Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)

Privacy Preserving Burst Detection of Distributed Time Series Data Using Linear Transforms

Lisa Singh
Department of Computer Science
Georgetown University
Washington, DC 20057
Email: singh@cs.georgetown.edu

Mehmet Sayal
Hewlett-Packard Labs
Palo Alto, CA 94304
Email: mehmet.sayal@hp.com

Abstract— **In this paper, we consider burst detection within the context of privacy. In our scenario, multiple parties want to detect a burst in aggregated time series data, but none of the parties want to disclose their individual data. Our approach calculates bursts directly from linear transform coefficients using a cumulative sum calculation. In order to reduce the chance of a privacy breech, we present multiple data perturbation strategies and compare the varying degrees of privacy preserved. Our strategies do not share raw time series data and still detect significant bursts. We empirically demonstrate this using both real and synthetic distributed data sets. When evaluating both privacy guarantees and burst detection accuracy, we find that our percentage thresholding heuristic maintains a high degree of privacy while accurately identifying bursts of varying widths.**

I. Motivation

In this paper we consider privacy preservation issues in distributed time series data sets. Sensor data, individual store branch (location) purchase data, and network data are examples of distributed time series data sets. In each of these cases, data is partitioned across servers, potentially at different physical locations. The distributed data can be horizontally or vertically partitioned. This paper focuses on a horizontal partitioning of the data where each company or agency has the same time series variable for a disjoint set of the data. For example, suppose VISA and DISCOVER have purchase transaction data. While the transaction purchase data is the same attribute or variable during a given time period, the actual transactions recorded are non-overlapping.

One problem that has received attention in time series data sets is burst detection [1] [2] [3] [4]. Our focus is identifying aggregate bursts in an environment where detection of burst is useful, but raw time series data cannot be shared. Privacy of the individual time series is paramount. Within the aggregate series, we want to identify time intervals containing significantly high aggregate values or bursts. For example, given store purchase data, a store may be interested in identifying products with unusually large sales across store locations. Because each store location maintains its data independently, the data must be merged to detect bursts across all locations. In this example, the store, e.g. Walmart, owns all the data. However, if the data aggregation involved is across independent companies, privacy preservation becomes a large concern. Examples include multiple credit card companies looking for bursts in purchases across an integrated data set, multiple state government agencies interested in detecting bursts in voting anomalies across a region, or multiple hospitals interested in identifying spikes in emergency room care or doctor-patient mortality ratios.

II. Related Literature

Much new research in the area of privacy preservation is surfacing. The work cited here is a representative sample of topics and is not meant to be an exhaustive list. Some work has investigated privacy preservation within the context of traditional data mining problems, e.g. classi□cation [5] [6], association rule mining [7] [8], clustering [9] [10], and regression analysis [11]. For a survey of current approaches and tools for privacy preservation in data mining, we refer you to [12] [13]. Our work differs from this literature since it focuses on detection of aggregate bursts in distributed time series data. To our knowledge, burst detection has not been studied in this context.

As previously mentioned, some privacy preservation work exists on horizontally partitioned data sets [8] [14]. Similar to other work, we employ secure multi-party computation during aggregation. However, this is not the only privacy preserving method we use. Our work differs from traditional burst detection approaches since we make adjustments for maintaining privacy during data aggregation within a distributed environment. Similar to some burst detection work, we use mathematical transforms, speci□cally wavelets [1]. However, in the work we have seen, the bursts are computed locally and are not aggregated from distributed sites. Also, privacy of local data in this context has not been investigated.

III. High Level Approach

This section begins by formally de□ning our notation related to time series data and bursts. We then highlight our approach for addressing the distributed burst detection problem outlined in Section I. In Section IV, we go through this approach in detail, investigating privacy and burst detection accuracy issues as they arise. We begin with two de□nitions related to time series data and bursts within the data.

