the original data. Motivated by this observation and in similar spirit, [10] proposed the random rotation technique for privacy preserving classification and [27] proposed data perturbation based on random projection. The work of [15] discussed a method to quantify the privacy breach for privacy preserving algorithms, namely $\alpha - \beta$ analysis or γ-amplification. The basic idea is that, on the perturbed data, the adversaries' knowledge measured by their confidence about a given property of the original data should not be increased more than a certain amount. The work in [5] considered the problem of setting the perturbation parameters while maintaining γ-amplification.

All these techniques have been developed for the traditional relational data model. There is no prior work on privacy preservation on data streams, except the work on private search over data streams [31, 8]. However, the goal there is to protect the privacy of the query over data stream, not of the data stream itself. Finally, our data perturbation techniques rely on PCA for data streams w.r.t both correlations and autocorrelations. Streaming PCA and eigenspace tracking of correlations (but not autocorrelation) among multiple data streams has been studied in [32, 18].

4. Privacy with Dynamic Correlations

We first give the insight behind our data perturbation and reconstruction methods, then present methods for correlation-based noise perturbation and reconstruction.

Insight and intuition Let us first illustrate how the perturbation and reconstruction work in detail (see figure 1(a)



(a) noise decomposition (b) i.i.d random noise (c) correlated noise

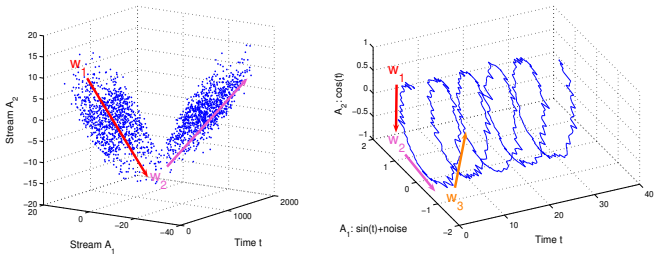
Figure 1. Impact of Correlation on Perturbing the Data

for the visualization). For the perturbation process, the stream measurements $\mathbf{A}_t$ at time t, represented as an N-dimensional point, are mapped to the perturbed measurements $\mathbf{A}_t^*$ with discrepancy σ^2. For any reconstruction effort, the goal is to transform the perturbed measurements, $\mathbf{A}_t^*$ onto $\tilde{\mathbf{A}}_t$ so that, hopefully, $\mathcal{D}(\mathbf{A}_t, \tilde{\mathbf{A}}_t)$ is small.

A principled way of reconstruction is to project the data onto the principal component subspace [20] such that most noise is removed, while the original data are maximally preserved, i.e, not much additional error is included. The idea is illustrated in figure 1(a). When $\mathbf{A}_t^*$ is projected onto the principal direction, the projection is exactly the reconstruction $\tilde{\mathbf{A}}_t$. Note that the distance between $\mathbf{A}_t^*$ and $\tilde{\mathbf{A}}_t$ consists of two parts: 1) *removed noise*, i.e., the perturbation that is removed by the reconstruction and 2) *projection error*, i.e, the new error introduced by the reconstruction. Finally, the distance between $\mathbf{A}_t$ and $\tilde{\mathbf{A}}_t$, i.e., the privacy, comes from two sides: 1) *remaining noise*, i.e, the perturbation noise that has not been removed, and 2) *projection error*.

When the noise is added exactly along the principal direction, the *removed noise* becomes zero. However, additional *projection error* is included. In this case, the perturbation is robust towards this reconstruction attempt, in the sense that $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}}) = \mathcal{D}(\mathbf{A}, \mathbf{A}^*)$. In general, a good practice is to add correlated noise following the trends present in the streams. Consider the example shown in Figure 1 where blue points represent the original data and green points represent the perturbed data with same amount of noise. Figure 1(b) and 1(c) show the i.i.d. noise and the correlated noise on the same data, respectively. Clearly, correlated noise has been successfully "hidden" in the original data and, therefore, is hard to remove.

Data streams often present strong correlations and these correlations change dynamically [39, 33, 32]. Consider the examples in figure 2, where the principal components are changing over time. In such case, online PCA is necessary to better characterize the evolving, local trends. Global, offline PCA will fail to identify these important properties as we will show later in the experiments. Next, we will show how to dynamically insert noise using online correlation tracking [32].



(a) Example one. (b) Example two.

Figure 2. Dynamic correlations in Data Streams

Algorithm 1: SCAN

Input : Original tuple $\mathbf{A}_t$, utility threshold σ^2
Old subspace $\mathbf{U} \in \mathbb{R}^{N\times k}$, $\mathbf{\Lambda} \in \mathbb{R}^{k\times k}$
Output: Perturbed tuple $\mathbf{A}_t^*$, new subspace $\mathbf{U}, \mathbf{\Lambda}$
update eigenvector $\mathbf{U}$, eigenvalue Λ based on $\mathbf{A}_t$)
Initialize δ, η to $\vec{0}_k$
//add noise in top-k principal component subspace
for $1 \leq i \leq k$ **do**
  δ(i)=$\sigma^2 \times \frac{\Lambda(i)}{\|(\Lambda)\|}$
  $\eta(i)$ = gaussian noise with variance δ(i)
// rotation back to the original space
$\mathbf{E}_t$=$\eta \times \mathbf{U}^\mathsf{T}$ and $\mathbf{A}_t^*$=$\mathbf{A}_t$+$\mathbf{E}_t$

Streaming Correlated Additive Noise (SCAN) SCAN does two things whenever new tuples arrive from the N input streams: 1) update the estimation of local principal components; and 2) distribute noise along the principal components in proportional to their eigenvalues.

