International Journal of Innovative Computing, Information and Control
Volume **8**, Number **5(B)**, May **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 3619–3638

PUBLISHING SENSITIVE TIME-SERIES DATA UNDER PRESERVATION OF PRIVACY AND DISTANCE ORDERS

Mi-Jung Choi, Hea-Suk Kim and Yang-Sae Moon*

Department of Computer Science
Kangwon National University
192-1 Hyoja2-Dong, Chunchon, Kangwon 200-701, Republic of Korea
{ mjchoi; hskim }@kangwon.ac.kr; *Corresponding author: ysmoon@kangwon.ac.kr

Received December 2010; revised September 2011

Abstract. *In this paper, we address the problem of preserving mining accuracy as well as privacy in publishing sensitive time-series data. For example, people with heart disease do not want to disclose their ECG time-series, but they still allow mining some accurate patterns from their time-series. Our privacy model assumes that (1) data sources publish their time-series independently, and (2) all information used in publishing time-series can be publicly revealed. Based on this model, we introduce three assumptions: full disclosure, equi-uncertainty, and independency. We also derive two requirements: uncertainty preservation and distance order preservation. We show that only randomization methods satisfy all three assumptions, but even those methods do not satisfy both the requirements. Thus, we discuss the randomization-based solutions that satisfy all assumptions and requirements. For this purpose, we present a novel notion of the noise averaging effect of piecewise aggregate approximation (PAA), which is derived from a simple intuition that the summation of random noise converges to 0. This noise averaging effect can alleviate the problem of destroying distance orders in randomly perturbed time-series. Based on the noise averaging effect, we first propose two naive solutions that use the random data perturbation in publishing time-series while exploiting the PAA distance in computing distances. There is, however, a tradeoff between these two solutions with respect to uncertainty and distance orders. We thus propose three more advanced solutions that take advantages of both naive solutions. Experimental results show that our advanced solutions are superior to the naive solutions in the preservation of uncertainty, distance orders, and clustering accuracy.*

Keywords: Time-series data, Privacy preservation, Data mining, Clustering, Distance orders

1. Introduction.

In recent years privacy preserving data mining (PPDM) [2, 25] has been investigated extensively motivated by the current practice by private and public organizations of collecting large amounts of often sensitive data. The aim of PPDM algorithms is to extract relevant knowledge from a large amount of data while protecting at the same time sensitive information [4]. PPDM algorithms can be roughly classified into four categories [1]: random data perturbation, k-anonymization, distributed privacy preservation, and privacy preservation of mining results.

In this paper[1] we address the problem of preserving both privacy and mining accuracy in publishing sensitive time-series data. Time-series data have been widely used in many applications [19, 21], and data mining on time-series data has been actively studied [9, 20, 29, 41, 43]. Figure 1 shows the data flow model that we assume. We use the centralized model [24, 30, 33] where multiple independent data sources provide their time-series to

[1]The preliminary version of this paper was published in *Proc. of the 21st Int'l Conf. on Database and Expert Systems Applications*, Bilbao, Spain, pp.17-31, 2010.

third parties. In this model, however, data sources do not trust third parties, so they do not wish to provide their original time-series, but they could still provide appropriately distorted time-series to get meaningful mining results. Typical examples of requiring both privacy and mining accuracy are as follows.

- Stock brokers do not wish to reveal their own buying or selling patterns, but they still would like to contribute to categorization of stock brokers or mining of general buying or selling patterns.
- Patients with heart disease do not want to disclose their private electrocardiogram (ECG) data, but they still would like to contribute their data to research and allow researchers to mine general or specific ECG patterns.
- Drivers do not wish to disclose their exact speed recorded in the vehicle monitoring system, but they still would like to contribute their data for city planning purposes and thus allow planners to mine general driving patterns [33].

Thus, we can say that accuracy preservation of mining results is as important as privacy preservation of sensitive time-series data.

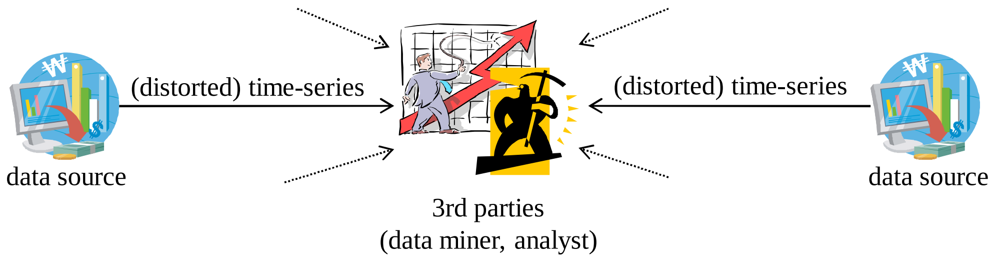


FIGURE 1. A data flow model of independent data sources and third parties

To address both privacy and mining accuracy, we first setup a privacy model that addresses the underlying assumptions and requirements. Our model has three assumptions: *full disclosure*, *equi-uncertainty*, and *independency*. Full disclosure means that all information used in distorting and publishing time-series can be revealed to third parties or attackers. Equi-uncertainty means that each of distorted time-series has the same amount of uncertainty, which represents the degree of difference between original and distorted time-series [33]. Independency means that each time-series can be independently distorted without considering other time-series. To meet our main goal of preserving privacy and mining accuracy, we also derive two preservation requirements: *uncertainty* and *distance order*. Uncertainty preservation means that original time-series cannot be reconstructed from the published, distorted time-series. Distance order preservation means that relative distance orders among time-series must be preserved after the distortion. According to our analysis in Section 2, only the random perturbation methods [1, 2, 33] satisfy all three assumptions, but even these randomization methods do not satisfy both the requirements. Therefore, in this paper we discuss the randomization-based solutions that satisfy all assumptions and requirements of the privacy model.

For the purpose of preserving distance orders, we present a novel notion of the *noise averaging effect* of piecewise aggregate approximation (PAA) [17, 19]. This notion is derived from a simple intuition that the summation of random noise eventually converges to 0. We show that the noise averaging effect can alleviate the problem of distorting distance orders in randomly perturbed time-series. PAA is a well-known lower-dimensional transformation of similarity search in time-series databases [9, 17, 21]. PAA extracts a fixed number of averages from a long time-series and uses those averages to compute the distance. Since PAA uses averages in computing distances, it naturally exploits the noise averaging effect on the distorted/published time-series. To exploit this noise averaging effect, we use *PAA distances* in computing distances of the distorted time-series.

In this paper, we propose two naive and three advanced solutions based on the random perturbation and the noise averaging effect. Our first solution simply adopts the random perturbation in publishing time-series, but it uses PAA distances to preserve distance orders by exploiting the noise averaging effect. The simple random perturbation, however, can be attacked by the wavelet filter [7]. We thus propose another solution that uses the recent wavelet-based perturbation method [33] which is secure against the wavelet filtering attack. These two solutions, however, are in a tradeoff relationship with respect to uncertainty and distance orders. We thus propose three more advanced solutions that take advantages of both naive solutions. Our advanced solutions can be seen as an engineering approach that preserves uncertainty and distance orders as much as possible through the recent wavelet-based perturbation and the noise averaging effect of PAA.

Our solutions provide a very practical approach to publish time-series data which guarantees mining accuracy as well as privacy. We do not use any complicated cryptography techniques or SMC protocols [6, 14, 35, 38], but simply adopt an intuitive notion of the noise averaging effect and the wavelet-based secure perturbation method. The contributions of the paper can be summarized as follows. (1) We present a privacy model characterized by assumptions required in a centralized data flow model and requirements for preserving privacy and mining accuracy. Under these assumptions and requirements, we analyze advantages and disadvantages of the previous solutions. (2) We present a novel notion of the noise averaging effect and show its effectiveness in computing distances of the perturbed time-series. (3) We propose two naive solutions that exploit the noise averaging effect and introduce three more advanced solutions that represent a compromise in the tradeoff relationships between those naive solutions. (4) Through extensive experiments we showcase the superiority of our advanced solutions.

2. Proposed Privacy Model.

Our privacy model uses the Euclidean distance [9, 17, 28, 29, 33] as the metric of (dis)similarity between time-series since it has been widely used in many clustering or classification algorithms [15, 27, 29]. Given two time-series $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, the Euclidean distance $D(X, Y)$ is defined as $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$. Our model assumes that each data source first distorts its time-series X to $X^d = \{x_1^d, x_2^d, \ldots, x_n^d\}$[2] independently, and then publishes the distorted time-series X^d. Attackers may try to recover the original time-series X from the published time-series X^d for the malicious purpose of obtaining privacy-sensitive data. We denote by $X^r = \{x_1^r, x_2^r, \ldots, x_n^r\}$ the recovered time-series recovered from X^d.

Under the data flow model in Figure 1, we present three assumptions of the privacy model. These assumptions have been separately introduced in many previous papers with some different terminologies and different meanings [2, 12, 24, 31, 33]. In this paper, we summarize these assumptions using the following three terminologies.

- *Full disclosure*: We assume that all information used in distorting time-series can be revealed to third parties or attackers. It means that distortion techniques and related parameters are published.
- *Equi-uncertainty*: We assume that every published time-series has the same amount of distorted information. In other words, all distorted time-series have the same amount of uncertainty. Here, the uncertainty represents the degree of difference between original and distorted time-series. (We will formally define it below.)

[2]The length of X^d can differ from that of X. For simplicity, however, we assume that they have the same length without loss of generality.

- *Independency*: We assume that each time-series can be independently distorted without considering other time-series. This is because, as shown in Figure 1, time-series data are scattered in multiple independent data sources, and those sources do not interact with each other. Thus, each data source independently distorts its time-series without interacting with other data sources or third parties.

