Time Series Compressibility and Privacy

Spiros Papadimitriou† Feifei Li‡ George Kollios‡ Philip S. Yu†

†IBM T.J. Watson Research Center
Hawthorne, NY, USA
{spapadim,psyu}@us.ibm.com

‡Computer Science Dept., Boston University
Boston, MA, USA
{lifeifei, gkollios}@cs.bu.edu

ABSTRACT

In this paper we study the trade-offs between time series compressibility and partial information hiding and their fundamental implications on how we should introduce uncertainty about individual values by perturbing them. More specifically, if the perturbation does not have the same compressibility properties as the original data, then it can be detected and filtered out, reducing uncertainty. Thus, by making the perturbation "similar" to the original data, we can both preserve the structure of the data better, while simultaneously making breaches harder. However, as data become more compressible, a fraction of the uncertainty can be removed if true values are leaked, revealing how they were perturbed. We formalize these notions, study the above trade-offs on real data and develop practical schemes which strike a good balance and can also be extended for on-the-fly data hiding in a streaming environment.

1. INTRODUCTION

Time series data are prevalent in a wide range of domains and applications, such as financial, retail, environmental and process monitoring, defense and health care. Additionally, massive volumes of data from various sources are continuously collected. However, data owners or publishers may not be willing to exactly reveal the true values due to various reasons, most notably privacy considerations. A widely employed and accepted approach for partial information hiding is based on random perturbation [4], which introduces uncertainty about individual values. Consider the following examples:

(E1) A driver installing a vehicle monitoring system [5, 36] may not wish to reveal his exact speed. How can he, e.g., avoid revealing small violations of the speed limit (say, by 3–5 mph) but still allow mining of general driving patterns or detection of excessive speeding?

(E2) A financial services company may wish to provide a discounted, lower-quality price ticker with a specific level of uncertainty, which is not useful for individual buy/sell decisions but still allows mining of trends and patterns. How can they ensure that the level of uncertainty is indeed as desired?

(E3) Similarly, a financial institution [43] may not wish to reveal the amounts of individual transactions over time, while still allowing mining of trends and patterns. How can they control the level of uncertainty (or, privacy) in the published data and ensure that nothing more can be inferred?

Prior work on numerical and categorical data has focused on the traditional relational model, where each record is a tuple with one or more attributes. Existing methods can be broadly classified into two groups and operate either (i) by direct perturbation of individual attributes separately [4, 3, 18] or of entire records independently [24, 23, 32, 10], or (ii) by effectively swapping or concealing values among a small and appropriately chosen group of "neighboring" records [38, 2, 6, 33].

Although some of the prior work on relational data has considered certain forms of privacy breaches that are possible by exploiting either the global or local structure of the data [33, 23, 24, 10], the additional aspect of time poses new challenges, some of which are related to fundamental properties of time series [17]. In particular: (i) Sophisticated filtering techniques may potentially reduce uncertainty thereby breaching privacy; (ii) Time series can be "described" in a large number of ways. In a sense, a univariate time series is a single point in a very high-dimensional space [1]—e.g., if the series has 1000 values, there are many 1000-dimensional bases to choose from; (iii) Time series characteristics may change over time and, in a streaming setting, new patterns may emerge while old ones change. For example, we cannot know about quarterly or annual trends while still collecting the first week of data. Thus, both static, global as well as fixed-window analysis are unsuitable.

In this paper we focus on univariate time series, examine the trade-offs of methods for partial information hiding via data perturbation, and propose a practical approach that we evaluate against both filtering attacks and, also, true value leaks. Additionally, our approach is suited for time-evolving (i.e., non-stationary) series and can be adapted for on-the-fly data hiding in a streaming setting.

The main idea is exemplified by two extreme cases, which are explained in more detail in Section 3.1. True value leaks reveal the perturbation at particular time instants. If we wish to ensure that such information does not help infer anything about the perturbation of other time instants, then

VLDB '07, September 23-28, 2007, Vienna, Austria.

Sym.	Description
x_t	True value at time t, $1 \leq t \leq N$.
$\{x_t\}$	Set of all values (i.e., series).
N	Number of values (total or so far).
n_t	Perturbation at time t.
y_t	Published value at time t, $y_t := x_t + n_t$.
$\tilde{y}_t$	Reconstruction via filtering.
$\hat{y}_t$	Reconstruction via linear regression.
$w_{\ell,t}$	Wavelet (aka. detail) coeff. at level ℓ and time t.
$v_{\ell,t}$	Scaling (aka. approximation) coeff. at level ℓ and time t.
$\{w_{\ell,t}\}$	Set of all wavelet coefficients.
L	Maximum wavelet level, $L \leq \log_2 N$.
χ_k	Fourier coefficient.
$\{\chi_k\}$	Set of all Fourier coefficients.
σ	Discord, $\sigma^2 := \mathrm{Var}[n_t] = \mathrm{E}[n_t^2]$.
$\tilde{\sigma}$	Uncertainty after filtering, $\tilde{\sigma}^2 := \mathrm{Var}[\tilde{y}_t - x_t]$.
$\hat{\sigma}$	Uncertainty after regression, $\hat{\sigma}^2 := \mathrm{Var}[\hat{y}_t - x_t]$.
K	Number of coefficients with magnitude greater than σ.

Table 1: Symbols and their descriptions.

each time instant must be perturbed independently of others. However, if the series exhibit certain patterns, such independent perturbation of each value in the time domain can be distinguished from the original data and filtered out. On the other hand, ensuring complete protection against any filtering method requires a perturbation that is completely indistinguishable from the original series. This can be achieved only by making the perturbation a rescaled, exact copy of the data. However, in this case, even a single true value can reveal how all other values have been perturbed.

In the first case, *each* time instant is perturbed independently, while in the second case *all* time instants are perturbed in the same way. But what if we perturb groups (or, windows) of values in the same way within a group, but differently across groups? How should these groups be chosen? Based on this insight, we address these questions using both Fourier and wavelet transforms.

Contributions. Our main contributions in this paper are the following:

- Expose and study the relationship between data representation, compressibility and partial information hiding via perturbation, in the context of time series.
- Introduce the notion of compressible perturbation, which determines how to perturb the data depending on the perturbation magnitude and on the properties of the data.
- Examine the trade-off between breaches that exploit compressibility via filtering operations and breaches that rely on leaks of true (i.e., unperturbed) values.
- Present schemes that are based on the Fourier transform and on wavelets. Our wavelet-based scheme is also amenable to streaming time series.

These are all challenging issues that are often overlooked and, even though the present work does not provide theoretical guarantees, it represents an important first step towards a practical framework that addresses these issues. We demonstrate the trade-offs between privacy and compressibility, as well as the efficiency and effectiveness of our approach on real time series.

The rest of the paper is organized as follows: Section 2 briefly explains the necessary background on wavelets and filtering. Section 3 lays the groundwork on compressibility and privacy of time series data. Section 4 presents our framework for compressible perturbation, in both batch and streaming settings. Section 5 presents the experimental evaluation and Section 6 discusses broadly related work. After a brief discussion in Section 7, we conclude in Section 8.

2. BACKGROUND

In this section we summarize the necessary background on wavelets and filtering. The main notation is summarized in Table 1.

2.1 Discrete wavelet decomposition

Wavelets are best introduced with the Haar transform, because of its simplicity. A more rigorous introduction to wavelets along with an introduction to the Fourier transform can be found, e.g., in [35]. Given a series with N points, we define $v_{0,t} := x_t$ to start the Haar DWT construction. At each iteration, or *level* $\ell = 1, 2, \ldots, \log_2 N$, we perform two operations on $v_{\ell-1,t}$ to compute the coefficients at the next level:

- Differencing, to extract the high frequencies of $v_{\ell-1,t}$, which gives the *wavelet coefficients* $w_{\ell,t} = \sqrt{2}(v_{\ell-1,2t} - v_{\ell-1,2t-1})$ that form the *detail* component of level ℓ.
- Smoothing, which averages each consecutive pair of values and extracts the remaining low frequencies of $v_{\ell,t}$, obtaining the *scaling coefficients* $v_{\ell,t} = \sqrt{2}(v_{\ell-1,2t} + v_{\ell-1,2t-1})$ that form the *smooth* component of level ℓ.

The scaling factor of $2^{-1/2}$ ensures that the total energy (i.e., sum of squares of all values) is preserved. The coefficients of level $\ell + 1$ are half as many as those of ℓ and correspond to a time window twice the size. We stop when $w_{\ell,t}$ consists of one coefficient, which happens at $\ell = \log_2 N + 1$. The total number of wavelet coefficients across levels is $N - 1$.