An important property of the SCAN algorithm is that when the noise is rotated back to the data space (line 6), its variance will be equal to the specified discrepancy threshold σ^2. Intuitively, SCAN tracks the covariance matrix and adds noise with essentially the same covariance as the data streams—proof details are ommited for space.

Theorem 1. *At any time instant T, the perturbed data*

Algorithm 2: SCOR

Input : Perturbed tuple $\mathbf{A}^*_t$, utility threshold σ^2
Old subspace $\mathbf{U} \in \mathbb{R}^{N\times k}$, $\mathbf{\Lambda} \in \mathbb{R}^{k\times k}$
Output: Perturbed tuple $\tilde{\mathbf{A}}_t$, new subspace $\mathbf{U}$, $\mathbf{\Lambda}$
update eigenvector $\mathbf{U}$, eigenvalue Λ based on $\mathbf{A}_t$)
//project to the estimated online principal components
$\widehat{\mathbf{A}}_t$=$\mathbf{A}^*_t \times \mathbf{U}_{N\times k} \times \mathbf{U}^{\mathsf{T}}_{N\times k}$

streams $\mathbf{A}^*$ *from SCAN satisfy* $\mathcal{D}(\mathbf{A}, \mathbf{A}^*) = \sigma^2$. *Additionally, SCAN preserves the eigenvectors of the (uncentered) covariance matrix of* $\mathbf{A}$.

Therefore, the SCAN perturbation will not affect any mining algorithms that rely on the second moments of the data (i.e., linear correlations).

Streaming correlation online reconstruction (SCOR): The privacy achieved by SCAN is determined by the best linear reconstruction an adversary could perform on $\mathbf{A}^*$ (see Section 2.2). For evolving data streams as illustrated in figure 2, the best choice for the adversary is to utilize online estimation of local principal components for reconstruction. The ability to estimating the local principal components of the *original* data streams depends on how the noise has been added. For SCAN, we know that the principal component directions are preserved, since the noise is added along their direction (Theorem 1). In general, we may assume the noise is small compared to the data—otherwise, the utility of the perturbed data is too low to be useful. Then, tracking the principal components of the perturbed streams $\mathbf{A}^*$ can give a good estimate of the principal components of the original streams $\mathbf{A}$. Formally, $cov(\mathbf{A}^*) \approx cov(\mathbf{A})$.

Intuitively, SCOR reconstruction removes all the noise orthogonal to the local principal components and inserts little additional projection error, since local PCA can usually track the data accurately. In other words, i.i.d. noise can usually be successfully removed, provided that the streams are correlated. However, the perturbation from $SCAN$ cannot be removed at all since the the noise is distributed along the "instantaneous" correlation in the streams.

Theorem 2. *The reconstruction error of* $SCOR$ *on the perturbation from* $SCAN$ *is* $\approx \sigma^2$.

Proof. Formally, given a linear reconstruction $\tilde{\mathbf{A}} = \mathbf{A}^*\mathbf{R}$, the privacy can be decomposed as

$$\begin{aligned}\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}}) &= \|\mathbf{A} - \mathbf{A}^*\mathbf{R}\|_F^2 \\ &= \|\mathbf{A} - (\mathbf{A} + \mathbf{E})\mathbf{R}\|_F^2 \\ &= \|\mathbf{A}(\mathbf{I} - \mathbf{R}) + \mathbf{E}\mathbf{R}\|_F^2. \\ &= \| \underbrace{\mathbf{A}(\mathbf{I} - \mathbf{U}\mathbf{U}^{\mathsf{T}})}_{\text{projection error}} + \underbrace{\mathbf{E}\mathbf{U}\mathbf{U}^{\mathsf{T}}}_{\text{remaining error}} \|_F^2\end{aligned}$$

where $\mathbf{R}$ is a projection matrix, meaning that $\mathbf{R} = \mathbf{U}\mathbf{U}^{\mathsf{T}}$ with $\mathbf{U} \in \mathbb{R}^{N\times k}$ orthonormal. Since the subspaces tracked by both SCOR and SCAN are the same, the remaining noise is σ^2, i.e., no noise is removed. Therefore, $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}}) \geq \sigma^2$ by the triangle inequality. □

Note that the projection error for SCOR is small, provided that the data are locally correlated. Therefore, the reconstruction error (i.e., privacy, as defined in Section 2.2) of SCOR is approximately σ^2, i.e., equal to the original discrepancy. Moreover, when σ^2 is small compared to the original data, other reconstruction methods will result in higher error, due to the large projection error.