DEFINITION 1: A time series data set D is described by a sequence of n events E at different times, where $E =$

$\{e_1 \ldots e_n\}$ and e_i is an event at time i. Suppose data is distributed across x sites, then the sequence of events at site j is described as $E_j = \{e_{j1} \ldots e_{jn}\}$ and the complete time series of interest is the sum of the events at the different sites,

$$E = \sum_{j=1}^{t} E_j \tag{1}$$

DEFINITION 2: A burst, B, is a set of events E' having a magnitude significantly greater than or less than the average magnitude of the events, $B \geq \mu_E + c\sigma_E$ or $B \leq \mu_E - c\sigma_E$, where μ is the mean, c is a small constant, and σ is the standard deviation. Typically, a burst is characterized by a small set of events relative to the entire set of events, but since different parties may be interested in different granularities of bursts, the ability to detect bursts at multiple resolutions or varying widths is necessary.

Because data is distributed across sites, we are concerned about external parties that are trying to discover local site data. In this paper, we consider two external groups, uninformed network intruders and informed network intruders.

DEFINITION 3: We define an *uninformed network intruder* as a person with no background knowledge trying to determine time series data from any local site that participates in this aggregate burst detection process. This person has no insight into the aggregation procedure and is not collaborating with a participant. In contrast, an *informed network intruder* understands the aggregation procedure. Similar to the uninformed network intruder, he is not collaborating with a participant. Therefore, while he has a sense of the process, he does not have access to any actual data.

We also assume that all participants are semi-honest, i.e. do not exhibit malicious behavior but will 'cheat' if sufficient information is given to them. All privacy statements and assertions will be described in terms of either uninformed network intruders, informed network intruders, or semi-honest participants.

Our high level approach for burst detection in a privacy constrained distributed environment is shown in Table I. In our distributed environment, one of the participants is designated as the master location or the leader. All of the processes begin and end at that location. The leader begins by sending the participants parameters determined apriori: date range and the minimum and maximum burst width of interest. The date range identifies the period of the analysis. This is used to ensure that the time series are aligned along the time dimension. The minimum and maximum burst widths are necessary to identify the number of events necessary for a burst to exist and the number of events in a window of the time series.

DEFINITION 4: Traditionally, window length, win_len, is a constant value that defines the number of events in equally partitioned subsequences. We extend this and define the win_len as the width of the largest burst of interest. If the largest burst of interest is equal to the number of events in the time series, then there is one window and $win\_len = n$. However, for large time series, smaller windows and shorter maximum burst widths will generally be of interest.

TABLE I
GENERAL APPROACH

Step I	Determine process parameters
Step II	Calculate local transform
Step III	Perturb transform coefficients (optional)
Step IV	Aggregate local coefficients
Step V	Determine aggregate bursts

At this stage (Step II), each participant uses a linear transform to generate the coefficients for a time series during a specified date range and win_len. The linear transforms we will evaluate in this paper are the Haar wavelet, the Daubechies wavelet, and the discrete Fourier transform [15]. Although we confine ourselves to these three, any linear transform can be selected. Once the local transform coefficients have been determined, a coefficient perturbation heuristic can be employed to increase the level of privacy (Step III).

During Step IV, we need to aggregate the data across sites before attempting to detect a burst. This is important because in a distributed environment, a burst may not be detectable by looking at the time series data at a single site. The converse is also true. A burst at a single site may be insignificant when considering the entire aggregated time series, E. We accomplish aggregation by using the secure-multiparty computation (SMC) protocol, where participants sum coefficient values to generate an aggregate set of linear transform coefficients [16]. Bursts are then detected directly from the coefficients by calculating cumulative sums and identifying bursts based on the sums (Step V).

Given the context of the problem, we have two competing goals - accurate burst detection and strong privacy preservation. We will now discuss the approach in more detail within the context of the competing goals. Our aim is to quantify and better understand the trade-off between accuracy and privacy for our general approach. We begin by describing the burst detection algorithm and seeing its accuracy using different linear transforms. We will show experiments throughout to clarify the discussion.

IV. Detailed Discussion of Burst Detection Using Linear Transformation Coefficients

A. *Burst Detection Algorithm*

Burst detection is a specific case of anomaly detection that incorporates change detection. In control theory, Statistical Process Control (SPC) methods are used to detect changes or out-of control conditions in single resolution data. Methods proposed include: Shewhart, moving averages (MA), exponentially weighed moving averages (EWMA), and cumulative sum (CUSUM) control charts [17] [18]. Because our focus is burst detection, Shewhart and CUSUM are the two candidates we will consider.