There are several families wavelet transforms that follow the above recursive *pyramid algorithm*, using a pair of filters, one high-pass and one low-pass. For example, in Haar wavelets, this pair consists of the simple first-order differencing and averaging filters, respectively. More generally, for each $L \geq 1$, Daubechies-L (or DB-L) wavelets use an L-th order difference filter for the high-pass operation and the corresponding low-pass filter (for more details, see [35]). These filters have $2L$ non-zero coefficients.

Time/frequency decomposition. Figure 1a illustrates how Haar wavelets decompose a series into time and scale. Each scale approximately corresponds to a frequency band and each wavelet coefficient within that band "summarizes" the corresponding frequency content within a localized time window. For comparison, Figure 1b shows a pure-frequency decomposition. Each coefficient contains information about a single frequency (sinusoid), but has no time information, since the basis (i.e., sinusoid) for each coefficient is not localized. In practice, series often exhibit jump discontinuities, and frequency shifts. Therefore some localization is necessary [13]. Short-window Fourier analysis uses DFT on a

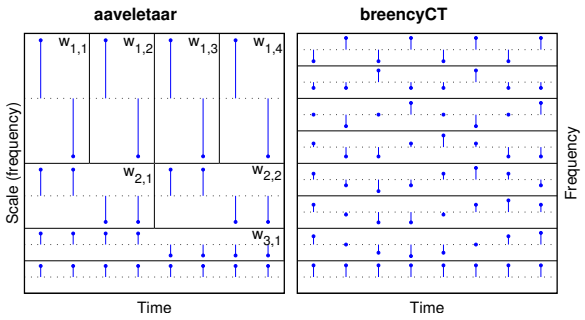


Figure 1: Illustration of time-frequency properties.

fixed-size window. This poses limitations on the minimum frequencies that can be captured, as well as the localization in time of each coefficient. In a wide range of application domains, the jointly varying window size and bandwidth make wavelets well-suited for analysis and representation [13, 29].

Streaming estimation. In the above example, note that estimation of both $v_{\ell,t}$ and $w_{\ell,t}$ requires only the two last scaling coefficients from the previous level, $v_{\ell-1,2t}$ and $v_{\ell-1,2t+1}$. In general, Daubechies-L DWT requires the last $2L$ scaling coefficients from the previous level. Thus, it is possible to perform the DWT incrementally as new points arrive, by buffering only $2L$ numbers for each of the $\ell \leq \log_2 N$ levels. The total time required is still proportional to N, i.e., constant per new value.

2.2 Compressibility and shrinkage

Because of their time/frequency decomposition properties, wavelets have been successfully used in signal estimation and denoising [16, 14]. Our presentation in this section summarizes that of [13].

Assume that we are given the representation of a time series with N points in some basis. This representation consists of N numbers and can be obtained by applying an orthonormal transform (i.e., change of coordinates in N-dimensional space) to the original series $\{x_t\}_{t=1}^N$. Also assume that the noise is i.i.d. (i.e., white) and its variance σ is known. Given the above, the ideal denoiser is simple: coefficients whose magnitude is below σ are discarded as noise, otherwise they are retained. Then, the important questions are: (i) how to choose an appropriate basis, (ii) how to estimate σ when it is not known, and (iii) what to do with the retained coefficients.

For the first question, we ideally want the basis that compresses the signal into the smallest possible number of coefficients or, equivalently, has the largest possible number of zero coefficients. This implies that the remaining, non-zero coefficients will have a large magnitude, making them easy to distinguish from noise coefficients. Of course, it is not possible to know this optimal representation for a single series; differently put, the optimal basis for a specific realization of a series is always just the series itself, which is not very useful. Therefore, we want to choose a representation that is appropriate for a class of signals. As already mentioned, wavelets successfully compress many real-world series [13], because of their time/frequency decomposition properties. Thus, they are very often an appropriate choice.

Having chosen wavelets to represent the series, it can be shown that the risk-optimal estimate of the noise variance is the median over t of the absolute magnitude, $|w_{1,t}|$, of the first-level coefficients [16]. Additionally, the best way to perform thresholding is to shrink each retained coefficient towards zero, rather than keeping them intact. This is also known as *soft thresholding* and its application to the wavelet representation is known as *wavelet shrinkage*.

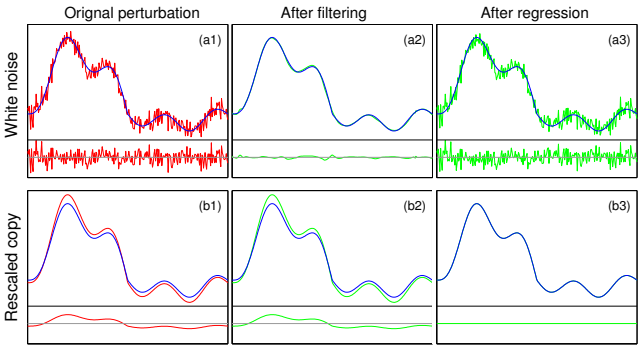


Figure 2: Illustration of intuition, via two simple, extreme examples: (a1–3) perturbation most resilient to any true value leaks, and (b1–3) most resilient to any linear filtering.

3. PRIVACY AND COMPRESSION

We first explain the fundamental intuition in Section 3.1, which exposes the key issues and questions. Subsequently, we address each of those questions in the remainder of this section and in Section 4.

3.1 Intuition and motivation

We illustrate the intuition with two extreme cases in Figure 2. The original series is shown in dark blue, and consists of 200 values. The perturbation added and the published series are shown in red. The perturbation that remains after either a filtering attempt or after true value leaks are shown in green.

For both extremes we assume that, in the worst case, an attacker may have full knowledge of the true data, but in different ways. In the first, we allow an attacker direct access to an arbitrary number of true values (in the time domain). In the second extreme, we allow the attacker to know the shape of the series with arbitrary accuracy (i.e., the attacker may know the one-dimensional subspace spanned by the series itself). We always assume that an attacker uses linear functions/filters to obtain estimates of the true data [23, 28].

Figures 2(a1–3) illustrate the perturbation that is resilient to any number of true value leaks. In this case, each time instant must be perturbed independently of others, in order to prevent any inferences across values. This requirement is always satisfied by white noise, i.e., independent, identically distributed random values. A realization of a white noise process is shown in the bottom panel of Figure 2(a1). We add this to the original series and obtain the published series, shown with a red line in the top panel of Figure 2(a1). The linear regression estimate of the true values versus the perturbed values is shown in Figure 2(a3). As expected, the true values cannot be recovered. However, white noise is also uncorrelated with the original data (no matter what the data are), leading to the potential vulnerability illustrated in Figure 2(a2), which shows the output of a wavelet-based filter.

Figures 2(b1–3) illustrate the perturbation that is resilient to knowledge of the exact shape of the series. In this case, the perturbation must be completely indistinguishable from the original series. In other words, it should be perfectly cor-

related with the original series. Clearly, this is guaranteed if the perturbation is an exact copy of the original series, except for rescaling of all values by the same factor. The result is shown in Figure 2(b1), with the same perturbation magnitude as in the previous example. As expected, any kind of linear filtering is unable to separate the perturbation from the true series—see Figure 2(b2). However, if even a single true value is leaked, then all true values can be inferred, as illustrated in Figure 2(b3), which shows the linear regression estimates.

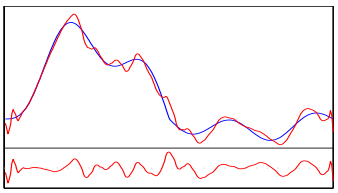
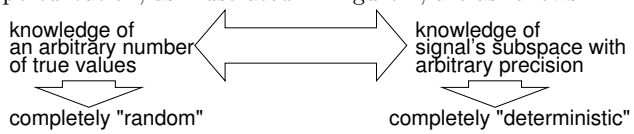


Figure 3: Our goal is to automatically find a perturbation with the same "smoothness" properties as the data, under a broad linear class of series (here, signals with compact wavelet representation), while simultaneously allowing for enough variation to prevent linear reconstruction from true value leaks.

Summarizing, the two extreme assumptions about background knowledge, and the corresponding best choices for perturbation, as illustrated in Figure 2, are as follows:



An adversary may have a combination of such knowledge, therefore we need to automatically find a balance between fully "deterministic" and fully independent perturbation—see Figure 3 for an example of our technique, where neither filtering nor linear estimation based on leaks can remove more than 1% of the perturbation. We propose practical techniques to address this challenge and evaluate them on a number of real datasets. We give the necessary definitions in Section 3.2 and describe our proposed techniques in Section 4.