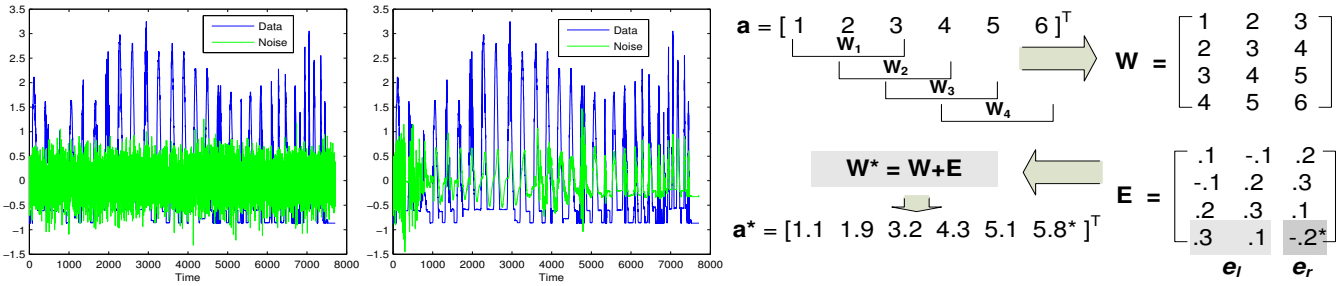
5 Dynamic autocorrelation

So far, we have presented the methods based on correlation across many streams. Now we exploit another important property, autocorrelation on a single stream and then propose the corresponding perturbation and reconstruction.
Intuition: The noise added should mirror the dominant trends in the series. Consider the following simple examples: If the stream always has a constant value, the right way to hide this value is to add the same noise throughout time. Any other noise can be easily filtered out by simple averaging. The situation is similar for a linear trend (this is also an example that cannot be captured by Fourier). If the stream is a sine wave, the right way to hide it is by adding noise with the same frequency (but potentially a different phase); anything else can be filtered out. Our algorithm is the generalization, in a principled manner, of these notions.

For example, the green and blue curves in figure 3(b) are the autocorrelated noise and the original stream, respectively, where the noise follows the same trends as the streams, over time. In comparison, figure 3(a) shows i.i.d. noise, which can be easily filtered out. The goal is to find a principled way to automatically determine what is the "right" noise, which is "most similar" to the stream.
Connection to correlation: In the previous section, we showed how to track the local statistical properties of the N-dimensional sequence of the vectors $\mathbf{A}_t$, indexed over time t. More specifically, we track the principal subspace of this matrix, thereby focusing on the most dominant (in a least-squares sense) of these relationships. We subsequently add noise that "mirrors" those relationships, making it indistinguishable from the original data.

Next, we will show that the same principles used to capture relationships across many attributes can be used to capture relationships of one attribute across time. In fact, there is a natural way to move between the original time domain and a high-dimensional sequence space, which is formalized next. The t-th *window* of the time series stream $\mathbf{a}(t)$ is an h-dimensional point,

$$\mathbf{W}_t := [\mathbf{a}(t), \mathbf{a}(t+1), \ldots, \mathbf{a}(t+h-1)]^{\mathsf{T}} \in \mathbb{R}^h.$$



(a) iid noise (b) Autocorrelated noise (c) Streaming autocorrelation additive noise

Figure 3. Dynamic Autocorrelation

The *window matrix* $\mathbf{W}$ has the windows $\mathbf{W}_t$ as rows. Thus, $\mathbf{W}_i^j = \mathbf{a}((i-1)h+j)$ by construction. The space spanned by the sequence of windows $\mathbf{W}_t$ is known as the h-th order *phase space* of the series $\mathbf{a}(t)$ [17]. Subsequently, we can essentially apply the same technique as before, using $\mathbf{W}$ in place of $\mathbf{A}$. All of the previous discussion and properties of our algorithm can be directly transferred to the autocorrelation case. An example is shown in the top of figure 3(c). However, there are some additional properties and issues that need to be resolved.

Hankel Constraint: Notice that the window matrix $\mathbf{W}$ is a *Hankel* matrix, i.e., the *anti*-diagonals are constants: $\mathbf{W}_i^j = \mathbf{W}_{i-1}^{j-1}$. Under the assumption that the series is stationary, the autocovariance matrix $\mathbf{W}^T\mathbf{W}$ is, in expectation is *circulant*, i.e., it is symmetric with constant diagonals. Additionally, if we perform a batch eigen-analysis on the global window matrix of a static series, the sample autocovariance matrix computed from the actual data (i.e., $\mathbf{W}^\mathsf{T}\mathbf{W}$ above) is also circulant. In this case, the eigenvectors of $\mathbf{W}^\mathsf{T}\mathbf{W}$ essentially provide the same information as the Fourier coefficients of the series $\mathbf{a}$. In that sense, our approach includes traditional Fourier analysis. If these assumptions do not hold, the technique we employ is more robust and effective. Detailed discussion is beyond the scope of this paper—interested readers may consult [17, 34, 19] for more details.