The major goal of our privacy model is to preserve privacy and at the same time assure mining accuracy. To discuss about the privacy preservation first, we use the *uncertainty* [33], also known as the mean square error or discrepancy [12, 16, 24], as the metric of privacy. Uncertainty between two time-series is defined as the standard deviation of differences of their corresponding entries [33]. More precisely, uncertainty between an original time-series X and its distorted time-series X^d is defined as $u(X, X^d) = \sum_{i=1}^{n} |x_i - x_i^d|^2$; uncertainty between X and its recovered time-series X^r is defined as $u(X, X^r) = \sum_{i=1}^{n} |x_i - x_i^r|^2$. The former uncertainty $u(X, X^d)$ can be seen as the noise amount enforced by the data source of X; the latter uncertainty $u(X, X^r)$ the noise amount remaining after the attack. Thus, the smaller difference between $u(X, X^d)$ and $u(X, X^r)$ the better privacy preservation is [33]. Based on this observation, we formally define the uncertainty preservation as follows.

Definition 2.1. *Given an original time-series X, its distorted time-series X^d, and its recovered time-series X^r, we say that the uncertainty of X^d is preserved if $|u(X, X^d) - u(X, X^r)|$ is less than the user-specified threshold.*

Using Definition 2.1, we now derive the privacy requirement.

Requirement 2.1. *Uncertainty of the published time-series needs to be preserved to assure that original time-series cannot be reconstructed from the published ones.*

We next discuss about the mining accuracy preservation. Different mining techniques use different accuracy measures, and we thus introduce a notion of *distance orders* as a general measure of mining accuracy for time-series data. Distance orders represent the relative orders among distances between time-series. In general, preserving both the absolute distances between time-series and their privacy is difficult. However, preserving the relative orders among distances are enough for providing higher accuracy in most mining algorithms [5, 13]. Based on this observation, we use the notion of distance order preservation for assuring mining accuracy.

Definition 2.2. *Let O, A, and B be time-series, and O^d, A^d, and B^d be the corresponding distorted time-series, respectively. We say that the distance order among O, A, and B is preserved if one of the following implications holds.*

$$D(O, A) \le D(O, B) \implies D(O^d, A^d) \le D(O^d, B^d),$$
$$D(O, A) \ge D(O, B) \implies D(O^d, A^d) \ge D(O^d, B^d)$$

In other words, we say that the distance order is preserved if their relative order of distances is not changed.

Using Definition 2.2, we now derive the mining accuracy requirement of the privacy model.

Requirement 2.2. *Distance orders among time-series need to be preserved for the purpose of providing high quality of mining results.*

Table 1 reports a summary comparison of related work with respect to the assumptions and requirements in our privacy model. In our comparison, we only considered approaches that can be applied to time-series data. (For the detailed analysis of existing solutions, refer to Section 6.) As shown in Table 1, except for random data perturbation solutions in [2, 33], all privacy preserving solutions do not satisfy one or more assumptions. This is because most solutions assume some constraints on distortion techniques or underlined environments. The randomization methods, which distort time-series by adding random noise locally and independently, satisfy all three assumptions, but even those methods do not satisfy both the requirements. Therefore, in this paper we aim at finding the best solution that solves the following problem.

Problem Statement In publishing time-series data, find a solution that satisfies the three assumptions of full disclosure, equi-uncertainty, and independency and the two requirements of privacy preservation and distance order preservation.

Table 1. Comparison of existing solutions

Previous Solutions	Assumptions			Requirements	
	Full disclosure	Equi-uncertainty	Independency	Privacy preservation*	Distance order preservation
[2]	○	○	○	×	×
[30]	×	○	×	×	○
[31]	○	×	○	○	×
[12, 16]	×	○	×	×	○
[29]	×	×	×	×	○
[24]	○	○	×	○	×
[33]	○	○	○	○	×

* Under assumption of full disclosure (i.e., all information is revealed.)

3. **PAA-based Intuitive Solutions.**

3.1. **Noise averaging effect of PAA.**

Random data perturbation (*randomization* in short) satisfies all three assumptions. Randomization generates white noise based on uniform or Gaussian distributions and adds that noise to original time-series. More formally, for an original time-series X, randomization generates a noise time-series $N = \{n_1, n_2, \ldots, n_n\}$ with mean 0 and standard deviation σ and constructs a distorted time-series as $X^d = \{x_1 + n_1, x_2 + n_2, \ldots, x_n + n_n\}$. Obviously, the standard deviation σ equals to $u(X, X^d)$, the uncertainty between original and published time-series. Full disclosure, the first assumption, is satisfied in randomization since we do not hide any information including the mean and standard deviation. Equi-uncertainty, the second assumption, is also satisfied since we use the same standard deviation for all time-series. Independency, the third assumption, is trivially satisfied since each time-series reflects its own noise time-series. Thus, the only thing we need to consider in randomization is whether it satisfies two requirements or not.

Randomization is known to well preserve privacy, but not mining accuracy [29]. White noise makes it difficult to disclose the exact value of each entry, but at the same time it destroys distance orders among time-series. As we increase the amount of noise for better privacy, mining accuracy decreases rapidly [29]. To solve this problem, we propose a novel notion of the *noise averaging effect*, which is derived from a simple intuition that the summation of white noise eventually converges to 0 since the mean of white noise is 0. To exploit the noise averaging effect in computing the distance between two distorted (i.e., noise-embedded) time-series, we simply use their averages of multiple entries instead of individual entries.

In this paper, we use the noise averaging effect of PAA [17, 19]. PAA transforms a time-series X $(=\{x_1,\dots,x_n\})$ and its distorted time-series X^d $(=\{x_1^d,\dots,x_n^d\})$ to their averaged sequences $\bar{X}$ $(=\{\bar{x}_1,\dots,\bar{x}_f\})$ and $\bar{X}^d$ $(=\{\bar{x}_1^d,\dots,\bar{x}_f^d\})$, respectively, as follows:

$$\bar{x}_i = \frac{f}{n}\sum_{j=\frac{n}{f}(i-1)+1}^{\frac{n}{f}i} x_j, \quad \bar{x}_i^d = \frac{f}{n}\sum_{j=\frac{n}{f}(i-1)+1}^{\frac{n}{f}i} x_j^d \tag{1}$$

As shown in Equation (1), PAA gets an average from each interval, and we thus naturally exploit the noise averaging effect if we use PAA in computing the distance. For this purpose, we define the *PAA distance* between two distorted time-series as follows.

Definition 3.1. *Given two distorted time-series X^d and Y^d, their PAA distance, denoted as $PD(X^d,Y^d)$, is defined as follows:*

$$PD(X^d,Y^d) = D(\bar{X}^d,\bar{Y}^d) = \sqrt{\sum_{i=1}^{f}(\bar{x}_i^d-\bar{y}_i^d)^2}. \tag{2}$$

Using PAA distances instead of original distances we may alleviate the problem of destroying distance orders in random data perturbation. Example 3.1 shows a desirable case that the noise averaging effect solves the problem of changing distance orders.

Example 3.1. *In Figure 2, we have three original time-series O, A, and B, and $D(O,A)$ is greater than $D(O,B)$. For the privacy preserving publication, we add 20% of white noise to each time-series. After this randomization, however, their distance order is changed, i.e., $D(O^d,A^d)$ becomes less than $D(O^d,B^d)$. This change of distance orders may frequently occur by white noise, and it severely lowers mining accuracy. To exploit the noise averaging effect of PAA, we now get PAA distances $PD(O^d,A^d)$ and $PD(O^d,B^d)$ of the distorted time-series. We note that $PD(O^d,A^d)$ is greater than $PD(O^d,B^d)$, and it means that the distance order is preserved if we use PAA distances.*

In the following two subsections, we propose PAA distance-based randomization methods to get the higher distance order preservation.

3.2. **RAND: random data perturbation and PAA distances.** Our first solution uses the randomization without any modification in distorting time-series, but it uses the PAA distance in comparing distance orders. We call this first solution *RAND*. Algorithm 1 shows the distortion procedure of RAND, which is very simple and self-explained. In Line 1, GaussRand $(0,\sigma)$ is a function of generating a random number based on the Gaussian

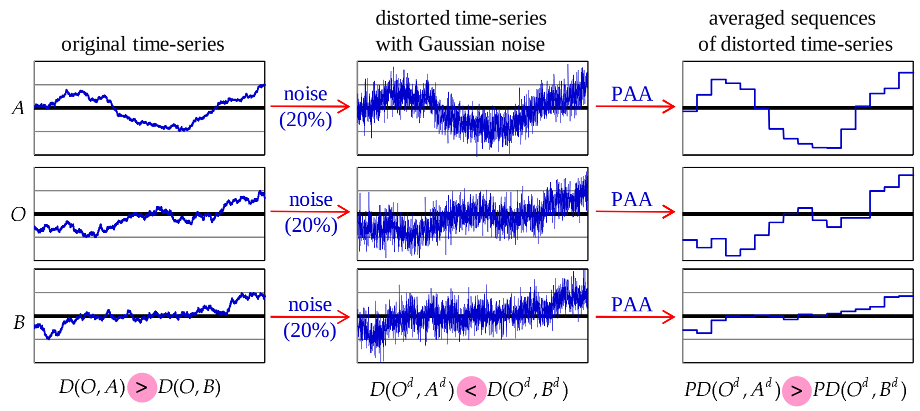


FIGURE 2. Noise averaging effect of PAA

distribution of mean 0 and standard deviation σ. According to the data flow model of Figure 1, each data source publishes its time-series using RAND, and third parties mine the meaningful patterns using the PAA distance to get the higher mining accuracy.