3.2 Measuring privacy

A common measure of uncertainty is standard deviation, i.e., root mean square value of a series. We will use standard deviation to measure two important aspects: (i) discord between perturbed and original data, and (ii) remaining uncertainty about the true values, after attempts to recover them. We want the discord to be as low as possible and, in particular, at most equal to a chosen threshold. The utility of the published data drops as the discord increases [20, 27]. On the other hand, given the discord, we want the remaining, "true" uncertainty to be as high as possible, ideally equal to the discord. Next, we formally define these notions.

DEFINITION 1 (ADDITIVE PERTURBATION). *Given a series x_t, for $t \geq 1$, we choose a corresponding perturbation series n_t with zero mean, $\mathrm{E}[n_t] = 0$, and publish the series $y_t := x_t + n_t$, for all $t \geq 1$.*

DEFINITION 2 (DISCORD). *The discord σ is the standard deviation of the perturbation, i.e.,*

$$\sigma^2 := \mathrm{Var}[y_t - x_t] = \mathrm{Var}[n_t] = \mathrm{E}[n_t^2].$$

The discord threshold is given and determines both the maximal loss of information we are willing to tolerate, as well as the maximum uncertainty that can be introduced. In fact, these two quantities should be equal and this is precisely our goal. However, they may not be equal, because an adversary can apply techniques that reduce the uncertainty.

Given the discord threshold, we will always fully exploit all the available perturbation latitude, i.e., our goal will be to add a perturbation amount equal to the threshold. Thus, from now on, we will not distinguish between the discord and its threshold, using σ to denote both.

Given the published values y_t, for $t \geq 1$, an adversary may attempt to obtain an estimate of the true values, which may reduce the overall uncertainty. The discord (i.e., uncertainty originally introduced by the data publisher) is the standard deviation of the difference between true and published values. Similar to this, we will measure the remaining uncertainty using the standard deviation of the difference between true values and the adversary's estimates. This remaining uncertainty is a measure of privacy achieved under each attack setting.

We shall consider two types of true value estimation attempts, each with different, worst-case assumptions about the available background knowledge. In both cases, we assume that an adversary applies *linear* functions or filters to obtain an estimate of the true values.

Reconstruction via filtering. The first type relies on linear filtering methods to separate the perturbation from the true data. The filtering technique we shall employ is described in Section 2.2 and has been proven very successful in a wide range of domains and applications. [16, 14].

DEFINITION 3 (FILTERING UNCERTAINTY). *Let $\tilde{y}_t$ be the result of a linear filtering operation on the published series y_t. The* filtering uncertainty *is the remaining uncertainty after this operation, i.e.,*

$$\tilde{\sigma}^2 := \mathrm{Var}[\tilde{y}_t - x_t].$$

In practice, we estimate the standard deviation $\tilde{\sigma}$ of the filter's output by applying the filtering operation on a finite time series consisting of N points and using the sample estimate of the standard deviation, $\tilde{s}^2 := \sum_{t=1}^{N} (\tilde{y}_t - x_t)^2 / N$. With a slight abuse of notation, we will denote the sample estimate with $\tilde{\sigma}$ instead of $\tilde{s}$.

In this case, an adversary has the background knowledge that the signal has a compact representation in some space, and more specifically, that the largest fraction of its energy is concentrated on a few transform coefficients. This is a common assumption in signal estimation and recovery [15, 14], and amounts to a constraint on the "shapes" that the series is allowed to have. All practical applications of signal recovery need an assumption about the actual transform. Wavelet-based techniques have been shown most successful for a wide range of real-world signals [16], typically performing at least as well as Fourier-based techniques.

Reconstruction from true value leaks. The second type of attempt to partially remove the perturbation relies on true value leaks. By construction $y_t = x_t + n_t$ and, if n_t is Gaussian white noise, this is precisely the model for least-squares linear regression. This observation leads naturally to the next definition.

DEFINITION 4 (LEAK UNCERTAINTY). *Let $\hat{y}_t$ be the linear regression estimate obtained by fitting a line to the true vs. perturbed values, i.e., $\hat{y}_t = \alpha y_t + \beta$ where $\hat{y}_t$ are chosen so as to minimize the residual error $\sum_t (x_t - \hat{y}_t)^2$. This RMS error is our measure of* true value leak uncertainty*, i.e.,*

$$\hat{\sigma}^2 := \mathrm{Var}[\hat{y}_t - x_t].$$

In practice, we need to estimate $\hat{\sigma}$ from a finite sample. The least-squares estimators of α and β are

$$a := \frac{\sum_{t=1}^{N}(x_t - m_x)(y_t - m_y)}{\sum_{t=1}^{N}(x_t - m_x)^2}, \text{ and } b := m_y - a m_n$$

where $m_x = \sum_{t=1}^{N} x_t/N$ and $m_y = \sum_{t=1}^{N} y_t/N$ are the sample means. The sample estimate of the residual variance is $\hat{s}^2 := \sum_{t=1}^{N}(x_t - a y_t - b)^2/N$. Since a and b are unbiased estimators, their expectation over all finite samples is $\mathrm{E}[a] = \alpha$ and $\mathrm{E}[b] = \beta$.

Leak uncertainty is the minimum error that any linear function can achieve in estimating the true values, even if we assume that an adversary knows all true values. Therefore, our measure is a worst-case estimate of privacy loss, under the assumptions that an adversary uses linear estimation techniques and has access to any number of true values.

Furthermore, the distribution of $N\hat{s}^2/\tilde{\sigma}^2$ is χ^2 with $N-2$ degrees of freedom [11]. Therefore, even if a small subset of $M < N$ samples was used to estimate $\hat{s}$, its expectation over all leaks of size M would still be $\mathrm{E}[\hat{s}^2] = \hat{\sigma}^2(M-2)/M \approx \hat{\sigma}^2$. The standard deviation $\mathrm{Dev}[\hat{s}^2]$ drops quickly, in proportion to $\hat{\sigma}^2/M$. Finally, again with a slight abuse of notation, from now on we will use $\hat{\sigma}$ instead of $\hat{s}$.

Utility. For single time series, trends and patterns often refer to bursts [42] and dominant periods [40]. Such analysis is largely performed on the spectrum of the signal. Whether a perturbation preserves these key properties depends on (i) how much perturbation is added (i.e., the discord), and (ii) how the perturbation is added. In most perturbation methods, the first is a parameter determined by the end user. Additionally, both of our perturbation techniques, by design preserve the spectral and "smoothness" properties of the original signal. Hence, the proposed perturbation techniques will be useful in preserving both privacy and utility of time series.

Summary. We consider two potential breaches separately, with different assumptions about background knowledge. In the first case, we assume an adversary knows that the series has a compact representation in some linear subspace (e.g., few non-zero wavelet or Fourier coefficients). In the second case we assume that an adversary knows any number of true values, in the time domain. In both cases we assume that linear estimation techniques are used. We propose practical techniques to address both challenges and we evaluate our techniques under the two different attack models on a number of real datasets.

4. COMPRESSIBLE PERTURBATION

As pointed out, the simple solution of perturbing the series with white noise does not work, because white noise is incompressible under any representation (or basis). As a result, the added perturbation is "diluted" over coefficients that are not important in representing the series. Consequently, a large portion of the white noise can be removed, leading to a significant decrease in remaining (i.e., true) uncertainty over individual true values. Our goal is to avoid this problem, by appropriately adapting the perturbation to the original series.

4.1 General algorithm

The perturbation n_t for each value at time t will be chosen based on a given discord σ and, of course, the series $\{x_t\}$ itself. Since (i) it is impossible to design a method that is optimally resilient against *both* filtering and leak attacks, and (ii) filtering is possible at any and all time instants since it requires no prior knowledge about the true data, we consequently use resilience to filtering as the primary guide in designing our techniques, but also take leak attacks into consideration. We evaluate our methods with respect to both potential attacks. The general steps to construct the perturbation are:

(S0) Choose a "description" or basis.

(S1) Perturb only those coefficients that are "important" (to be made precise later) in the chosen description.

(S2) Determine by how much to perturb them.

The first step consists of applying an orthonormal transform which, given the N time domain values x_t, for $1 \le t \le N$, will produce another set of N coefficients, c_i for $1 \le i \le N$. Let us assume for the moment that we add Gaussian white noise with variance σ^2. This means that we perturb *each* coefficient by a random number c_i' drawn from a Gaussian distribution with zero mean and standard deviation σ, $c_i' \leftarrow \textsc{GaussRandom}(0, \sigma)$ for all $1 \le i \le N$. We may think of this as allocating N noise "units" (each corresponding to a per time instant perturbation of magnitude σ) equally among all N coefficients. In this case, the resulting perturbation sequence[1] n_t for $1 \le t \le N$ has the same statistical properties (i.e., Gaussian white noise with the same variance) under any orthonormal basis. Therefore, for i.i.d. Gaussian n_t, the choice of representation is not important.