Constraint on autocorrelated noise: Next, we address the issues that arise from the fact that $\mathbf{W}$ is a Hankel matrix. Similarly, the noise matrix $\mathbf{E}$ has to be a Hankel matrix (see figure 3(c) for an example). Similar to the correspondence between $\mathbf{a}$ and $\mathbf{W}$, the noise matrix $\mathbf{E}$ has a corresponding noise sequence $\mathbf{e}$, such that

$$\mathbf{E}_t \equiv [\mathbf{e}(t), \mathbf{e}(t+1), \ldots, \mathbf{e}(t+h-1)]^\mathsf{T} \in \mathbb{R}^h.$$

We will essentially use the same insight, that $\mathbf{E}_t$ has to lie in the subspace of $\mathbf{U}$, but in a different way. Formally stated, the residual $\mathbf{E}_t - \mathbf{U}\mathbf{U}^\mathsf{T}\mathbf{E}_t$ must be zero, or

$$(\mathbf{I} - \mathbf{U}\mathbf{U}^\mathsf{T})\mathbf{E}_t \equiv \mathbf{Q}\mathbf{E}_t = \mathbf{0}, \qquad (1)$$

where $\mathbf{P} = \mathbf{U}\mathbf{U}^\mathsf{T}$ is the projection operator onto the subspace of $\mathbf{U}$ and $\mathbf{Q} = \mathbf{I} - \mathbf{P} = \mathbf{I} - \mathbf{U}\mathbf{U}^\mathsf{T}$ is the projector onto the orthogonal complement.

Algorithm 3: SACAN

Input : Original value $\mathbf{a}^*(t)$, utility σ^2
Old subspace $\mathbf{U} \in \mathbb{R}^{h\times k}$, $\mathbf{\Lambda} \in \mathbb{R}^{k\times k}$
Output: Perturbed value $\mathbf{a}^*(t)$, new subspace $\mathbf{U}$, $\mathbf{\Lambda}$

1 Construct window $\mathbf{W}_{t-h+1} = [\mathbf{a}(t-h+1), \ldots, \mathbf{a}(t)]^\mathsf{T}$
2 Update $\mathbf{U}$, $\mathbf{V}$ using $\mathbf{W}_{t-h+1}$
3 **every** k arriving values **do**
4 Let $[\mathbf{w}_l^\mathsf{T} \mid \mathbf{w}_r^\mathsf{T}]^\mathsf{T} \equiv \mathbf{W}_{t+h+1}$
5 Solve equation 2 to obtain $\mathbf{e}_r$
6 Rescale $\mathbf{e}_r$ based on σ^2
7 Perturbed values $\mathbf{w}_r^* = \mathbf{w}_r + \mathbf{e}_r$
8 Publish values $\mathbf{a}^*(t-k+i-1) = \mathbf{w}_r^*(i)$, $1 \le i \le k$

Assume that we have chosen the noise values up to time $t-k$. Based on these and on the current estimate of $\mathbf{U}$, we will determine the next k noise values (where k is the principal subspace dimension)—the reason for determining them simultaneously will become clear soon. Let

$$\mathbf{E}_{t-h+1} \equiv [\mathbf{e}(t-h+1), \ldots, \mathbf{e}(t-k) \mid \mathbf{e}(t-k+1), \ldots, \mathbf{e}(t)]^\mathsf{T}$$
$$\equiv [\mathbf{e}_l^\mathsf{T} \mid \mathbf{e}_r^\mathsf{T}]^\mathsf{T},$$

where $\mid$ denotes elementwise concatenation (for example, $[1,2 \mid 3,4]$ results into two row vectors $[12]$ and $[34]$. The first block $\mathbf{e}_l \in \mathbb{R}^{h-k}$ consists of $h-k$ known values, whereas the second block $\mathbf{e}_r \in \mathbb{R}^k$ consists of the k unknown noise values we wish to determine. Similarly decomposing $\mathbf{Q} \equiv [\mathbf{Q}_l \mid \mathbf{Q}_r]$ into blocks $\mathbf{Q}_l \in \mathbb{R}^{h\times(h-k)}$ and $\mathbf{Q}_r \in \mathbb{R}^{h\times k}$, we can rewrite equation 1 as

$$\mathbf{Q}_l\mathbf{e}_l + \mathbf{Q}_r\mathbf{e}_r = \mathbf{0} \quad \text{or} \quad \mathbf{Q}_r\mathbf{e}_r = -\mathbf{Q}_l\mathbf{e}_l. \qquad (2)$$

This is a linear equation system with k variables and k unknowns. Since the principal subspace has dimension k by construction, the linear system is full-rank and can always be solved. The bottom right of 3(c) highlights the known $\mathbf{e}_l$ and unknown $\mathbf{e}_r$ (with one principle component $k=1$).