Algorithm 1 RAND ($X = \{x_1, \ldots, x_n\}$, σ)

```
1: Generate a noise time-series $N$ where $n_i$ := GaussRand $(0, \sigma)$;
2: Make a distorted time-series $X^d$ from $X$ and $N$; // $x_i^d := x_i + n_i$
3: Publish the distorted time-series $X^d$ to third parties;
```

Figure 3 shows the preliminary experimental result on distance order preservation of RAND. In this experiment, we used a set of random walk time-series of length 2048 and extracted 32 PAA features from each time-series. We measured how much percent of distance orders were preserved. As shown in Figure 3, the PAA distance in RAND well preserves distance orders; i.e., it improves the percentage of distance order preservation, especially in higher uncertainty. We note that the PAA distance makes the negative effect when the uncertainty is low. This is because PAA itself may change distance orders. In most cases, however, the PAA distance exploits the positive effect due to the noise averaging effect.

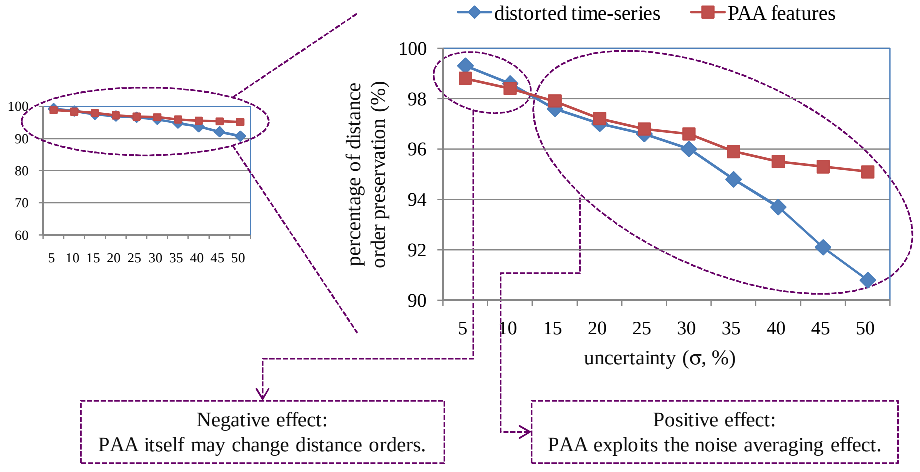


Figure 3. Preliminary experimental result on distance order preservation of RAND

RAND, however, has a critical problem in preserving privacy. The problem is that white noise can be easily removed by the wavelet filter [7, 33]. Example 3.2 shows how we can remove the white noise from the distorted time-series.

Example 3.2. *In Figure 4, we first distort an original time-series X to X^d by adding 20% of white noise. We then perform the discrete wavelet transform (DWT)*[3] *on X^d and get wavelet coefficients from X^d. Through DWT, most energy is concentrated in the first few coefficients. We next filter less energy coefficients; more specifically, we filter the coefficients whose absolute values are less than σ* [33][4]. *We finally recover the time-series through the inverse DWT. As a result, the recovered time-series X^r has only 4.8% of white noise. It means that the uncertainty is significantly reduced from 20% ($= u(X, X^d)$) to 4.8% ($= u(X, X^r)$) by the wavelet filter.*

[3]We can use Daubechies-L and Haar wavelets [34] for the de-noising filters [33]. For simplicity, we use the Haar wavelet in this paper.

[4]Refer to SureShrink [7] for the detailed de-noising method.

Likewise, the uncertainty, i.e., the white noise can be removed by the wavelet filter, and we can say that privacy is not well preserved in RAND.

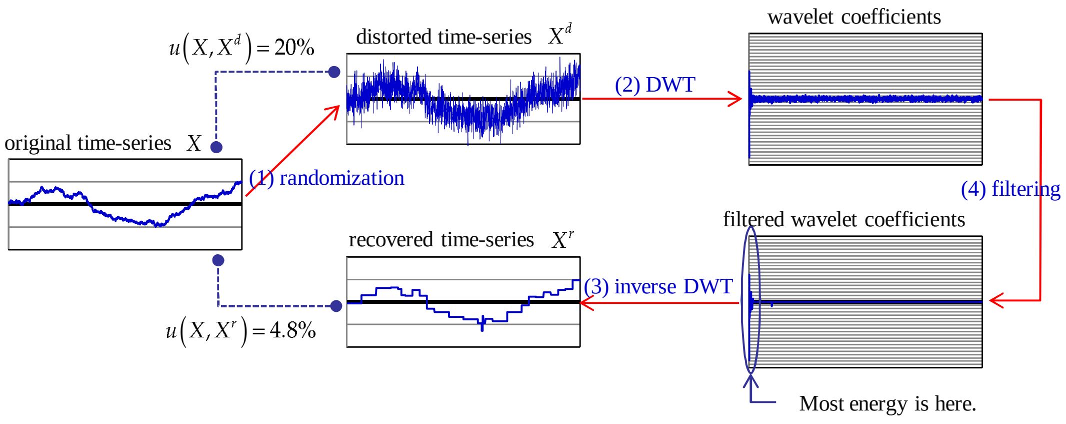


FIGURE 4. Recovery of similar time-series by the wavelet filter

3.3. **WAVE: wavelet-based noise and PAA distances.** Papadimitriou et al. [33] pointed out the privacy problem of randomization and proposed a novel solution to avoid the filtering attack. Their solution generates a noise time-series by considering wavelet coefficients of an original time-series; more specifically, less energy coefficients have no contribution to making a noise time-series, but higher energy coefficients have much contribution to making it. Figure 5 shows how their solution makes a distorted time-series. As shown in Figure 5, we first get a sequence of wavelet coefficients, $X^w = \{x_1^w, \ldots, x_n^w\}$, from an original time-series X. We then construct a sequence of noise coefficients, $N^w = \{n_1^w, \ldots, n_n^w\}$, based on X^w. A noise coefficient n_i^w is set to 0 if its corresponding wavelet coefficient x_i^w is less than the given uncertainty σ; in contrast, n_i^w is set to GaussRand $(0, c \cdot \sigma)$ if x_i^w is not less than σ, where $c = \sqrt{n/|\{x_i^w | x_i^w \geq \sigma\}|}$ (refer to [33] for details). This process explains that less energy coefficients are ignored, but higher energy coefficients have much noise. We next make a noise time-series N from N^w through the inverse DWT. We finally obtain a distorted time-series X^d by adding N to X and publish it to third parties. Papadimitriou et al. [33] showed that the noise of the resulting time-series was not removed by the wavelet filter[5].

We now propose another randomization method that uses the wavelet-based noise [33] in distorting time-series. We call it *WAVE*. WAVE solves the recovering problem of RAND by using the wavelet-based noise, but it still uses the PAA distance in comparing distance orders as in RAND. The formal algorithm of WAVE is given in Algorithm 2. In Line 1, we get wavelet coefficients from the given time-series. In Lines 2-6, we obtain noise coefficients by considering energy of original coefficients. In Line 7, we make a noise time-series from the noise coefficients through the inverse DWT. In Lines 8-9, we construct a distorted time-series and publish it to third parties. Like RAND, each data source publishes its time-series using WAVE, and third parties use the PAA distance to mine the meaningful patterns.

WAVE, however, incurs another problem of destroying distance orders. According to our preliminary experiment, distance orders are severely destroyed in WAVE even though we use the PAA distance. Figure 6(a) shows the experimental result that WAVE is very worse in preserving distance orders. Experimental data and measures are the same

[5]In [33], a DFT-based solution was also proposed. According to the experiments, however, the DWT-based solution was much stronger than the DFT-based solution in preserving the uncertainty, and we thus consider the DWT-based solution only in this paper.

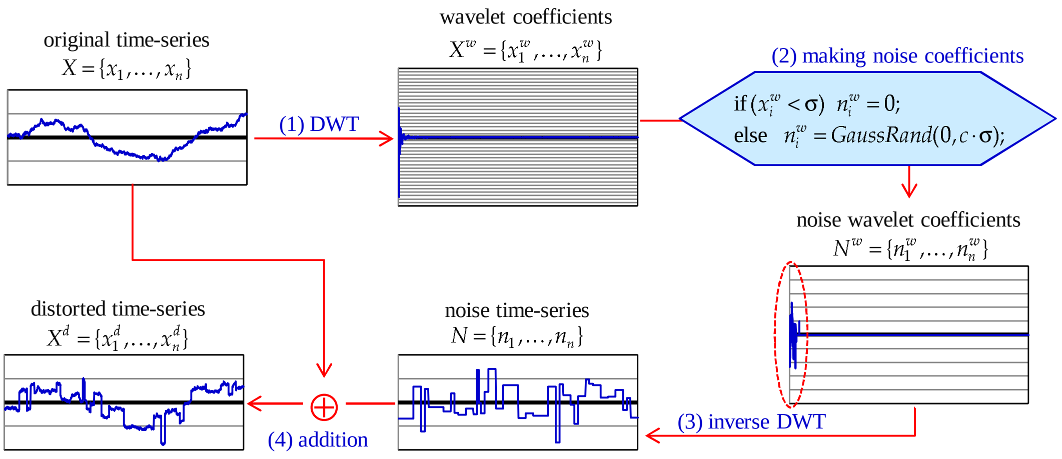
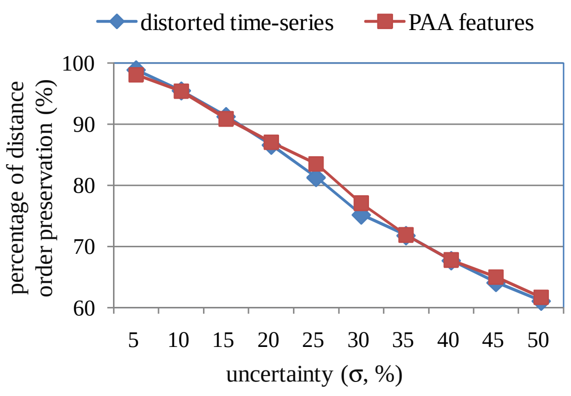

original time-series
$X = \{x_1, \ldots, x_n\}$
(1) DWT
wavelet coefficients
$X^w = \{x_1^w, \ldots, x_n^w\}$
(2) making noise coefficients
if $(x_i^w < \sigma)$ $n_i^w = 0$;
else $n_i^w = GaussRand(0, c \cdot \sigma)$;
noise wavelet coefficients
$N^w = \{n_1^w, \ldots, n_n^w\}$
(3) inverse DWT
noise time-series
$N = \{n_1, \ldots, n_n\}$
(4) addition
distorted time-series
$X^d = \{x_1^d, \ldots, x_n^d\}$