However, this approach is susceptible to filtering attacks. Therefore, we will choose a basis that successfully compresses a large class of time series, in the sense that it concentrates their energy into few transform coefficients. Recall that the ideal denoiser, given a basis, discards all coefficients below the (true or estimated) noise variance. Therefore, any noise embedded into such coefficients is "wasted," as it can be easily separated from the dominant coefficients. This observation leads to the conclusion that only those coefficients with magnitude greater than σ are "important" for perturbing the data in a way resilient to filtering attacks.

Therefore, instead of allocating the N available noise units into all N coefficients, we will allocate them to the set of coefficients whose magnitude exceeds σ. Let $\mathcal{I} := \{i : |c_i| \ge \sigma\}$ be the set of their indices. However, in order to ensure that $\mathrm{Var}[n_t] = \sigma^2$, we need to also change the variance of the random number that will be added to each c_i, for $i \in \mathcal{I}$. For example, a simple choice would be a random number with variance $\sigma\sqrt{\rho_i}$ to each of them, where $K := |\mathcal{I}|$ is the number of coefficients that exceed σ and $\rho_i := N/K$ is the "noise allocation density." This ensures that $\mathrm{E}[\sum_i c_i'^2/N] =$

[1]The sequence $\{n_t\}$ is obtained by applying the inverse transform on $\{c_i'\}$.

$\mathrm{E}[\sum_{i\in\mathcal{I}} c_i'^2]/N + \mathrm{E}[\sum_{i\notin\mathcal{I}} c_i'^2]/N = K\rho_i\sigma^2/N + (N-K)\cdot 0/N = K(N/K)\sigma^2/N + 0 = \sigma^2$, since each $c_i' \in C$ is perturbed independently. Thus, the expected sample variance of the perturbation series will be σ^2 as desired. More generally, we can choose any ρ_i such that $\sum_i \rho_i = N$.

The general steps (S0–2) are shown in detail below (algorithm CompressiblePerturbation). We will make them concrete next, in Sections 4.2 and 4.3.

Algorithm 1 CompressiblePerturbation ($\{x_t\}_{t=1}^N$, σ)

0 $\{c_i\} \leftarrow$ Transform($\{x_t\}$) *//Transform coefficients*
1 $\mathcal{I} \leftarrow \{i : |c_i| \geq \sigma\}$ *//Important w.r.t. σ*
2 $\rho_i \leftarrow \begin{cases} \text{Importance}(c_i) & \text{if } i \in \mathcal{I} \\ 0 & \text{if } i \notin \mathcal{I} \end{cases}$, for all $1 \leq i \leq N$ such that $\sum_i \rho_i = N$
for each coefficient c_i **do**
$c_i' \leftarrow$ GaussRandom($0, \sigma\sqrt{\rho_i}$) *//Perturbation coeffs*
$\{n_t\} \leftarrow$ InverseTransform($\{c_i'\}$) *//Perturbation*
$y_t \leftarrow x_t + n_t$, **for all** $1 \leq t \leq N$ *//Published series*

4.2 Batch perturbation

In this section, we propose two batch perturbation methods that rely on pure frequency or on time/frequency representations of the series. The first is based on the well-established Fourier representation of the entire, length-N series. The second is based on the wavelet representation. First, we study Fourier and wavelet perturbation in a batch setting. We revisit the wavelet-based scheme in Section 4.3, adapting it to a streaming setting.

Pure frequency perturbation. CompressiblePerturbation using the Fourier representation, which decomposes the series into pure sinusoids, is shown in algorithm FourierPerturb. We denote with χ_k, for $1 \leq k \leq N$, the Fourier transform of x_t, $1 \leq t \leq N$, and with ν_k the Fourier transform of the perturbation n_t that we want to construct. For simplicity, the pseudocode only shows the case for N odd. If N is even, then the Fourier coefficient $\chi_{N/2+1}$ at the Nyquist frequency must be treated as a special case.

Algorithm 2 FourierPerturb ($\{x_t\}_{t=1}^N$, σ)

$M \leftarrow (N-1)/2$ *//Case N odd only, due to space*
$\{\chi_k\} \leftarrow \mathrm{FFT}(\{x_t\})$ *//Fourier transform*
for $k = 1$ to M **do**
1 $p_k \leftarrow \sqrt{2}|\chi_{k+1}|$ *//All freqs, except DC (i.e., mean value)*
$\mathcal{I} \leftarrow \{k : p_k \geq \sigma\}$
$K \leftarrow 2|\mathcal{I}|$ *//No. of freqs exceeding σ*
2 $P \leftarrow \sum_{k\in\mathcal{I}} p_k^2$
$\nu_1 \leftarrow 0$ *//Zero DC coeff for perturbation*
for $k = 1$ to M **do**
if $p_k \geq \sigma$ **then**
3 $\rho_k \leftarrow M(p_k^2/P)$
4 $\nu_{k+1} \leftarrow \text{GaussRnd}\left(0, \frac{\sigma}{2}\sqrt{\rho_k}\right) + i\,\text{GaussRnd}\left(0, \frac{\sigma}{2}\sqrt{\rho_k}\right)$
5 $\nu_{n-k+1} \leftarrow \nu_{k+1}^*$ *//Complex conjugate*
else
ν_{k+1} and $\nu_{n-k+1} \leftarrow 0$
$\{n_t\} \leftarrow \text{InvFFT}(\{\nu_k\})$ *//Inverse FFT (zero DC)*
$y_t \leftarrow x_t + n_t$, **for all** t *//Published series*

Intuitively, each sinusoid is perturbed by randomly changing its magnitude and phase (lines 4–5 in FourierPerturb). In more detail, since x_t is real-valued, its Fourier transform is symmetric, i.e.,

$$\chi_{k+1} = \chi_{N-k+1}^*, \text{ for } k = \begin{cases} 1,\ldots,(N-1)/2 & \text{if } N \text{ odd} \\ 1,\ldots,N/2-1 & \text{if } N \text{ even} \end{cases}, \quad (1)$$

where χ_{N-k+1}^* denotes the complex conjugate of χ_{N-k+1}. The DC coefficient χ_1 is always real and equal to the series' mean. If N is odd (this case is not considered in FourierPerturb), then $\chi_{N/2+1}$ is also real. We ensure that ν_k, $1 \leq k \leq N$, also satisfies the same property (line 5 in FourierPerturb), so that the perturbation is also real-valued.

Because of Equation (1), essentially the first half of the Fourier transform carries all the necessary information. We compute the square root of per-frequency energy in line 1 of FourierPerturb. From Equation (1), $|\chi_{k+1}| = |\chi_{N-k+1}|$, so that $\sum_k p_k^2 = \sum_t x_t^2$ (assuming that x_t is zero mean). We then use this information to decide which frequencies to perturb.

For each frequency that exceeds σ, we choose a complex Gaussian random number, which perturbs the amplitude and phase independently, as shown in Figure 4.

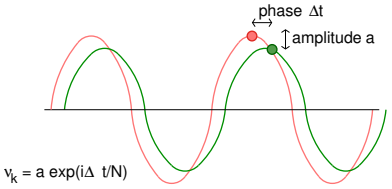


Figure 4: Illustration of lines 4–5 in FourierPerturb.

The allocation of "noise units" into the important frequencies is done in proportion to N/K (as explained in Section 4.1) as well as in proportion to the energy content of each perturbed frequency (factor of p_k^2/P in line 3 of FourierPerturb). This is the best choice for resilience to filtering attacks, as it tends to concentrate most of the perturbation into a few dominant frequencies. However, this may increase the "regularity" of the perturbation and make it somewhat more susceptible to true value leaks. We found that *per-band weighting* of the frequencies above the threshold σ (i.e., inclusion of the p_k^2/P factor in line 3 of FourierPerturb) has small impact on true value leaks, while in certain cases significantly reduces resilience to filtering attacks. As we shall see later, the wavelet representation does not suffer from such problems, allowing a simpler decision on how to allocate "noise units".

Fourier-based perturbation generally performs well for series dominated by a few frequencies which do not change over time. If the series has discontinuities or frequency shifts, then Fourier may perform worse, because phenomena localized in time are spread across frequencies. This effect would allow a potential attacker to remove more uncertainty, roughly in proportion to the magnitude of such discontinuities (either in time or in frequency) and in inverse proportion to the number of frequencies.