The above equation cannot be applied for initial values of the noise; we will use i.i.d. noise for those. Initially, we

Algorithm 4: SACOR

Input : Perturbed value $\mathbf{a}^*(t)$
Old subspace $\mathbf{U} \in \mathbb{R}^{N \times k}$, $\mathbf{\Lambda} \in \mathbb{R}^{k \times k}$
Output: Reconstruction $\tilde{\mathbf{a}}(t)$, new subspace $\mathbf{U}, \mathbf{\Lambda}$

1 Construct window $\mathbf{W}_{t-h+1} = [\mathbf{a}(t-h+1), \ldots, \mathbf{a}(t)]^\mathsf{T}$
2 Update $\mathbf{U}$, $\mathbf{V}$ using $\mathbf{W}_{t-h+1}$
3 Project onto est. eigenspace $\tilde{\mathbf{W}} = \mathbf{U}\mathbf{U}^\mathsf{T}\mathbf{W}_{t-h+1}$
4 Reconstruction is the last element of $\tilde{\mathbf{W}}$, $\tilde{\mathbf{a}}(t) = \tilde{\mathbf{W}}_t^h$

do not know anything about the patterns present in the signal, therefore i.i.d. noise is the best choice, since there are no correlations yet. However, the adversary has also not observed any correlations that can be leveraged to remove that noise. The important point is that, as soon as correlations become present, our method will learn them and use them to intelligently add the noise, before the adversary can exploit this information.

Figures 3(b) and 6 clearly show that our approach accurately tracks the dominant local trends, over a wide range of stream characteristics. Algorithm 3 shows the pseudocode.

The algorithm for reconstructing the original data is simpler; we only need to project each window $\mathbf{W}_t$ onto the current estimate of $\mathbf{U}$, exactly as we did for the correlation case. The pseudocode is shown in Algorithm 4.

The analogues of Theorems 1 and 2 are summarized in the following theorem (proof ommited for space).

Theorem 3. *The perturbed stream from SACAN satisfies $\mathcal{D}(\mathbf{A}, \mathbf{A}^*) = \sigma^2$ and preserves the eigenvectors of the autocovariance matrix. The squared reconstruction error of SACOR on this perturbed stream is approximately σ^2.*

Preserving the autocorrelation properties, in addition to the privacy, is desirable, since several fundamental mining operations, such as autoregressive modelling and forecasting as well as periodicity detection [9], rely on them.

Multi-dimensional extension. If we wish to capture both correlations as well as autocorrelations on multi-dimensional streams, we can decompose the problem in a fashion very similar to [32]. Details are beyond the scope of this work, but we briefly present the main idea. We track the eigenspace of the covariance matrix. However, instead of using it only for adding noise, we also perform PCA on the stream collection, to obtain $k \ll N$ streams of "hidden variables." Subsequently, we can apply our autocorrelation tracking scheme independently on each of these uncorrelated (across dimension) streams. SPIRIT performs precisely the same decomposition of the problem (while controlling the PCA approximation error) [32], except it does so for multi-dimensional autoregression, rather than autocorrelation tracking.

Data Streams	Dimension	Description
Chlorine [13]	4310×166	Environmental sensors
Lab [11]	7712×198	Room sensors
Stock [21]	8000×2	Stock price

Table 2. Three Real Data Sets

6. Experiment

We have implemented the proposed algorithms and study their performance on real data streams. Specifically, we show that: 1) in terms of preserving the input streams' privacy, SCAN and SACAN outperform both i.i.d. noise as well as noise added based on offline analysis; 2) SCOR and SACOR achieve smaller reconstruction error than static, offline algorithms; 3) all proposed algorithms have considerably small computation and memory overhead.

6.1 Setup

The real-world data sets we use are summarized in table 2. Chlorine water quality in a drinking water distribution system and Lab measures light, humidity, temperature and voltage of sensors in the Intel Research Berkeley lab.

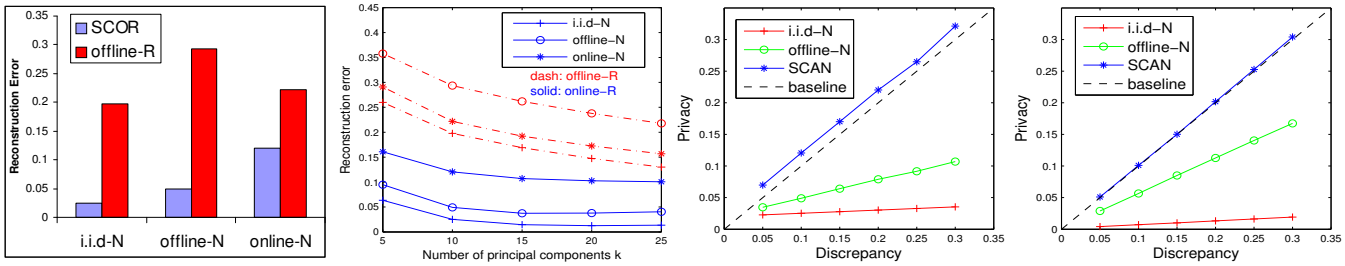
For simplicity, both discrepancy and reconstruction error are always expressed relative to the energy of the original streams, i.e., $\mathcal{D}(\mathbf{A}, \mathbf{A}^*)/\|\mathbf{A}\|_F$ and $\mathcal{D}(\mathbf{A}, \tilde{\mathbf{A}})/\|\mathbf{A}\|_F$, respectively. Equivalently, the streams are normalized to zero mean and unit variance, which does not change their correlation or autocorrelation properties. The random noise distribution is zero mean Gaussian, with variance determined by the discrepancy parameter. Maximum discrepancy is 30%, as large noise will destroy the utility of the perturbed data, making them practically useless for the mining application. Without loss of generality and to facilitate presentation, we assume that perturbation and reconstruction use the same number of principal components—see discussion in Section 6.5. Our prototype is implemented in Matlab and all experiments are performed on an Intel P4 2.0GHz CPU.