FIGURE 5. WAVE-based noise time-series and its distorted time-series

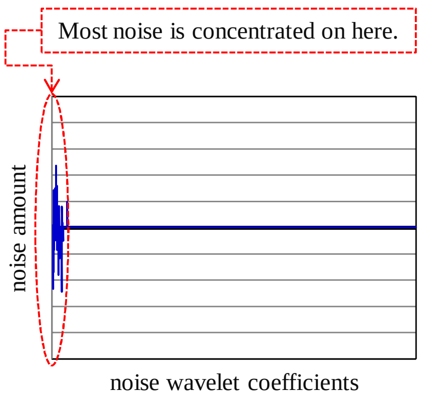
Algorithm 2 WAVE $(X = \{x_1, \ldots, x_n\}, \sigma)$

```
1: Get a sequence $X^w$ of wavelet coefficients from $X$;
2: $c := \sqrt{n/|\{x_i^w | x_i^w \geq \sigma\}|}$;
3: for $i := 1$ to $n$ do // get a seq. $N^w$ of noise wavelet coefficients
4:     if $x_i^w < \sigma$ then $n_i^w := 0$;
5:     else $n_i^w :=$ GaussRand $(0, c \cdot \sigma)$;
6: end-for
7: Make a noise time-series $N$ from $N^w$ through the inverse DWT;
8: Get a distorted time-series $X^d$ from $X$ and $N$; // $x_i^d = x_i + n_i$
9: Publish the distorted time-series $X^d$ to third parties;
```

as those of RAND in Figure 3. As shown in Figure 6(a), compared with RAND (see the upper left graph of Figure 3), the percentage of distance order preservation sharply decreases as the uncertainty increases. In particular, even the PAA distance does not alleviate this sharp decrease. The reason why WAVE destroys distance orders is that only a few high levels of wavelet coefficients are considered to make a noise time-series. That is, most noise is concentrated on a very small part of noise wavelet coefficients as shown in Figure 6(b). This concentration simply makes WAVE stronger against the wavelet filtering attack and preserves the uncertainty well, but at the same time it significantly destroys distance orders.



(a) Distance order preservation of WAVE.



(b) Noise concentration in WAVE

FIGURE 6. Destruction of distance orders in WAVE and its reason

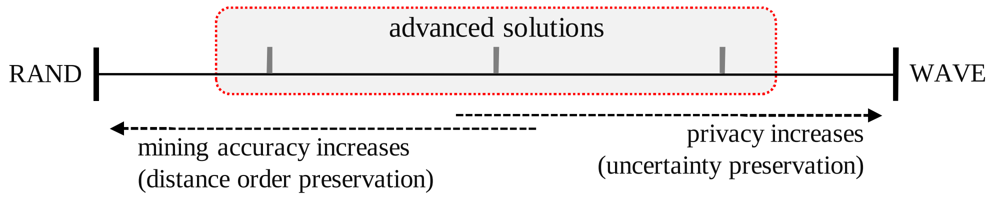

advanced solutions
RAND
WAVE
mining accuracy increases
(distance order preservation)
privacy increases
(uncertainty preservation)

FIGURE 7. Advanced solutions between RAND and WAVE

4. Advanced Solutions.

In this section, we propose the advanced solutions that overcome the problems of RAND and WAVE. As we explained in Section 3, RAND has a problem in preserving privacy; WAVE has another problem in preserving distance orders. In other words, RAND is an extreme example of focusing on distance orders; WAVE is an extreme one of focusing on privacy. Thus, we can say that there is a tradeoff relationship between privacy and mining accuracy. It means that, as shown in Figure 7, there can be some intermediate solutions in between RAND and WAVE. These solutions are derived from two basic principles: (1) use the wavelet-based noise to get the higher degree of uncertainty preservation, and at the same time (2) use the PAA distance to get the higher degree of distance order preservation. In this section we discuss those advanced solutions that take advantages of both RAND and WAVE.

4.1. SNIL: Spread Noise to Intermediate Levels of wavelet coefficients.

Our first advanced solution spreads the noise to intermediate levels of wavelet coefficients in making a noise time-series. We call it *SNIL*. SNIL comes from an observation that WAVE concentrates most noise to only a few high levels of wavelet coefficients, and this concentration destroys distance orders. Based on this observation, SNIL spreads the noise to several intermediate levels instead of a few high levels. By not using a few high levels, some of noise can be removed by the wavelet filter; in contrast, by using intermediate levels rather than low levels, much noise can be preserved as the uncertainty. Moreover, using the PAA distance SNIL preserves distance orders relatively well since it spreads the noise to several levels of wavelet coefficients. Likewise, by spreading the noise to multiple intermediate levels, SNIL preserves the uncertainty more than RAND, and at the same time it preserves distance orders more than WAVE.

Algorithm 3 shows the distortion procedure of SNIL. As inputs to the algorithm, start and end levels, l_s and l_e, are given together with an original time-series X and the uncertainty σ. We let $l_s < l_e$, i.e., l_s be lower than l_e in wavelet levels. Thus, if the highest level is $L\,(=\log_2 n)$, the start and end levels, l_s and l_e, have 2^{L-l_s} and 2^{L-l_e} $(< 2^{L-l_s})$ coefficients, respectively. Like Algorithm 2, in Line 2 we compute how many coefficients will have the noise and obtain the constant factor c of that noise. In Lines 3-6 we assign the noise to the coefficients whose levels are in between start and end levels. This noise assignment spreads the given uncertainty to intermediate levels of l_s to l_e. The rest of Algorithm 3 is the same as Algorithm 2. Figure 8 shows a graphical representation of SNIL. Comparing Figure 8 with Figure 5 of WAVE, SNIL spreads the noise to more wavelet coefficients of intermediate levels. This spread leads a well distribution of noise compared with WAVE, and through this well distribution we get the noise time-series that is strong to the wavelet filter and adequate to the PAA distance.

Start and end levels of l_s and l_e in Algorithm 3 represent on which levels we concentrate the noise. If those levels are close to the highest level (i.e., level $L(=\log_2 n)$), the resulting time-series becomes similar to that of WAVE; in contrast, if those levels are close to the lowest level (i.e., level 1), the resulting time-series becomes similar to that of RAND. In this paper we use $\left\lceil \frac{1}{2}\log_2 n \right\rceil$ as the start level l_s and $\left\lfloor \frac{3}{4}\log_2 n \right\rfloor$ as the end level l_e. For

Algorithm 3 SNIL ($X = \{x_1, \ldots, x_n\}$, σ, start l_s, end l_e)

```
1: Get a sequence $X^w$ of wavelet coefficients from $X$;
2: $c := \sqrt{n / \sum_{l=l_s}^{l_e} 2^{L-l}}$; // $\sum_{l=l_s}^{l_e} 2^{L-l}$: the number of coefficients in $l_s$ to $l_e$ levels
3: for $i := 1$ to $n$ do // get a seq. $N^w$ of noise wavelet coefficients
4:     if $x_i^w$'s level is in $[l_s, l_e]$ then $n_i^w := GaussRand\,(0, c \cdot \sigma)$;
5:     else $n_i^w := 0$;
6: end-for
7: Make a noise time-series $N$ from $N^w$ through the inverse DWT;
8: Get a distorted time-series $X^d$ from $X$ and $N$; // $x_i^d = x_i + n_i$
9: Publish the distorted time-series $X^d$ to third parties;
```

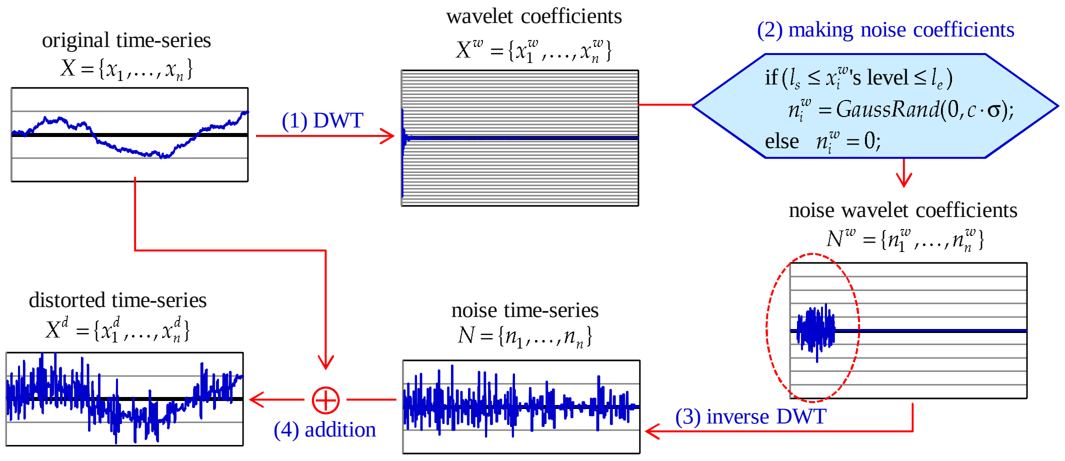
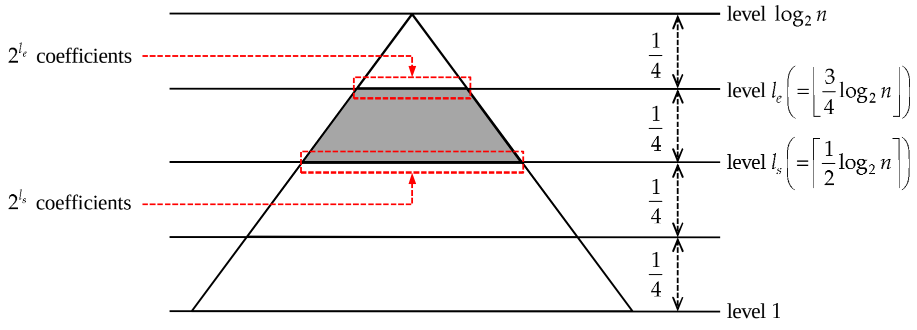