Finally and more importantly, the Fourier transform of a growing series cannot be updated incrementally. One potential solution might be to use the short-time Fourier transform (STFT). As we shall see, a a fixed-size time window is undesirable. Next, we develop a wavelet-based perturbation method. Wavelets decompose the series using multiple window sizes and are also amenable to streaming estimation.

Algorithm 3 WAVELETPERTURB ($\{x_t\}_{t=1}^N$, σ)

$\{w_{\ell,t}\} \leftarrow \text{DWT}(\{x_t\})$ *//Wavelet transform (detail coeffs)*
$\mathcal{I}_\ell \leftarrow \{t : |w_{\ell,t}| \geq \sigma\}$ and $K_\ell \leftarrow |\mathcal{I}_\ell|$ **for each** level ℓ
1 $K \leftarrow \sum_\ell K_\ell$ *//No. of coeffs exceeding* σ
$\rho \leftarrow N/K$ *//Noise "density"*
for each detail $w_{\ell,t}$ **do**
2 **if** $t \in \mathcal{I}_\ell$ **then** *//*$|w_{\ell,t}| \geq \sigma$
3 $\omega'_{\ell,t} \leftarrow \text{GAUSSRND}\left(0, \sigma\sqrt{\rho}\right)$
else
4 $\omega'_{\ell,t} \leftarrow 0$
$\{n_t\} \leftarrow \text{INVDWT}(\{\omega'_{\ell,t}\})$ *//Inverse DWT (zero smooths)*
$y_t \leftarrow x_t + n_t$, **for all** t *//Published series*

Time/frequency perturbation. COMPRESSIBLEPERTURBATION using the wavelet transform is shown in algorithm WAVELETPERTURB. We denote $w_{\ell,t}$ and $\omega'_{\ell,t}$ the wavelet coefficients of the data x_t and of the perturbation n_t, respectively. WAVELETPERTURB follows the same general design of Section 4.1. In fact, wavelet coefficients are always real numbers and the procedure is simpler and more intuitive than FOURIERPERTURB. We allocate "noise units" only to those coefficients that exceed σ in absolute value. The perturbation is allocated equally among them, i.e., only in proportion to N/K, and not in proportion to per-coefficient or per-level energy. This simple choice makes the perturbation more resilient to true value leaks. Unlike FOURIERPERTURB, this does not sacrifice resilience to filtering attacks in practice, because time-localized phenomena do not lead to smearing of energy across wavelet coefficients.

Wavelets have been successful in a wide range of settings [29] and are more resilient to changes in series' characteristics. They decompose the series into translated and dilated, localized waves at multiple scales, which correspond to a particular time and frequency window. Short windows are employed for high frequencies (i.e., short periods) and longer windows for lower frequencies (i.e., longer periods)—see Figure 1.

The localization of bases in time has the additional desirable characteristic that, intuitively, each period is perturbed independently of others. For example, assume that by following an automobile, we learn its true speed over a period of 15 minutes. However, if periodic trends shorter than 15 minutes are perturbed independently, our collected true values can tell us nothing about the future perturbation at scales of up to 15 minutes. For periodic trends in the next scale of 30 minutes, perhaps the information learned will be useful for another 15 minutes but not longer, and so on for scales of 60 minutes, etc.

Finally, the DWT can be computed in $O(N)$ time, as opposed to $O(N \log N)$ time required by FFT [35]. Thus, even in a batch setting, it is computationally more efficient. Furthermore, wavelets can be estimated incrementally, using only $O(\log N)$ total space and $O(1)$ amortized time per value (see Section 2.1). From now on we focus on wavelets, since they have several desirable benefits.

4.3 Streaming perturbation

Our goal is to choose an effective perturbation that is hard to remove, but we want to perturb values as they arrive, before seeing the entire series, which grows indefinitely. Furthermore, we want to minimize or eliminate publishing delay. We explain this requirement next.

The Fourier transform needs, by definition, the entire series which is not possible in a streaming setting. One solution is to partition the series into fixed-size windows and apply Fourier on each of them. However, if we use a small window, we cannot capture trends with period larger then the window length. For example, if we use a 5-minute window to perturb driving speed, it is still possible to leverage hourly or daily driving patterns to reduce uncertainty. If we use a large window, then we may have to delay publishing the data until the window is filled up, so we can analyze it and perturb it. Alternatively, we could use the frequencies from the previous window to perturb the current one. However, if the window is large, it may not capture trends that have substantially changed in the new window. For example, a car might have been on the highway driving with a constant speed during the last hour, but has now entered a city and is in stop-and-go traffic. If we use a single one-hour window, the perturbation will follow the wrong trends.

Thus, the time/frequency decomposition of wavelets, which use multiple windows proportional to the period is desirable. In this case, we would use the information of the last, e.g., 5 minutes to decide if and how to perturb patterns up to 5 minutes long, during the next 5 minutes. However, we use the information of the last 10 minutes to make the same decision for smoother, longer patterns (up to 10 minutes) during the next 10 minutes, and so on. Steps (S1–2) in the general algorithm (see Section 4.1) need to be re-examined in a streaming context.

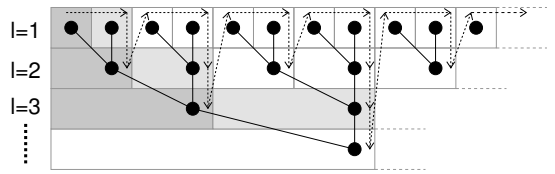


Figure 5: Order of incremental estimation: post-order traversal of wavelet coefficient tree.

Revisiting step (S1). If we want to make an exact decision whether to perturb a coefficient $w_{\ell,t}$ based on its actual magnitude (lines 2 and 3–4 in WAVELETPERTURB), then we have to wait time proportional to 2^ℓ for coefficients at level ℓ. In order to perform the inverse wavelet transform to publish a value, we need to wait for all coefficients across all levels that may affect its value. However, since the series size N grows indefinitely, so does the number of levels $L = O(\log N)$, which implies an indefinite publication delay.

We can impose a maximum delay (equivalently, a maximum level we are willing to wait for), but that is effectively the same as using a fixed-length window. Instead, we embed the noise into the *next* coefficient of the same level, i.e., we use $\omega'_{\ell,t+1}$ instead of $\omega'_{\ell,t}$ in lines 3 and 4. Said differently, the important coefficients in step (S1) are chosen based on the magnitude of *previous* coefficient in the same frequency band. For example, in Figure 5 the first coefficients of each level (darker shade) won't be perturbed, whereas the decision on whether to perturb the lightly shaded coefficients will be based upon the previous (darker) coefficient on the same level.

This simple one-step prediction is effective, since we are only interested in predicting whether a coefficient exceeds σ, rather than its exact value. More specifically, periodic trends result in uniformly large coefficients at the corre-

Dataset	Description
`Light`	Environmental sensor light intensities.
`Chlorine`	Chlorine concentration in drinkable water.
`SP500`	Standard & Poor's 500 stock index.

Table 2: Summary of datasets.

sponding wavelet level. Bursts also tend to affect more than one consecutive coefficient—if not, that is the only case we may "miss." However, such very short bursts generally occur at small scales and can often be ignored.

Revisiting step (S2). The number K of coefficients exceeding σ (lines 1 of WAVELETPERTURB) is not available at the time we need to make a decision about how to perturb the data. This quantity is needed to determine $\rho := N/K$. Our approach is to substitute these with incremental estimates. Therefore, whenever a new wavelet coefficient $w_{\ell,t}$ for any ℓ and t is produced, we update our estimate of ρ as follows:

$N \leftarrow N + 1$
if $|w_{\ell,t}| \geq \sigma$ **then**
  $K \leftarrow K + 1$
$\rho \leftarrow \lambda\rho + (1 - \lambda)(N/K)$

The order in which wavelet coefficients are incrementally computed is shown in Figure 5. This is the order in which the running counters N and K are updated. The decay factor $\lambda = 0.9$ is meant to prevent excessive fluctuations, particularly in the beginning of the series, when both N and K are relatively small. The inverse wavelet transform can be performed incrementally in a fashion similar to the forward transform.