6.2 Dynamic Correlation

The perturbation and reconstruction methods investigated in our experiments are summarized in Table 3, where "N" stands for noise and "R" for reconstruction. The offline algorithms, for both perturbation and reconstruction, are essentially the existing work on the static, relational data model, using PCA on the entire stream history to identify correlations and add or remove noise accordingly. Except otherwise specified, we set the number of principal components k to 10. Although the offline algorithms may not be applicable in a streaming setting due to large storage requirements, they are included for comparison and we

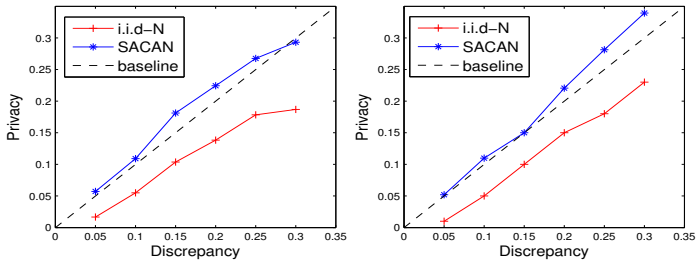
Perturbation	i.i.d-N	offline-N	online-N:SCAN
Reconstruction	baseline	offline-R	online-R:SCOR

Table 3. Perturbation/Reconstruction Method



(a) Reconstruction Error: SCOR vs. offline-R (b) Reconstruction Error: vary k (c) Privacy vs. Discrepancy, online-R: Lab data set (d) Privacy vs. Discrepancy, online-R: Chlorine data set

Figure 4. Privacy Preservation for Streams with Dynamic Correlations



(a) Privacy vs. Discrepancy: Chlorine (b) Privacy vs. Discrepancy: Stock

Figure 5. Privacy vs. Discrepancy: Online Reconstruction

show that, besides high overheads, their performance is sub-optimal due to the time-evolving nature of streams. Finally, baseline reconstruction is simply the perturbed data themselves (i.e., no attempt to recover the original data).

Reconstruction Error Figure 4(a) shows the reconstruction error of online and offline reconstruction, w.r.t all types of noise. The figure presents results from Lab data with discrepancy is set to 10%. In all cases, SCOR clearly outperforms the offline method. The main reason is that offline-R has considerable projection error for streams with dynamically evolving correlation, whereas SCOR has almost negligible projection error and its reconstruction error is dominated by the remaining noise. Similar phenomena were observed for other discrepancy values. Threfore, online reconstruction should be the candidate for measuring the privacy of the perturbed data.

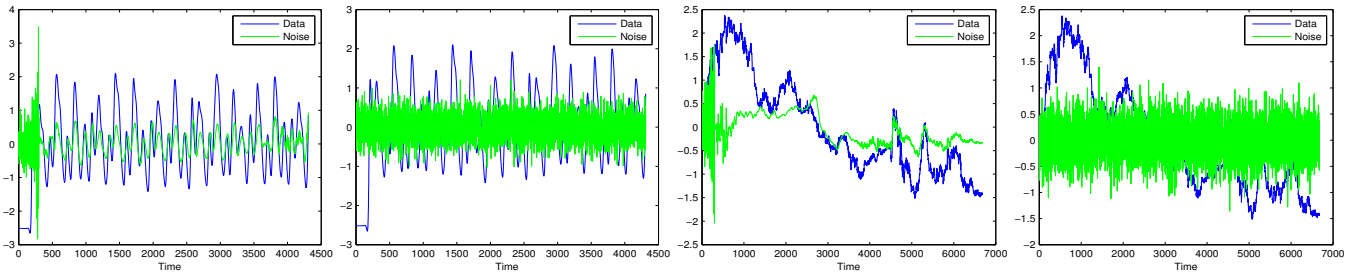
Effect of k The number of principal components k will affect both the projection error and the remaining noise, which in turn have an impact on the overall reconstruction error. Figure 4(b) studies the effect of k on both offline-R and online-R on the Lab data with discrepancy fixed at 10%. For both approaches, reconstruction errors decrease as k increases. Two interesting facts are reflected. First, online-R requires smaller k to reach a "flat," stable reconstruction error. This is beneficial since both the computation and memory cost increase in proportion to k, as we will see in Section 6.4. Second, online-R achieves smaller reconstruction error than offline-R, for all types of noise. This reinforces our decision measure to the privacy of the perturbed data using online-R (SCOR).