original time-series
$X = \{x_1, \ldots, x_n\}$
(1) DWT
wavelet coefficients
$X^w = \{x_1^w, \ldots, x_n^w\}$
(2) making noise coefficients
if ($l_s \le x_i^w$'s level $\le l_e$)
$n_i^w = GaussRand(0, c \cdot \sigma)$;
else $n_i^w = 0$;
noise wavelet coefficients
$N^w = \{n_1^w, \ldots, n_n^w\}$
(3) inverse DWT
noise time-series
$N = \{n_1, \ldots, n_n\}$
(4) addition
distorted time-series
$X^d = \{x_1^d, \ldots, x_n^d\}$

FIGURE 8. SNIL-based noise time-series and its distorted time-series

example, for a time-series of which length n is 2048 $(= 2^{11})$, we set l_s to 6 $(= \lceil \frac{1}{2} \log_2 2^{11} \rceil)$ and l_e to 8 $(= \lfloor \frac{3}{4} \log_2 2^{11} \rfloor)$, respectively. This is based on the real experimental result on random walk time-series such that if l_s is lower than $\lceil \frac{1}{2} \log_2 n \rceil$, too much noise is removed by the wavelet filter, and if l_e is higher than $\lfloor \frac{3}{4} \log_2 n \rfloor$, too many distance orders are destroyed. Figure 9 shows that those two levels are located in which levels among all wavelet levels. As shown in the figure, we spread the noise to $(2^{l_s} - 2^{(l_e-1)})$ coefficients of total n coefficients. We also note that optimal l_s and l_e vary according to the data set. If we use high values for l_s and l_e, SNIL might be better in preserving uncertainty, but might be worse in preserving distance orders and clustering accuracy; in contrast, if we use low values, it might be better in preserving distance orders and clustering accuracy, but might be worse in preserving uncertainty. Finding the optimal parameters of SNIL, however, is a challenging issue, and we leave it as a further study. For simplicity, we use $\lceil \frac{1}{2} \log_2 n \rceil$ and $\lfloor \frac{3}{4} \log_2 n \rfloor$ only in the experiment of Section 5.

4.2. **DAPI: Divide And Perturb Independently.** The second advanced solution, called *DAPI*, divides a time-series into several pieces and perturbs those pieces independently. DAPI exploits the noise averaging effect on each piece of a time-series by adding the noise to that piece independently. That is, it localizes the noise to small individual pieces, and thus the summation of noise in each piece becomes close to 0. Regarding each piece as a unit of computing the PAA distance, we can easily exploit the noise averaging effect in DAPI. On the other hand, considering the wavelet filter we use the same distortion procedure of WAVE in distorting each piece: it computes a wavelet-based noise for each piece and adds the noise to that original piece. Likewise, by localizing the noise



2^{l_e} coefficients
2^{l_s} coefficients
level $\log_2 n$
level $l_e\left(=\left\lfloor\frac{3}{4}\log_2 n\right\rfloor\right)$
level $l_s\left(=\left\lceil\frac{1}{2}\log_2 n\right\rceil\right)$
level 1
$\frac{1}{4}$
$\frac{1}{4}$
$\frac{1}{4}$
$\frac{1}{4}$

FIGURE 9. Start and end levels and their wavelet coefficients in SNIL

to individual pieces, DAPI preserves the uncertainty more than RAND and preserves distance orders more than WAVE.

Algorithm 4 shows the distortion procedure of DAPI. As inputs to the algorithm, the number of pieces p is given together with an original time-series X and the uncertainty σ. In Line 1 of the algorithm, we divide a time-series of length n to p pieces of length $\frac{n}{p}$. In Lines 2-5, for each original piece, we get a corresponding noise piece that contains the given uncertainty σ. This procedure of obtaining the noise piece is the same as Lines 1-7 of Algorithm 2 in WAVE. In Lines 6-8, we simply concatenate all noise pieces to a noise time-series, construct a distorted time-series, and publish it to third parties. Figure 10 depicts graphical procedures of Algorithm 4. As shown in the figure, before adding the noise, we divide a time-series into several pieces to localize the noise to individual pieces; after dividing the time-series, we add the same amount of noise to those pieces independently to enforce the given uncertainty.

Algorithm 4 DAPI ($X = \{x_1, \ldots, x_n\}$, σ, # of pieces p)

```
1: Divide the original time-series $X$ into $p$ pieces of length $\frac{n}{p}$; // $p$ is a factor of $n$.
2: for $i := 1$ to $p$ do
3:     Let $Y_i$ be the $i$-th piece of the time-series $X$;
4:     Get a noise piece $N_i$ from $Y_i$ using WAVE (Lines 1-7);
5: end-for
6: Make a noise time-series $N$ by concatenating the noise pieces $N_i$;
7: Get a distorted time-series $X^d$ from $X$ and $N$; // $x_i^d = x_i + n_i$
8: Publish the distorted time-series $X^d$ to third parties;
```

To use DAPI in distorting time-series, we need to determine the number of pieces, which is given as an input p to Algorithm 4. As the number of pieces (p) increases, the number of entries ($\frac{n}{p}$) contained in a piece decreases. The smaller number of entries (i.e., the larger number of pieces) makes it difficult to add the given uncertainty correctly, but it well preserves distance orders. This is because a small length time-series cannot have a large amount of noise while its average relatively well reflects all entries. In contrast, the larger number of entries (i.e., the smaller number of pieces) well preserves the uncertainty, but it does not preserve distance orders well. In other words, as we increase the number of pieces, DAPI shows a similar trend with RAND; in contrast, as we decrease the number, it shows a similar trend with WAVE. We use $\frac{1}{2}$ and $\frac{3}{4}$ of wavelet levels in SNIL, and to be consistent with SNIL we choose their average $\frac{7}{8}\left(=\frac{\frac{1}{2}+\frac{3}{4}}{2}\right)$ as the number of pieces in DAPI. More precisely, we set p to a factor of n that is closest to $\frac{7}{8}\log_2 n$. For example, for a time-series of length 2048 ($= 2^{11}$), we set p to 8, which is a factor of 2048 and closest to

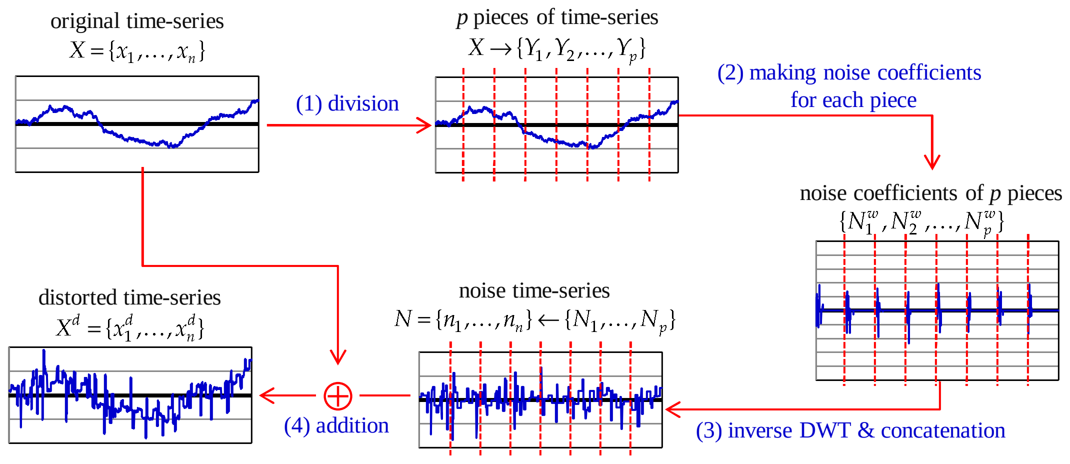

original time-series
$X = \{x_1, \ldots, x_n\}$
(1) division
p pieces of time-series
$X \rightarrow \{Y_1, Y_2, \ldots, Y_p\}$
(2) making noise coefficients for each piece
noise coefficients of p pieces
$\{N_1^w, N_2^w, \ldots, N_p^w\}$
(3) inverse DWT & concatenation
noise time-series
$N = \{n_1, \ldots, n_n\} \leftarrow \{N_1, \ldots, N_p\}$
(4) addition
distorted time-series
$X^d = \{x_1^d, \ldots, x_n^d\}$

FIGURE 10. DAPI-based noise time-series and its distorted time-series

$\frac{7}{8} \log_2 2^{11}$ ($= 9.625$). As we explained in SNIL of Section 4.1, however, finding the optimal parameter p of DAPI is a difficult and challenging issue, and we thus leave it as a further study.