LEMMA 1 (INCREMENTAL INVERSE DWT). *The inverse DWT can be computed incrementally in $O(1)$ time per value, using $O(\log N)$ space.*

PROOF. (Sketch) The forward transform can be performed incrementally because it is a post-order traversal of the coefficient tree (see Figure 5). The inverse transform is a pre-order traversal of the same tree. □

5. EXPERIMENTAL EVALUATION

Datasets. We evaluate our methods on several series from the UCR Time Series Data Mining Archive (TSDMA) [26], which range from environmental monitoring to financial data, with a wide variety of characteristics—see Table 2 for a summary of the datasets. All datasets are normalized to unit variance to standardize comparisons. The length of `Light` and `Chlorine` is 2048 and of `SP500` it is 16384—the choice of powers of two is without loss of generality, to simplify implementation. `Chlorine` is collected using a EPANET 2.0[2] that accurately simulates the hydraulic and chemical phenomena within drinking water distribution systems, given a realistic description of the network, demand patterns, pressures and flows at each node. The time series represents the chlorine concentration level at one junction in the network. The content of these measurements is concentrated on a few frequencies, which do not change over time, and the remaining frequencies have almost-zero (i.e., below σ) content across time . The `Light` dataset consists of light intensity measurements collected using a Berkeley Mote at a particular location in a lab. These measurements exhibit strong daily periodic trends. However, the trends' shape is non-sinusoidal, with many sharp edges and discontinuities. The `SP500` dataset contains the daily values of the Standards & Poors 500 stock market index, over a period of approximately 60 years. Even though the frequency content over such a long period is concentrated on few frequencies, there several above σ.

[2] `http://www.epa.gov/ORD/NRMRL/wswrd/epanet.html`

Setup. Our prototype is built in Matlab 7, running on a Pentium M 2GHz with 2GB memory. We use the Wavelet Toolbox for batch wavelet transforms, as well as for wavelet denoising (SureShrink [16], with DB-4 wavelets and the rigorous version of single-level noise estimation). We perform one experimental run for several different values of the discord σ, ranging from 5% to 40% of the total series standard deviation, at steps of 5%. For each experiment and for each method, we run ten perturbation trials. Each trial produces a different random perturbation. Our baseline method is white noise (i.i.d. Gaussian random perturbation) and we include (i) batch wavelet perturbation (DWT), (ii) its streaming version (Streaming DWT), and (iii) Fourier perturbation (FFT, comparing two noise allocation schemes—all figures are with per-band weighting as in FOURIERPERTURB, line 3, unless otherwise noted).

Uncertainty reduction. First we examine how much uncertainty can be removed by either a filtering or a true value leak attack on data perturbed with each method. In particular, we examine the fraction of uncertainty removed, i.e.,

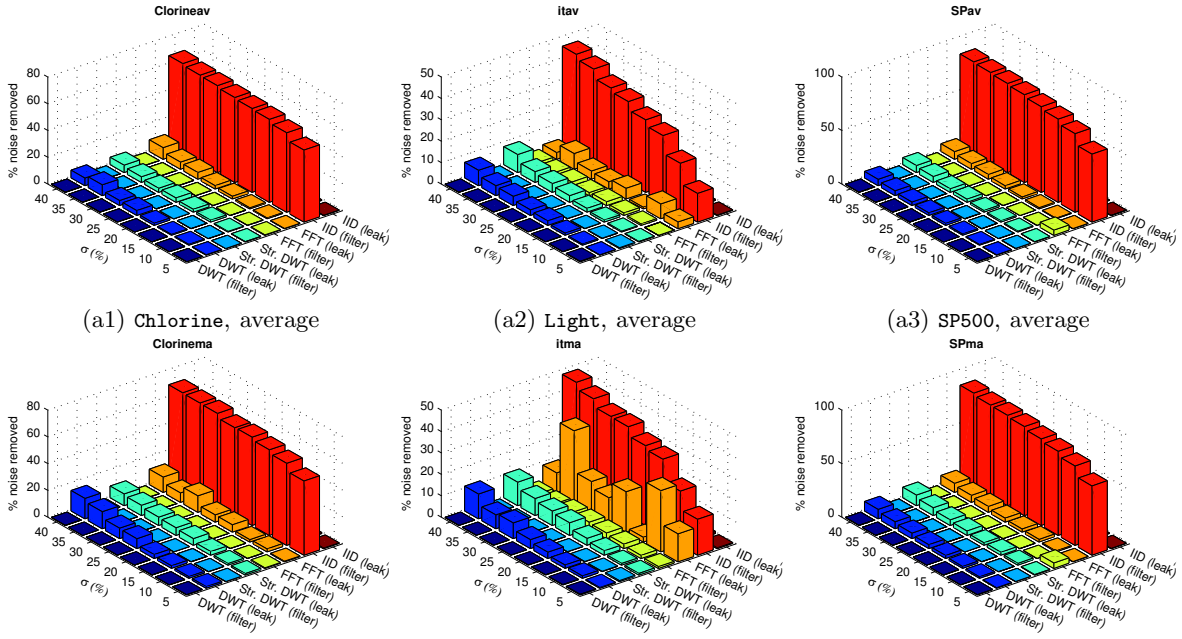
$$\tilde{f}(\sigma) := (\sigma - \tilde{\sigma})/\sigma \quad \text{and} \quad \hat{f}(\sigma) := (\sigma - \hat{\sigma})/\sigma,$$

for several different values of the discord σ (ranging from 5% to 40%). We estimate both the maximum (i.e., worst-case value) and average of $\tilde{f}$ and $\hat{f}$ across the ten perturbation trials in each experiment.

Figure 6 shows the percent of uncertainty removed by filtering, for each of the methods: (i) filtering and leak attack reduction for the batch wavelet method (first two bars from left, dark blue and blue); (ii) filtering and leak reduction for the streaming wavelet method (next two bars, light blue and cyan); (iii) filtering and leak reduction for the Fourier method (always batch, next two bars, light green and orange); and (iv) filtering and leak reduction for white noise (last two bars to the right, red and brown).

Note that, by construction, true value leaks do not help reconstruct values perturbed with white noise (even though not visible, all bars are zero). However, filtering can very successfully remove from 20–30% of the perturbation (for `Light`) up to almost 90% (for `SP500`). Thus, the need to take into account the characteristics of the series by using an appropriate, concise description is clear.

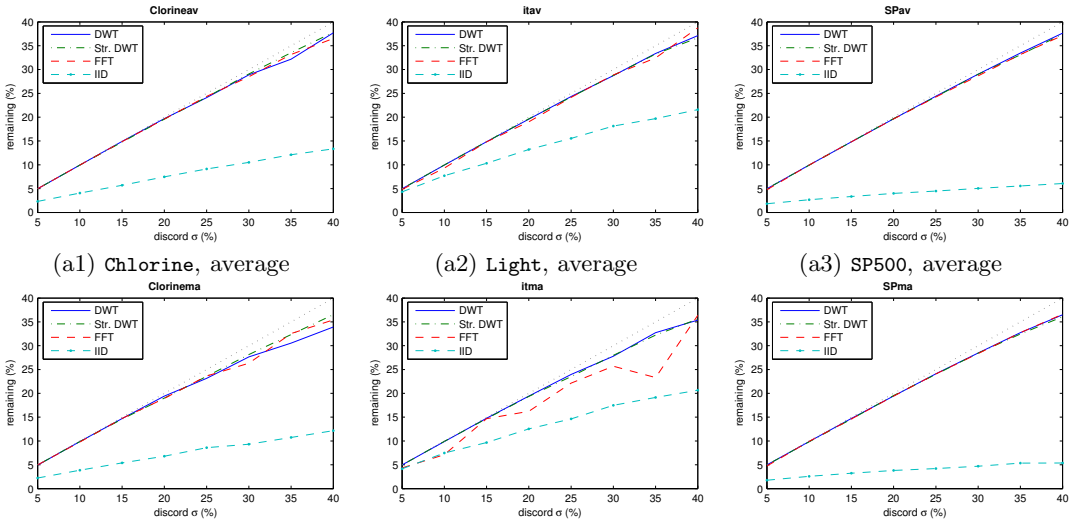
Having established this, we observe that all three of our proposed methods perform similarly. The streaming, wavelet perturbation method performs slightly better than the other two in some occasions. The reason is that it may initially overestimate the "density" $\rho = N/K$, particularly for series that have a large number of coefficients below the discord σ. This results in adding slightly more noise which, however, was never beyond 1-3% more than desired. Fourier perturbation may perform somewhat worse on certain data. However, as we shall see in the discussion later, it may exhibit



(a1) `Chlorine`, average (a2) `Light`, average (a3) `SP500`, average

(b1) `Chlorine`, maximum (b2) `Light`, maximum (b3) `SP500`, maximum

Figure 6: Percent uncertainty removed.



(a1) `Chlorine`, average (a2) `Light`, average (a3) `SP500`, average

(b1) `Chlorine`, maximum (b2) `Light`, maximum (b3) `SP500`, maximum

Figure 7: "True" remaining uncertainty after both attacks.

sensitivity to certain data characteristics and, in particular, the presence of sharp discontinuities. Overall, however, all three of our methods perform well on a wide variety of series and stay close to the optimal (the diagonal in Figure 7).