A Shewhart chart uses a threshold to identify when an observation or an event falls outside a desired control limit

TABLE II
BURST DETECTION ALGORITHM

Step I	Calculate the mean and standard deviation of the aggregate transformation coefficients.
Step II	Use this information to determine the cumulative sum values for the coefficients.
Step III	At a given point, if the cumulative sum is more than c standard deviations away from zero (the ideal difference between the actual value and the mean), then a change point, cp_1, has been detected.
Step IV	Identify each consecutive change point, cp_i. Mark the first and last change points as the start and end of one burst.
Step V	Repeat steps III and IV for all the coefficients.

[19]. The idea is to identify a meaningful threshold, e.g. $\mu + c\sigma$. If a value is above the threshold, it is considered a potential change point. The popularity of this approach results from its simplicity. However, the Shewhart method can only detect sharp change points, not gradually accumulating change.

CUSUM is more robust with respect to slight departures from a desired model. It uses sequentially accumulated information to detect out-of-control conditions [20]. As shown in Equation 2, the sequence of cumulative sums is computed by adding the difference between the current event value, e_i and a suitable reference value, e.g. the mean μ, to a cumulating sum.

$$CUSUM = \Sigma_{i=1}^{n} e_i - \mu \tag{2}$$

The CUSUM is useful in picking out general trend changes from random noise as noise will tend to cancel out in the long-run (there are just as many positive and negative values of true noise), but a consistent change will show up as a gradual departure from zero in the CUSUM. Therefore, CUSUM can be used for detecting not only sharp changes, but also gradual, changes. Our algorithm for burst detection using CUSUM on coefficients is presented in Table II.

While CUSUM is effective for detecting bursts on raw time series data, one question we are attempting to answer is whether or not CUSUM is a good approach for detecting bursts from transform coefficients. Do the linear transformations we consider result in coefficient values that obscure the burst?

B. Linear Transformations for Time Series Data

A linear transformation uses a function to translate a vector into a new vector that is a linear combination of the original vector. When we use a linear transformation of an n length time series, we are projecting the coordinates of the time series $T\_original$ into a new n dimensional space, T_coef, using n orthogonal basis vectors. The new representation of the data gives us an opportunity to highlight different aspects of the time series that may be beneficial for certain applications. Further, because the transforms are linear with orthogonal basis vectors, addition and scalar multiplication are preserved. The original time series vector $T\_original$ can be reconstructed without introducing error by multiplying the coefficients with the basis vectors and summing them up. As previously mentioned, we will consider the Discrete Fourier Transform (DFT) and Discrete Wavelet Transforms (DWT), both Haar and Daubechies, in this paper.

A Fourier transform uses basis vectors that are defined by the trigonometric functions sine and cosine to generate Fourier coefficients for an arbitrary vector. One of the disadvantages of the DFT is that the transformation produced is a single resolution. In other words, time localization is only captured at the window level. Wavelets allow time series to be viewed in multiple resolutions. At each resolution, the size of the input data is halved. A time series of length 16 has 4 resolutions. Level 4 maintains the highest frequency coefficients and is represented using eight coefficients. Level 3 has four coefficients. The second level has two coefficients. Level 1 has one coefficient. Finally, level 0, which is sometimes referred to as the 'remainder', is also a single coefficient. From this we see that wavelet transforms give gradually refined representations of a time series at different scales. There are many different families of wavelet functions. The difference between the Haar and Daubechies wavelets is the function used to generate the coefficients. Due to space limitations, we refer you to [21] [15] [22] for more details on the Fourier and wavelet transforms.

Referring back to the question posed at the end of the last subsection, we pause to show that CUSUM can be used directly on coefficients to identify bursts. Table III presents burst detection results for each of the linear transforms on different data sets using a window size of 128. [1] We use approximately 20 different time series data sets ranging from ticker to sensor to synthetic in six aggregation experiments to show that our approach calculates bursts at multiple resolutions with the same accuracy as CUSUM applied to the time series directly. While none of these data sets require privacy preservation as described in Section 1, they serve as a representative set of time series from different domains with different burst and signal energy characteristics. Due to space limitations we only show examples of two of the aggregated data sets in Fig. 1, aggregated sensor data collected from 3 different sensors in a room housing a server cluster (Fig. 1(a)) and random walk (Fig. 1(b)) data. Most of the non-synthetic data sets were obtained from the UCR time series data repository [23]. We generated the different synthetic data sets. More details about their parameters and links to the other data sets can be found at [24].