4.3. **SNAM: Spread Noise to As Many levels as possible.** Our third advanced solution, which is an improved version of SNIL, tries to spread the given uncertainty to as many wavelet levels as possible. We call it *SNAM*. SNIL in Section 4.1 spreads the noise to all coefficients of the predefined levels. Considering all coefficients evenly, however, makes many coefficients have low noise, which can be easily removed by the wavelet filter. To overcome this problem, SNAM concentrates the noise only on the representative high energy coefficients of each level, which would not be filtered out, instead of considering all coefficients evenly. For this purpose, we exploit the *use-and-probe* strategy, which put the noise to the same level coefficients and select the representative ones that might be remained after filtering. This use-and-probe strategy works as follows. First, we put the noise to the lowest level coefficients (*use*), try to filter out the noise (*probe*), and keep the coefficients that might be remained after filtering. Through this strategy, some noise is assigned to the lowest level coefficients. Second, we put the remaining noise to the next lower level using the same use-and-probe process. Third, we repeat this step from the next lower to highest levels. Considering all wavelet levels, SNAM spreads the noise to arbitrary coefficients of many levels, and it thus preserves distance orders well compared with WAVE (or SNIL). In addition, since it concentrates the noise on the representative wavelet coefficients, it also preserves the uncertainty well compared with RAND (or SNIL).

The formal algorithm of SNAM is shown in Algorithm 5. Like RAND and WAVE, but unlike SNIL and DAPI, inputs to the algorithm are only an original time-series X and the uncertainty σ. Like other algorithms, in Line 1 SNAM gets wavelet coefficients from the given time-series. In Line 2 we set the remaining uncertainty to the given uncertainty (we use its variance to easily notate the remaining uncertainty in Line 15). After then, we apply the use-and-probe strategy on each level (Lines 3 to 16). Lines 4 to 10 represent the *use* stage which puts the noise to the current level and gets the corresponding distorted time-series; Lines 11 to 14 shows the *probe* stage which investigates whether each coefficient might be remained or not after filtering. In Line 15 we update the remaining uncertainty by reflecting the noise amount of the current level. After obtaining noise coefficients of all wavelet levels, we combine them into one sequence of noise coefficients in Line 16. Finally, we obtain the noise time-series (Line 17), construct a distorted time-series (Line 18), and publish it to third parties (Line 19). SNAM does not require additional parameters, such as l_s and l_e in SNIL and p in DAPI, because it considers all wavelet levels one by one to spread the noise. Thus, we can say that SNAM is more adequate to normal users since it avoids complex additional parameters. This advantage, however, can also be a disadvantage because SNAM cannot control the importance of uncertainty or distance orders while SNIL and DAPI can do that control.

Algorithm 5 SNAM ($X = \{x_1, \ldots, x_n\}$, σ)

```
1: Get a sequence $X^w$ of wavelet coefficients from $X$;
2: $\sigma_{cur}^2 := \sigma^2$;
3: for $l := 1$ to $L := \log_2 n$ do // for each wavelet level
4:     $c := \sqrt{n/2^{L-l}}$; // $2^{L-l}$: # of coefficients in $l$-th level
5:     for $i := 1$ to $n$ do
6:         if $x_i^w$ is in $l$-th level then $n_{l,i}^w :=$ GaussRand $(0, c \cdot \sigma_{cur})$;
7:         else $n_{l,i}^w := 0$;
8:     end-for
9:     Make $N_l$ from $N_l^w$ $(= \{n_{l,1}^w, \ldots, n_{l,n}^w\})$ by the inverse DWT;
10:    Get $Y$ $(= \{y_1, \ldots, y_n\})$ from $X$ and $N_l$; // $y_i = x_i + n_{l,i}$
11:    Obtain a sequence $Y^w$ of wavelet coefficients from $Y$;
12:    for $i := n/2^l + 1$ to $n/2^{l-1}$ do // for each coef. in $l$-th level
13:        if $|y_i^w| < \sigma$ then $n_{l,i}^w := 0$; // check if it might be filtered
14:    end-for
15:    $\sigma_{cur}^2 := \sigma_{cur}^2 - \frac{2^l}{n} \sum_{i:=n/2^l+1}^{n/2^{l-1}} (n_{l,i}^w)^2$; // remaining uncertainty
16: end-for
17: for $i := 1$ to $n$ do
18:    $n_i^w := \sum_{l:=1}^{\log_2 n} n_{l,i}^w$; // combine noise coefficients of all levels
19: end-for
20: Make a noise time-series $N$ from $N^w$ through the inverse DWT;
21: Get a distorted time-series $X^d$ from $X$ and $N$; // $x_i^d = x_i + n_i$
22: Publish the distorted time-series $X^d$ to third parties;
```

We note that, compared Algorithm 5 with Algorithms 1 to 4, SNAM is much more complicated than all other algorithms including SNIL and DAPI. However, all the algorithms proposed in the paper can be seen as a preprocessing step, which is executed before publishing time-series data, and they do not affect uncertainty preservation and distance order preservation at all. Thus, in this paper we do not analyze complexity of the algorithms in detail. On the other hand, if we need to publish time-series data in real-time for the streaming environment, the complexity analysis will be an important issue. We leave this analysis for the real-time environment as our further work.

5. Experimental Evaluation.

In the experiment we compared the proposed solutions using different data sets obtained from the UCR data [22]. The UCR data were widely used in time-series analysis, and we selected three data sets, CBF (143 time-series of length 60), ECG200 (600 time-series of length 319), and Two_Patterns (512 time-series of length 1024), which were suitable for evaluating clustering and classification algorithms on time-series data.

We implemented our five distortion methods: two naive methods of RAND and WAVE in Section 3 and three advanced methods of SNIL, DAPI, and SNAM in Section 4. We measured three evaluation metrics: (1) *uncertainty preservation*, how many time-series preserved their uncertainty after the distortion; (2) *distance order preservation*, how many time-series preserved their distance orders; (3) *clustering accuracy preservation*, how many clusters preserved their original clustering results. In particular, for the clustering accuracy preservation, we first obtained the actual clustering result from the original data set and then compared it with the results of the distorted data sets. The hardware platform was SUN Ultra 25 workstation equipped with UltraSPARC IIIi CPU 1.34GHz, 1.0GB RAM, and an 80GB HDD. Its software platform was the Solaris 10 operating system. We used C/C++ language for implementing our solutions and k-means clustering algorithm.

5.1. **Uncertainty preservation.** To evaluate the uncertainty preservation of each distortion method, we performed the following two steps: (1) the distorted time-series was generated by adding the noise to the original time-series; (2) the recovered time-series was obtained by applying the wavelet filter to the distorted time-series. Through these two steps, we investigated how much noise was remained after the filtering attack; more specifically, for a given time-series X, we compared the remaining uncertainty $u(X, X^r)$ with the given uncertainty $u(X, X^d)$. In the experiment, we measured $u(X, X^d)$ and $u(X, X^r)$ for every time-series X, and used their average as the result.

Figure 11 shows the experimental result on uncertainty preservation. First, in Figure 11(a) of CBF, we note that WAVE shows the best result while RAND shows the worst result. As we explained in Section 4, this is because WAVE extremely focuses on preserving the uncertainty while RAND extremely focuses on preserving distance orders. In Figure 11(a) we also note that our advanced solutions, SNIL, DAPI, and SNAM, are in between WAVE and RAND. In particular, SNIL and SNAM, which try to spread the noise to many wavelet levels, are comparable with WAVE in preserving the uncertainty. This is because, even though the given noise is spread to many wavelet coefficients, large portion of the noise is still concentrated on a small number of higher wavelet levels, and it is not easily removed by the wavelet filter. More precisely, SNIL is very comparable to or even better than WAVE in preserving uncertainty (see ECG200 of Figure 11(b)). This is because, in SNIL, most noise is concentrated in the intermediate levels, where the noise is not removed well. On the other hand, DAPI is relatively worse than SNIL and SNAM because it has difficulty in generating the full amount of noise due to the small size of individual pieces.

Figures 11(b) and 11(c), the results of ECG200 and Two_Patterns, show the similar trend with Figure 11(a) of CBF. In summary of Figure 11, our advanced solutions take advantage of WAVE and show the better uncertainty preservation compared with RAND. In particular, SNIL and SNAM are comparable with WAVE in preserving the uncertainty.

5.2. **Distance order preservation.** To evaluate the distance order preservation, we first constructed 10,000 triplets from each data sets. Each triplet consisted of three time-series

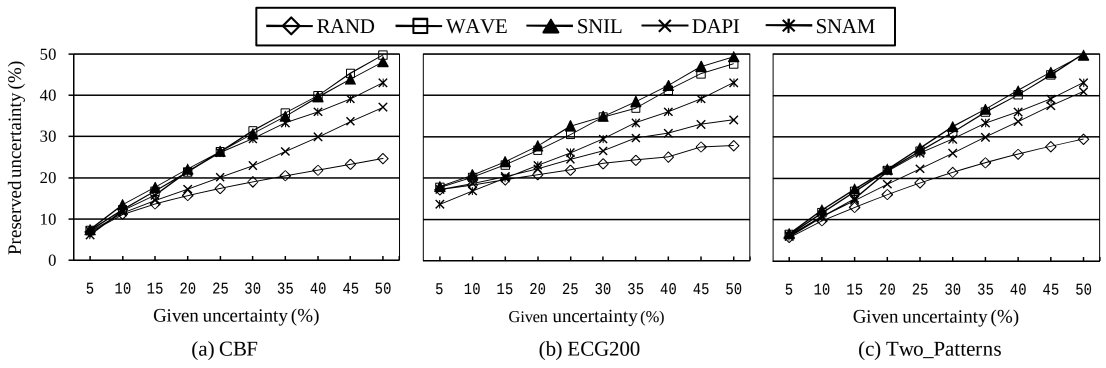


FIGURE 11. Experimental result on uncertainty preservation

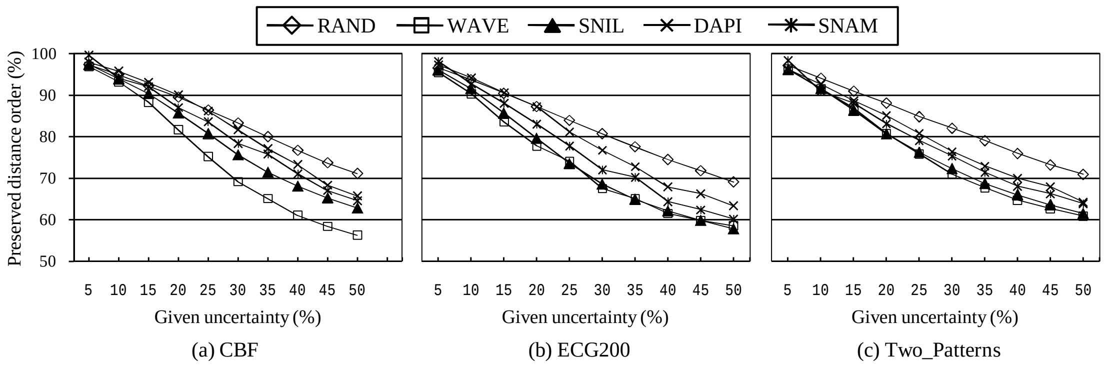


Figure 12. Experimental result on distance order preservation

as in Definition 2.2. We measured how much percent of triplets preserved their distance orders by investigating all 10,000 triplets for each method.