Finally, for wavelet-based perturbation, the average and maximum uncertainty reduction are closer to each other. In some cases the discrepancy between the two is larger for Fourier. Thus, even though all three methods have similar average behavior, wavelets have more consistent performance.

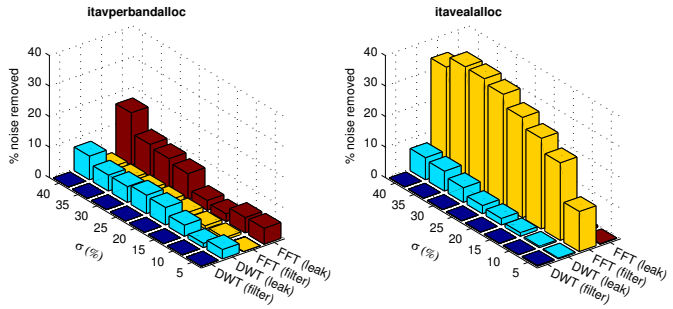
True uncertainty. In order to measure the uncertainty $u(\sigma)$ that remains after attempted attacks of any type, we also show the fraction of the perturbation that remains in the worst case (i.e., after the most successful of the two attacks). In particular,

$$u(\sigma) := \min\{\sigma(1 - \tilde{f}(\sigma)), \sigma(1 - \hat{f}(\sigma))\},$$

where $\tilde{f}(\sigma)$ and $\hat{f}(\sigma)$ are estimated over ten trials, as explained before.

Figure 7 shows the remaining uncertainty for all different methods. The axis diagonal, which represents the ideal case (i.e., remaining uncertainty equal to the discord) is plotted with a light gray, dashed line. The closer a method lies to this line, the better its overall performance.

First, it is clear in these plots as well that white noise performs very poorly, allowing a very large reduction of uncertainty. All three of our proposed methods perform similarly. In `Light`, which exhibits sharp discontinuities, the largest fraction of the energy concentrated on daily and half-daily periods. Most of the remaining energy is smeared across frequencies, due to the frequent jumps. Thus, this concentration of energy on a few frequencies allows somewhat larger uncertainty reduction via leaks, due to the regularity of the perturbation.



(a) `Light`, per-band (b) `Light`, equal

Figure 8: Noise allocation for `Light`—see Figure 10.

Noise unit allocation. In this section we compare (i) noise allocation in proportion to frequency band energy, and (ii) equal allocation. We perform the comparison for both Fourier and wavelet perturbation. By default, FourierPerturb as presented uses per-band allocation. Changing line 3 to $\rho_k \leftarrow N/K$ and ignoring the p_k^2/P factor is the modification necessary to do equal allocation. On the other hand, the default for WaveletPerturb is equal allocation. To change it into per-band allocation, we first estimate the level energy, $p_\ell \leftarrow \sum_{t \in \mathcal{I}_\ell} w_{\ell,t}^2$, and the total energy, $P \leftarrow \sum_\ell p_\ell$. From these we estimate $\rho_\ell \leftarrow (p_\ell/P) \cdot (N/K_\ell)$ and then use ρ_ℓ instead of ρ in line 3 of WaveletPerturb. Figures 8 and 10 show the comparison of allocation schemes on the two most representative datasets. The evaluation justifies the default allocation schemes for each algorithm and shows they are in line with our design principle: make the simplest choice that is resilient to filtering attacks, while also keeping true value leak attacks in check.

On `Chlorine`, which consists mainly of a few, unchanging frequencies, Fourier perturbation performs similarly under both allocation schemes—see Figure 10(a1–2). However, `Light` has a dominant daily trend but also a large number of discontinuities that are smeared across frequencies. Thus, with equal allocation, Fourier "wastes" too much noise units on those frequencies and this can be effectively detected and removed by filtering—see Figure 10(b2). With per-frequency allocation, Fourier performs acceptably, on average. However, its performance is less stable than the wavelet perturbation, as is evident in Figure 7(b2) which shows worst-case measurements. Overall, wavelets perform at least as well as Fourier, in a more consistent fashion due to their time-localization properties.

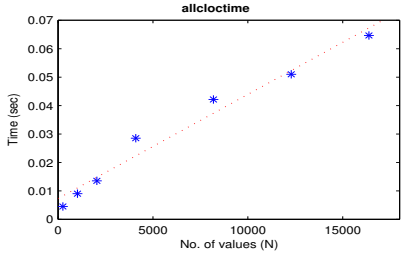


Figure 9: Scalability w.r.t. number of values.

Scalability. Figure 9 verifies that the wavelet perturbation scheme scales linearly with respect to time series stream size. Even though our prototype is implemented in Matlab, the average processing time per value is approximately 35μsec, when the stream size is large enough to compensate for initialization overheads.

Summary. Our experimental evaluation clearly shows that white noise is insufficient for effective perturbation, particularly under the filtering attacks which are our primary concern. Thus, it is necessary to take the structure of the data into consideration, by seeking an effective, concise description of real data. We propose three methods which perform similarly on average. More specifically, for series with stable spectral content limited to a small number of frequencies, all methods perform similarly. If the spectral content changes, then Fourier performs well on average but is less consistent overall. Our perturbation method that uses time/frequency wavelet analysis performs as well as or better than Fourier and is also suitable for streaming estimation.

6. RELATED WORK

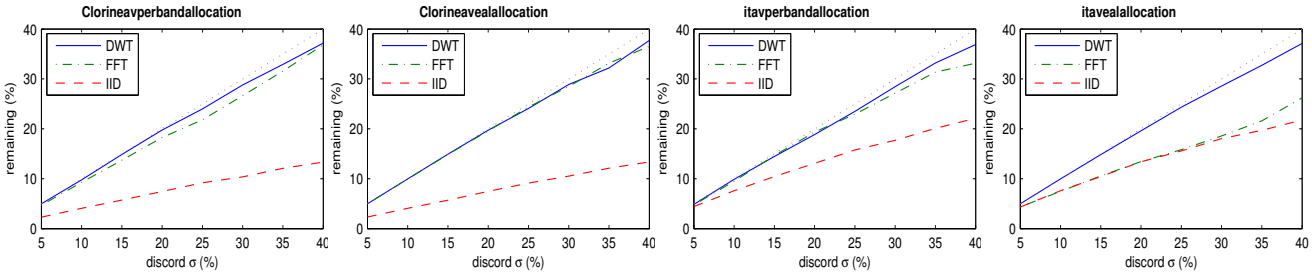
Privacy preserving data mining was first proposed in [4] and, simultaneously, in [31]. Various privacy techniques have been proposed since, which apply to the traditional relational model can be broadly classified into methods based on secure multiparty computation (SMC) [31, 39] and into methods based on partial information hiding. The latter can be further subdivided into data perturbation [4, 3, 18, 24, 23, 32, 10, 28] and k-anonymity [38, 2, 33, 27, 41, 6] methods.

Wavelets have been succesfully applied on a wide variety of data mining or data summarization applications [22, 34, 8, 30, 21, 25]—see [29] for a comprehensive survey. Thus, the ability of wavelets to succinctly describe data in several application domains and their effectiveness in practice as a general analysis tool is well-established.

The work of [24] and [23] also use a linear transformation to detect patterns in the data and, consequently, leverage them to add noise that is hard to remove. In particular, they use principal components analysis (PCA) on a static, relational table with n numerical attributes and obtain a rank-k, $k < n$, approximation of its covariance matrix. This covariance is used to perturb each tuple with noise that has the same correlation among attributes. Even though [23] mentions the potential of using the auto-covariance for perturbation, it does not propose a solution. The work in [28] has extended the discussion into streaming data to deal with dynamic correlation and auto-correlation exhibited by multiple data streams. However, it does not consider connections between privacy and compressibility, nor does it take into account true value leaks. Furthermore, the perturbation and reconstruction techniques in [28] do not employ multi-scale analysis, but rather rely on simple fixed-window techniques.

Among other work on privacy, ℓ-diversity [33] and personalized privacy [41] are similar in spirit to our work, in the sense that they point out potential privacy breaches that exploit certain structure of the data. For example, among several aspects studied, both [33] and [41] point out that if all k tuples with the same generalized quasi-identifier are associated with the same sensitive value, the privacy of these tuples is compromised. This *homogeneity attack* [33] may be viewed as a very specific case of compressibility: the distribution of the sensitive values for this set of k tuples has zero variance. Both [33] and [41] make a number of important observations. However, none of them addresses the challenges posed by the aspect of time.