Perturbation Performance Next, we measure the ability of different perturbation methods to preserve privacy of data streams with dynamic correlations. Results on the Lab and Chlorine data are presented in figures 4(c) and 4(d). Clearly, for both data streams, SCAN achieves the best privacy over all discrepancy values. SCAN effectively achieves the best privacy w.r.t. the allowed discrepancy. Compared to the baseline method, online-R removes no noise at all from SCAN.

6.3 Dynamic Autocorrelation

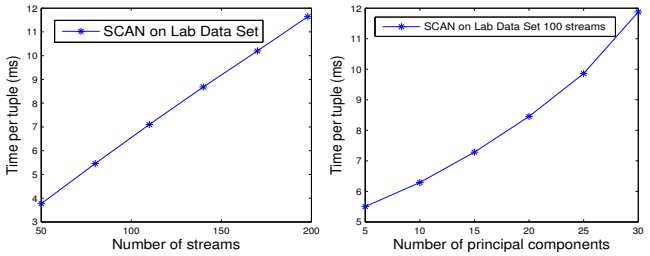
This section presents the results that demonstrate the correctness and effectiveness of our algorithms for data perturbation in streams with autocorrelation. In all experiments, except otherwise specified, the window size h is set to 300 and the number of principal components k is 10. Since there is no previous work that explores autocorrelation in the offline case for privacy preservation, we compare our method against i.i.d. noise and we use the online reconstruction algorithm proposed in this paper, SACOR, in order to measure privacy.

Effectiveness of SACAN The key idea of SACAN is to track the data stream's autocorrelation in an online fashion and produce noise with similar autocorrelation. To illustrate the effectiveness of SACAN, we apply SACAN and i.i.d. noise on different types of streams. We show the results from the Chlorine and Stock data sets in figure 6. The discrepancy is set to 10% for all experiments. We observe from figure 6(a) and 6(c) that SACAN initally produces i.i.d. noise but it is quickly able to estimate and track the autocorrelation of the input stream. Hence, SACAN adds random noise that closely follows the estimated autocorrelation. Intuitively, the noise generated by SACAN exhibits: 1) the same frequency as the input data stream; 2.) amplitude that is determined by the discrepancy. Thus, since the SACAN perturbation follows the same trend as the input stream, it is hardly distinguishable or separable once added.



(a) Chlorine: SACAN (b) Chlorine: i.i.d (c) Stock: SACAN (d) Stock: i.i.d.

Figure 6. Online Random Noise for Stream with Autocorrelation



(a) Time vs. Number of Streams (b) Time vs. Number of Principal Components

Figure 7. Running Time of SCAN

The advantage of SACAN becomes clear when comparing the results shown in figures 6(b) (SACAN) and 6(d) (i.i.d noise).

Privacy of SACAN and Effectiveness of SACOR Using SACOR for online reconstruction, more than $\frac{1}{3}$ of the i.i.d. noise added to the Chlorine and Stock streams can be removed. However, the noise produced by SACAN could not be removed at all. Figure 5 demonstrates these results, for varying discrepancy. The reconstruction error from SACOR is used to measure privacy. Baseline is the discrepancy between perturbed and input streams (i.e., no attempt at reconstruction). Since SACAN produces noise that follows the same trend as the input data stream, it is hard to remove any such noise from the perturbed data.

6.4 Cost Analysis

The cost metrics include the computational requirements, measured in seconds, and the memory consumption, measured in bytes. Figure 7 shows the running time of SCAN on the Lab data set. Figure 7(a) investigates the effects of increasing the number of input streams N while fixing $k = 10$, whereas the second experiment studies the impact of keeping different number of principal components with $N = 100$ input streams. In both cases, we take the average processing time per tuple, consisting of N elements, one from each input stream. The running time of SCAN is linear w.r.t. the number of input streams and almost linear w.r.t. the number of principal components. This indicates that it has good scalability. On average, SCAN is able to process input from hundreds of streams in a few milliseconds. The same experiments performed for other online algorithms (SCOR, SACAN, SACOR) lead to similar observations.

Memory consumption for all of our algorithms is dominated by two factors: 1) the online estimation of local principal components, either for correlation or for autocorrelation; 2) the number of tuples that need to be buffered. For SCAN and SCOR no buffering is required. Hence, memory consumption memory is required only for the k local principal component directions, each of which is represented by an N-dimensional vector, where N is the number of input streams. Therefore, the total memory required is $Nk|\mathbb{R}|$, where $|\mathbb{R}|$ is the floating point representation size. For SACAN/SACOR, each principal direction is an h-dimensional vector. Additionally, the last h values need to be buffered, since the entire window is needed to update the principal direction estimates. Hence, the total memory consumption is $hk|\mathbb{R}| + h|\mathbb{R}|$.