Figure 12 shows the experimental result on distance order preservation. As shown in the figure, for all three data sets, RAND shows the best result, but WAVE shows the worst result. This result is exactly opposite to that of uncertainty preservation. As we explained earlier, this is because RAND emphasizes distance orders while WAVE focuses on uncertainty. Similar to uncertainty preservation, our advanced solutions are in between RAND and WAVE by taking advatage of RAND in preserving distance orders. Unlike Figure 11, however, DAPI is better than SNIL and SNAM in Figure 12 because DAPI well preserves the PAA distance by dividing a long time-series into smaller pieces.

A notable point in comparing Figures 11 and 12 is a tradeoff relationship between uncertainty (i.e., privacy) and distance orders (i.e., mining accuracy). We note that the uncertainty result of "RAND $\lesssim$ DAPI $\lesssim$ SNAM $\lesssim$ SNIL $\lesssim$ WAVE" is exactly opposite to the distance order result of "WAVE $\lesssim$ SNIL $\lesssim$ SNAM $\lesssim$ DAPI $\lesssim$ RAND". In case of SNIL and DAPI, we can emphasize one of uncertainty and distance orders by adjusting the input parameters ((l_s, l_e) in SNIL and p in DAPI). Thus, we can say SNIL and DAPI are more flexible than other methods in adjusting the tradeoff relationship.

5.3. **Clustering accuracy preservation.** We also experimented the actual clustering accuracy of the proposed distortion methods. As the measure of clustering accuracy, we used *F-measure* [29], which was widely used in information retrieval or data mining to evaluate the accuracy of retrieved or mined results. F-measure was computed by comparing the resulting clusters of original time-series and those of distorted time-series. In our experiment, the higher F-measure means the more accurate clustering result. For the detailed explanation of computing F-measures, readers are referred to [29].

After executing k-means algorithm for each of RAND, WAVE, SNIL, DAPI, and SNAM, we obtain their F-measures, respectively. Figure 13 shows the resulting F-measures on three data sets. As shown in the figure, the experimental result of F-measures is very similar to that of distance order preservation in Figure 12. This means that preserving distance orders well reflects preserving clustering (or mining) accuracy. In other words, if a distortion method preserves distance orders well, it also preserves mining accuracy well. In summary, like the distance order preservation, RAND is the best while WAVE is the worst, and the advanced solutions are still in between RAND and WAVE.

6. **Related Work.**

As we mentioned in Section 1, PPDM solutions can be classified into four categories [1]: random data perturbation [12, 16, 24, 33], k-anonymization [3, 23, 36], distributed privacy preservation [6, 14, 35, 37, 38, 39], and privacy preserving of mining

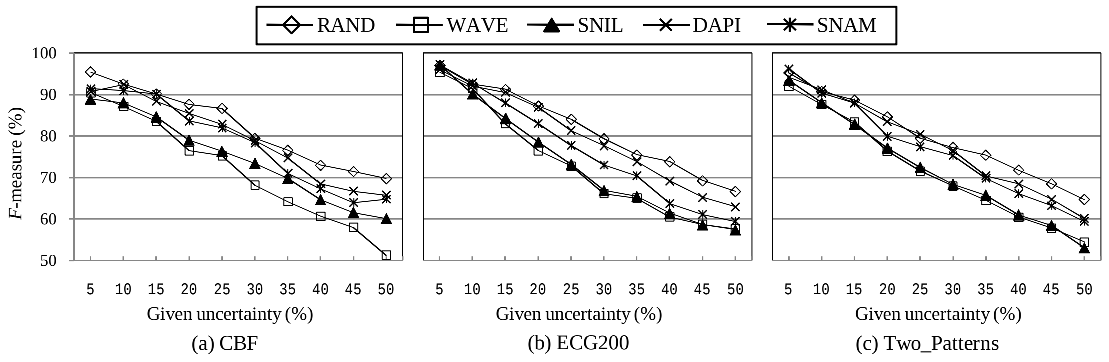


Figure 13. Experimental result on clustering accuracy preservation

results [26, 32, 40]. We review these previous solutions with respect to our perturbation approach.

First, the random data perturbation is a technique for PPDM in which noise is added to the data in order to mask the sensitive values of data [1]. Agrawal and Srikant [2] first proposed random perturbation-based PPDM solutions. This random perturbation can be easily used for adding noise to time-series data, but it distorts distance orders as well and eventually incurs bad mining accuracy [33]. Like RAND in Section 3.2, it also has a noise filtering problem by the wavelet filter [33]. That is, the simple random perturbation does not satisfy our requirements of privacy and distance order preservation even though it satisfies all three assumptions (see [2] in Table 1). Geometric transformation [16, 30] and rotation perturbation [12] were proposed to get a *set* of distorted time-series from a *set* of original time-series. These solutions relatively well preserve distance orders and provide higher clustering/classification accuracy. However, they cannot deal with an individual time-series of a specific data source, and their related parameters should not be disclosed to preserve privacy. That is, these solutions do not satisfy full disclosure and independency assumptions as well as the privacy preservation requirement. Li et al. [24] exploited dynamic correlation and auto-correlation to perturb multiple streams. This solution, however, does not satisfy the independency assumption due to considering multiple streams at the same time. Oliveira and Zanane [30] presented the random projection that extracted k dimensions from n dimensions. To choose the projected dimensions, however, their method should investigate all time-series, and this does not satisfy our independency assumption. Also, if the projected dimensions are revealed, the privacy is not preserved anymore. Papadimitriou et al. [33] proposed a novel perturbation solution which generated the wavelet-based noise to preserve privacy (i.e., uncertainty) against the filtering attack. This solution, on which our WAVE and advanced solutions are based, satisfies all three assumptions and the privacy preservation requirement, but it is still inadequate to the distance order preservation requirement as we explained in WAVE. Likewise, all the previous perturbation methods do not satisfy one or more assumptions or requirements. We compared and summarized these perturbation methods in Table 1 of Section 2.

Second, k-anonymization increases anonymity of data by reducing the granularity of data representation with the use of generalization and suppression [1]. Many research results [3, 23, 36] have been proposed to provide various types of anonymity, including k-anonymity, l-diversity, and t-closeness, on various types of databases or environments. We can adopt the concept of k-anonymity in publishing time-series as follows: "Ensure at least k time-series should be similar (i.e., at least k time-series should be less than or equal to the given tolerance)." This anonymity problem is orthogonal to our distortion problem, and accordingly, we may use our solutions to solve this problem.

Third, distributed privacy preservation provides secure mining protocols for the distributed environment. In general, those solutions use the cryptography-based secure multiparty computation (SMC) techniques to preserve data privacy of individual entities. Clifton and Vaidya [6, 39] proposed various types of SMC operations and used those operations for privacy preserving clustering [38] and association rule mining [37]. Pinkas [35] discussed various cryptography techniques for PPDM, and recently Jiang et al. [14] used the secure scalar product to detect similar documents with preserving privacy. Those SMC-based solutions, however, are not suitable for our privacy model of publishing time-series since they do not satisfy our independency and full disclosure assumptions; that is, in their solutions data sources should have to co-work together, or some encryption parameters should be hidden from others.

Fourth, privacy preserving of mining results prevents the outputs (i.e., mining results) from disclosing data privacy [1]. For example, Oliveira et al. [32] and Verykios et al. [40] proposed solutions to hide the sensitive association rules that might disclose private data. We can also adopt this concept in publishing time-series as follows: "Ensure that the original time-series cannot be recovered even though the mining results are published." Like k-anonymity, this output problem is also orthogonal to our distortion problem, and we may use our solutions to solve this problem.

Recently, Mukherjee and Chen [29] proposed a novel solution to privacy preserving clustering on time-series data. They pointed out that the data perturbation [2, 30, 33] severely distorted (Euclidean) distance orders of time-series. To solve this problem, they tried to publish a few Fourier coefficients instead of a whole time-series. Since the Fourier coefficients well preserve the Euclidean distance, their method provides a higher clustering accuracy. Their method, however, may cause privacy breach if positions of Fourier coefficients are revealed. To avoid this problem, they tried to hide the exact positions through the sophisticated permutation protocol [29]. However, the positions can be easily found if only one original time-series and its published coefficients are disclosed to attackers.

7. Conclusions.

In this paper we addressed the problem of publishing time-series data. Time-series data, which have been widely used in many mining areas, are very sensitive since a time-series itself may disclose its corresponding private information (e.g., identifier). Thus, preserving both privacy and mining accuracy is an important issue in publishing time-series data. In this paper we presented two naive and three advanced solutions which considered mining accuracy preservation as well as privacy preservation.