More generally, recent work has begun to realize that, when the entire collection of values is considered as a sin-



(a1) `Chlorine`, per-band (a2) `Chlorine`, equal (b1) `Light`, per-band (b2) `Light`, equal

Figure 10: Noise allocation: per-band weighting versus equal allocation.

gle data object, any regularity or structure that is present may lead to potential privacy breaches. In this paper we address these challenges for time series data. We consider filtering attacks as well as true value leaks and we propose a practical, effective method that automatically strikes an appropriate balance between the two. When the number of attributes becomes very high, the well-known *dimensionality curse* [1] poses additional challenges for privacy. In our context, a univariate time series is a *single* point in a very high-dimensional space (and in a streaming setting, the dimension increases indefinitely) and the challenges faced are even harder.

A different model of *interactive privacy*, similar in spirit to that of [12], has been proposed in a series of papers, including [7, 19]. In contrast to the non-interactive model, where the dataset itself is perturbed, the interactive model adds a perturbation to the outputs of arbitrary aggregate queries over the dataset. This thread of work provides an important theoretical analysis on privacy, under certain assumptions. First, the weak form of independence assumption made by [7] may be reasonable for datasets with millions or billions of tuples (e.g., a global census database, where we may assume that an adversary's belief about my age is largely independent of their belief about my parent's age). However, in ordered data, such as time series, this assumption may be problematic.

The work on watermarking numeric data streams [37] faced similar challenges as we do, i.e., how to inject digital watermarks into continuously arriving data tuples in a streaming fashion, while maintaining high resilience under various attacks. However, watermarking has a different goal and considers different types of attacks. In particular, it is generally assumed that all of the original data (or, at least, a large fraction thereof) are available, so the "noise" component of watermarked data can be exactly recovered and subsequently analyzed to verify the presence of a watermark.

Finally, in the field of statistical signal processing, the notion of *compressed sensing* [15, 9] has recently emerged, building upon previous fundamental results in signal estimation and recovery [17, 16], and it is partly related to our work. Traditional signal recovery assumes that a time series has a concise representation in some space and examines the relationship between reconstruction accuracy versus number of observed samples. Compressed sensing generalized these results from individual sample observations to arbitrary observed functionals on the signal (such as the total energy, or the averages of neighboring values, etc). However, this work does not address issues that arise in partial information hiding.

7. DISCUSSION

We consider two potential breaches, with different assumptions about background knowledge, each of which captures situations that may arise in practice. In particular, the first set of assumptions is most common in signal estimation and recovery applications, and essentially imposes either "global smoothness" constraints (via the background assumption of compact representation in the frequency domain) or "local smoothness" constraints (via the assumption of compact representation in the wavelet domain). The second set of assumptions deals with true value leaks and efforts for linear estimation of other true values, based on those that were leaked. In this case we take the worst-case view that an arbitrary number of true values may be leaked. Our leak uncertainty is a statistical measure of the maximum possible loss of privacy under these assumptions.

In the present paper, we focus on practical aspects and we extensively evaluate our methods under both attack models, demonstrating that both are important in practice. In addition, our experimental evaluation presents both average-case results, in Figures 6(a1–3) and Figures 7(a1–3), as well as worst-case results, in Figures 6(b1–3) and Figures 7(b1–3). Average-case results are important to judge the overall behavior of a technique, but worst-case results more accurately reflect what may happen on a *particular* publication instance of one dataset. Perhaps because of the challenges in proving meaningful statements in the latter case, the worst case has been largely overlooked. Our evaluation demonstrates the practical robustness of our techniques on a number of datasets.

In general, filtering attacks based on background knowledge about the "smoothness" properties of the data are the most important in practice. This is clear in all cases of Figure 6, where between 50–90% of an i.i.d. perturbation may be removed. Among the two classes of smoothness assumptions an adversary may make (global, via Fourier, or localized at multiple scales, via wavelets), wavelet-based techniques perform at least as well as Fourier-based techniques. Only for `Chlorine` with smaller perturbation magnitudes, the Fourier-based technique performs slightly better. However, Fourier-based global analysis is not suitable for streaming publication of the data. Furthermore, for datasets with both strong periodic components as well as local discontinuities, such as `Light`, Fourier-based perturbation tends to concentrate on a few frequencies, resulting in regularities that may be exploited by true value leaks, as illustrated in Figure 6(b2).

In summary, we focus on two novel aspects of partial information hiding and privacy. We consider two real-world scenarios, design robust and practical techniques which are also suitable for a streaming setting. For each aspect, we evaluate our techniques extensively on real data. A challeng-

ing and interesting problem for future research is to come up with a unified attack model, which combines background knowledge of both types (i.e., knowledge about some true values and also about the shape of the series).

8. CONCLUSION

From the first, seminal work on privacy preservation via partial data hiding [4, 38] until today, there is an increasing realization that subtle potential privacy breaches may arise when any regularity or structure is present in the entire collection of values considered as a single, complex data object [24, 23, 41, 33]. In this paper we address these challenges for time series data. We consider true value leaks as well as filtering attempts, study the fundamental trade-offs involved in addressing both and propose a practical, effective method that is based on the wavelet transform, which has been widely successful in capturing the essential characteristics of data [29].

Future work includes investigating theoretical guarantees by treating the problem as a form of signal recovery, consolidating the two attack models and extending our techniques beyond time series data.

Acknowledgements. This work was partially supported by NSF grant IIS-0133825.

9. REFERENCES

[1] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, 2005.

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

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

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

[5] D. Automotive. CarChip. `http://www.carchip.com/`.

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

[7] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *PODS*, 2005.

[8] A. Bulut and A. Singh. SWAT: Hierarchical stream summarization in large networks. In *ICDE*, 2003.

[9] E. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE TOIT*, 52(2), 2006.

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

[11] M. H. DeGroot and M. J. Schervish. *Probability and Statistics*. Addison Wesley, 3rd ed. edition, 2002.

[12] D. E. Denning. Secure statistical databases with random sample queries. *TODS*, 5(3), 1980.

[13] D. L. Donoho. Progress in wavelet analysis and WVD: A ten minute tour. In Y. Meyer and S. Rogues, editors, *Progress in Wavelet Analysis and Applications*. Frontiéres, 1993.

[14] D. L. Donoho. De-noising via soft thresholding. *IEEE TOIT*, 41(3), 1995.

[15] D. L. Donoho. Compressed sensing. *IEEE TOIT*, 52(4), 2006.

[16] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *J. Am. Stat. Soc.*, 90, 1995.

[17] D. L. Donoho and P. B. Stark. Uncertainty principles and signal recovery. *SIAM SIAP*, 49(3), 1989.

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

[19] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

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

[21] M. Garofalakis and P. B. Gibbons. Wavelet synopses with error guarantees. In *SIGMOD*, 2002.

[22] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, 2001.

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

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

[25] P. Karras and N. Mamoulis. One-pass wavelet synopses for maximum-error metrics. In *VLDB*, 2005.

[26] E. Keogh and T. Folias. UCR time series data mining archive. `http://www.cs.ucr.edu/~eamonn/TSDMA/`.

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

[28] F. Li, J. Sun, S. Papadimitriou, G. Mihaila, and I. Stanoi. Hiding in the crowd: Privacy preservation on evovling streams through correlation tracking. In *ICDE*, 2007.

[29] T. Li, Q. Li, S. Zhu, and M. Ogihara. A survey on wavelet applications in data mining. *SIGKDD Explorations*, 4(2), 2002.

[30] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT*, 2004.

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

[32] K. Liu, J. Ryan, and H. Kargupta. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE TKDE*, 18(1), 2006.

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

[34] S. Papadimitriou, A. Brockwell, and C. Faloutsos. AWSOM: Adaptive, hands-off stream mining. In *VLDB*, 2003.

[35] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge Univ. Press, 2000.

[36] W. P. Schiefele and P. K. Chan. SensorMiner: Tool kit for anomaly detection in physical time series. Technical report, `http://www.interfacecontrol.com/`, 2006.

[37] R. Sion, M. Atallah, and S. Prabhakar. Rights protection for discrete numeric streams. *IEEE TKDE*, 18(5), 2006.

[38] L. Sweeney. k-anonymity: A model for protecting privacy. *IJUFKS*, 10(5), 2002.

[39] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, 2002.

[40] M. Vlachos, P. S. Yu, V. Castelli, and C. Meek. Structural periodic measures for time-series data. *DMKD*, 12(1), 2006.

[41] X. Xiao and Y. Tao. Personalized privacy preservation. In *SIGMOD*, 2006.

[42] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD*, 2002.

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