6.5 Discussion

The experimental evaluation validates the superiority of our approaches. Essentially, for both correlations and autocorrelation, our algorithms are able to produce random noise that follows the same trend as the input streams, in a online fashion. In addition, our algorithms have small computation and memory overhead. Finally, we should point out that, with relatively small amount of noise injected into the original data streams, regardless of the type of the noise, the principal components of the perturbed data will be a good approximation of input data streams. Of course, there are ways to mathematically infer the principal components of the input data streams, given the perturbed data streams and some knowledge of the noise properties, such as its distribution and variance. However, in this paper, we do not consider releasing any knowledge about the noise. In fact, we believe this is the right choice as the goal is to maximally preserve the privacy, while maintaining certain utility. Releasing information about the noise might seriously compromise privacy. The assumption that injected noise cannot be too large is automatically guaranteed by the utility requirement of the target mining application. Hence, the esti-

mates for the number of dominant principal components at reconstruction will be similar to that of perturbation. This justifies using the same number of principal components for both reconstruction and perturbation. However, different k in perturbation and reconstruction does not affect the trend and phenomena in our experiments.

7. Conclusion

Data streams in the real world typically exhibit both significant correlations as well as autocorrelation, thereby providing ample opportunities for adversaries to breach privacy. In this paper we develop the basic building blocks for privacy preservation on numerical streams. In particular, we focus on the fundamental cases of correlation across multiple streams and of autocorrelation within one stream. We present methods to dynamically track both and subsequently add noise that "mirrors" these statistical properties, making it indistinguishable from the original data. Thus, our techniques prevent adversaries from leveraging these properties to remove the noise and thereby breach privacy. We provide both a mathematical analysis and experimental evaluation on real data to validate the correctness, efficiency, and effectiveness of our algorithms. Our techniques track the evolving nature of these relationships and achieve much better results than previous static, global approaches. Furthermore, to the best of our knowledge, autocorrelation-based attacks have not been previously addressed.

References

[1] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *EDBT*, 2004.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS*, 2001.

[3] R. Agrawal and R. Srikant. Privacy preserving data mining. In *SIGMOD*, 2000.

[4] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *SIGMOD*, 2005.

[5] S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *ICDE*, 2005.

[6] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS*, 2002.

[7] E. Bertino, B. Ooi, Y. Yang, and R. Deng. Privacy and ownership preserving of outsourced medical data. In *ICDE*, 2005.

[8] J. Bethencourt, D. Song, and B. Waters. Constructions and practical applications for private stream searching. In *IEEE Symposium on Security and Privacy*, 2006.

[9] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasing*. Springer, 2nd edition, 2003.

[10] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *ICDM*, 2005.

[11] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2005.

[12] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *SIGKDD*, 2003.

[13] EPANET, 2002. `http://www.epa.gov/ORD/NRMRL/wswrd/epanet.html`.

[14] European-Union. European union:directive on privacy protection, 2002. http://europa.eu.int/eur-lex/pri/en/oj/dat/2002/l_201/l_20120020731en00370047.pdf.

[15] A. Evfimevski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.

[16] A. Evfimievski, R. Srikant, R. Agarwal, and J. Gehrke. Privacy preserving mining of association rules. In *SIGKDD*, 2002.

[17] M. Ghil, M. Allen, M. Dettinger, K. Ide, D. Kondrashov, M. Mann, A. Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Yiou. Advanced spectral methods for climatic time series. *Rev. Geophys.*, 40(1), 2002.

[18] S. Guha, D. Gunopulos, and N. Koudas. Correlating synchronous and asynchronous data streams. In *KDD*, 2003.

[19] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, 4th edition, 2002.

[20] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *SIGMOD*, 2005.

[21] INET ATS, Inc. http://www.inetats.com/.

[22] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[23] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM*, 2003.

[24] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *SIGMOD*, 2006.

[25] F. Li, C. Chang, G. Kollios, and A. Bestavros. Characterizing and explorting reference locality for data stream applications. In *ICDE*, 2006.

[26] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYTO*, 2000.

[27] K. Liu, H. Kargupta, and J. Ryan. Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining. *IEEE TKDE*, 18(1), 2006.

[28] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[29] S. Muthukrishnan. Data streams: Algorithms and applications. Technical report, Computer Science Department, Rutgers University, 2003.

[30] A. V. Oppenheim and A. S. Wilsky. *Signals and Systems*. Prentice-Hall, 1983.

[31] R. Ostrovsky and W. Skeith. Private searching on streaming data. In *CRYPTO*, 2005.

[32] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, 2005.

[33] S. Papadimitriou and P. Yu. Optimal multi-scale patterns in time series streams. In *SIGMOD*, 2006.

[34] R. O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Trans. Ant. Prop.*, 34(3), 1986.

[35] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 2002.

[36] K. Thearling. Data mining and privacy: A conflict in making. In *DS**, 1998.

[37] J. Vaidya and C. W. Clifton. Privacy prserving association rule mining in vertically partitionaed data. In *SIGKDD*, 2002.

[38] K. Xiao and Y. Tao. Personalized privacy preservation. In *SIGMOD*, 2006.

[39] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, 2002.