Our work can be summarized as follows. First, we proposed a privacy model of publishing sensitive time-series data and derived the related assumptions and requirements. Second, we analyzed the randomization-based solutions on the proposed privacy model and presented common problems of these previous solutions. Third, we introduced a novel notion of the *noise averaging effect* of PAA and explained that the *PAA distance* might well preserve distance orders for the higher mining accuracy. Fourth, we proposed two naive randomization methods, RAND and WAVE, by exploiting the PAA distance. Fifth, to take advantages of both RAND and WAVE, we proposed three more engineering solutions, SNIL, DAPI, and SNAM. Sixth, through extensive experiments, we showcased the superiority of our solutions.

As we explained in the related work, the future work may include the extension of our solution to support the issues of k-anonymization or privacy preserving of mining results. We will also tackle the problem of finding optimal input parameters of SNIL and DAPI and the complexity analysis of the proposed algorithms for the real-time environment.

Acknowledgement. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MEST) (No. 2011-0013235).

REFERENCES

[1] C. C. Aggarwal and P. S. Yu, Privacy-preserving data mining: A survey, in *Handbook of Database Security: Applications and Trends*, M. Gertz and S. Jajodia (eds.), Springer, 2007.

[2] R. Agrawal and R. Srikant, Privacy-preserving data mining, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Dallas, TX, pp.439-450, 2000.

[3] R. J. Bayardo and R. Agrawal, Data privacy through optimal k-anonymization, *Proc. of the 21st Int'l. Conf. on Data Engineering*, Tokyo, Japan, pp.217-228, 2005.

[4] E. Bertino, D. Lin and W. Jiang, A survey of quantification of privacy preserving data mining algorithms, in *Privacy-Preserving Data Mining: Models and Algorithms*, C.-C. Aggarwal and P. S. Yu (eds.), Kluwer Academic Publishers, 2008.

[5] P. Ciaccia, M. Patella and P. Zezula, M-tree: An efficient access method for similarity search in metric spaces, *Proc. of the 23rd Int'l Conf. on Very Large Data Bases*, Athens, Greece, pp.426-435, 1997.

[6] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin and M. Y. Zhu, Tools for privacy preserving distributed data mining, *SIGKDD Explorations*, vol.4, no.2, pp.28-34, 2002.

[7] D. L. Donoho and I. M. Johnstone, Adapting to unknown smoothness via wavelet shrinkage, *Journal of the American Statistical Association*, vol.90, no.432, pp.1200-1224, 1995.

[8] A. V. Evfimievski, R. Srikant, R. Agrawal and J. Gehrke, Privacy preserving mining of association rules, *Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Edmonton, Canada, pp.217-228, 2002.

[9] C. Faloutsos, M. Ranganathan and Y. Manolopoulos, Fast subsequence matching in time-series databases, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Minneapolis, Minnesota, pp.419-429, 1994.

[10] B. Goethals, S. Laur, H. Lipmaa and T. Mielikainen, On secure scalar product computation for privacy-preserving data mining, *Proc. of the 7th Annual Int'l Conf. in Information Security & Cryptology*, Seoul, Korea, pp.104-120, 2004.

[11] J. Han and M. Kamber, Data mining: Concepts and techniques, 2nd Edition, Morgan Kaufmann Publishers, 2006.

[12] Z. Huang, W. Du and B. Chen, Deriving private information from randomized data, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Baltimore, Maryland, pp.37-48, 2005.

[13] A. Inan, M. Kantarcioglu and E. Bertino, Using anonymized data for classification, *Proc. of the 25th Int'l Conf. on Data Engineering*, Shanghai, China, pp.429-440, 2009.

[14] W. Jiang, M. Murugesan, C. Clifton and L. Si, Similar document detection with limited information disclosure, *Proc. of the 24th IEEE Int'l Conf. on Data Engineering*, Cancun, Mexico, pp.735-743, 2008.

[15] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-Interscience, 1990.

[16] H. Kargupta, S. Datta, Q. Wang and K. Sivakumar, On the privacy preserving properties of random data perturbation techniques, *Proc. of the 3rd IEEE Int'l Conf. on Data Mining*, Melbourne, FL, pp.99-106, 2003.

[17] E. Keogh, K. Chakrabarti, M. J. Pazzani and S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, *Knowledge and Information Systems*, vol.3, no.3, pp.263-286, 2001.

[18] E. Keogh, Exact indexing of dynamic time warping, *Proc. of the 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, pp.406-417, 2002.

[19] E. Keogh, J. Lin and A. Fu, HOT SAX: Efficiently finding the most unusual time series subsequence, *Proc. of the 5th IEEE Int'l Conf. on Data Mining*, Houston, TX, pp.226-233, 2005.

[20] E. Keogh, L. Wei, X. Xi, S.-H. Lee and M. Vlachos, LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures, *Proc. of the 32nd Int'l Conf. on Very Large Data Bases*, Seoul, Korea, pp.882-893, 2006.

[21] E. Keogh and C. A. Ratanamahatana, Indexing and mining large time series databases, *Proc. of the 12th Int'l Conf. on Database Systems for Advanced Applications*, Tutorial, Bangkok, Thailland, 2007.

[22] E. Keogh, X. Xi, L. Wei and C. A. Ratanamahatana, *The UCR Time Series for Classification/Clustering*, http://www.cs.ucr.edu/ẽamonn/time_series_data.

[23] K. LeFevre, D. DeWitt and R. Ramakrishnan, Mondrian multidimensional k-anonymity, *Proc. of the 22nd IEEE Int'l Conf. on Data Engineering*, Atlanta, GA, p.25, 2006.

[24] F. Li, J. Sun, S. Papadimitriou, G. Mihaila and I. Stanoi, Hiding in the crowd: Privacy preserving on evoling streams through correlation tracking, *Proc. of the 23rd IEEE Conf. on Data Engineering*, Istanbul, Turkey, pp.686-695, 2007.

[25] Y. Lindell and B. Pinkas, Privacy preserving data mining, *Journal of Cryptology*, vol.15, no.3, pp.36-54, 2002.

[26] M.-C. Lu, C.-Y. Ku, L.-C. Hwang and H.-M. Chao, Using smart card in RFID infrastructure to protect consumer privacy, *International Journal of Innovative Computing, Information and Control*, vol.7, no.4, pp.1777-1788, 2011.

[27] J. MacQueen, Some methods for classification and analysis of multivariate observations, *Proc. of the 5th Berkeley Symp. on Math. Stat. Prob.*, CA, pp.281-297, 1967.

[28] Y.-S. Moon, B.-S. Kim, M. S. Kim and K.-Y. Whang, Scaling-invariant boundary image matching using time-series matching techniques, *Data & Knowledge Engineering*, vol.69, no.10, pp.1022-1042, 2010.

[29] S. Mukherjee, Z. Chen and A. Gangopadhyay, A privacy-preserving technique for Euclidean distance-based mining algorithms using fourier-related transforms, *The VLDB Journal*, vol.15, no.4, pp.293-315, 2006.

[30] S. Oliveira and O. Zanane, Privacy-preserving clustering by data transformation, *Proc. of Brazilian Symp. on Databases*, Amazonas, Brazil, pp.304-318, 2003.

[31] S. Oliveira and O. Zanane, Privacy-preserving clustering by object similarity-based representation and dimensionality reduction transformation, *Proc. of the Workshop on Privacy and Security Aspects of Data Mining*, Brighton, UK, pp.21-30, 2004.

[32] S. Oliveira, O. Zaiane and Y. Saygin, Secure association-rule sharing, *Proc. of the 8th Pacific-Asia Conf. on Advances in Knowledge Discovery & Data Mining*, Sydney, Australia, pp.74-85, 2004.

[33] S. Papadimitriou, F. Li, G. Kollios and P. S. Yu, Time series compressibility and privacy, *Proc. of the 33rd Int'l Conf. on Very Large Data Bases*, Vienna, Austria, pp.459-470, 2007.

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

[35] B. Pinkas, Cryptography techniques for privacy-preserving data mining, *SIGKDD Explorations*, vol.4, no.2, pp.12-19, 2002.

[36] L. Sweeney, k-anonymity: A model for protecting privacy, *Int'l Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol.10, no.5, pp.557-570, 2002.

[37] J. Vaidya and C. Clifton, Privacy preserving association rule mining in vertically partitioned data, *Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Alberta, Canada, pp.639-644, 2002.

[38] J. Vaidya and C. Clifton, Privacy-preserving k-means clustering over vertically partitioned data, *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Washington D.C., pp.206-215, 2003.

[39] J. Vaidya and C. Clifton, Privacy-preserving data mining: Why, how, and when, *IEEE Security & Privacy*, vol.2, no.6, pp.19-27, 2004.

[40] V. S. Verykios, A. Elmagarmid, E. Bertino, Y. Saygin and E. Dasseni, Association rule hiding, *IEEE Trans. on Knowledge & Data Engineering*, vol.16, no.4, pp.434-447, 2004.

[41] H.-L. Wong, C.-C. Wang and Y.-H. Tu, Optimal selection of multivariate fuzzy time series models to non-stationary series data forecasting, *International Journal of Innovative Computing, Information and Control*, vol.6, no.12, pp.5321-5332, 2010.

[42] B.-K. Yi and C. Faloutsos, Fast time sequence indexing for arbitrary L_p norms, *Proc. of the 26th Int'l Conf. on Very Large Data Bases*, Cairo, Egypt, pp.385-394, 2000.

[43] H.-H. Yu, C.-H. Chen and V. S. Tseng, Mining emerging patterns from time series data with time gap constraint, *International Journal of Innovative Computing, Information and Control*, vol.7, no.9, pp.5515-5528, 2011.