The results show that while DFT is reasonable, the burst detection results using wavelets misses only 1% of the bursts when using all the coefficients except the remainder. We surmise that this results from the multi-resolution property of wavelets. Therefore, as we continue our analysis, we will focus the discussion on wavelets.

Each site calculates the local linear transform for the time series data. The first privacy step here is that the raw time

[1] One disadvantage to our approach compared to applying CUSUM directly on the time series is that our results return the window and resolution the burst is detected in. This gives a bound on the width of the burst, but not the actual width. In contrast, the CUSUM result gives the endpoints of the burst.

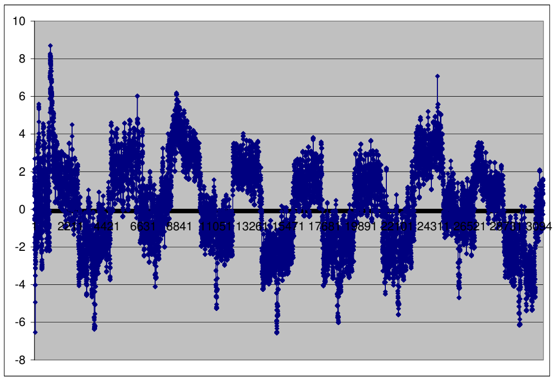
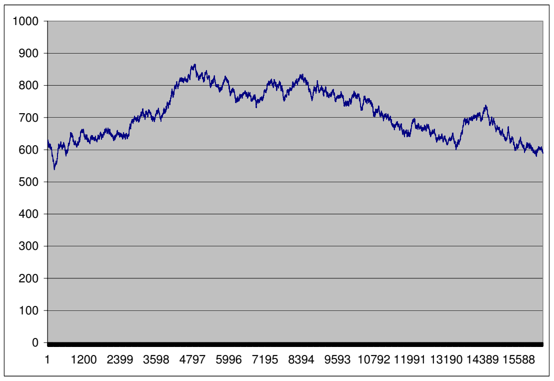

Fig. 1. Example Aggregate Time Series (Left: Sensor Data) (Right:Random Walk Data)

TABLE III
BURST DETECTION RESULTS

Data Set Id	Actual	Haar	Daubechies	Fourier
Stock	160	159	160	156
Sensor	243	242	242	242
Synthetic	133	132	132	132
Mix1	235	234	234	104
Walk	128	128	128	128
Mix2	133	132	132	132

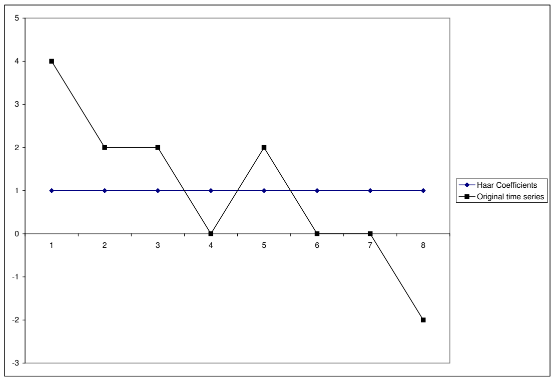

Fig. 2. Comparison of Time Series Values and Transformation Values

series data is not being sent over the network. Suppose an uninformed network intruder was able to capture the values being transmitted. The uninformed network intruder would not realize that the data being transmitted are coefficients. Fig. 2 shows a simple example of raw time series data and its coefficients. We see that there is a difference between the two time series.

DEFINITION 5: We define ts_norm_error to be the *normalized error rate* between two time series. The simple approach to measuring error is using the Euclidean distance between the data points in the two time series. This is a reasonable choice since our comparison involves linear transforms and it has been shown in Parsavals theorem that Euclidean distance is preserved for orthonormal transforms [25]. Because the length of time series varies, we need to standardize the Euclidean distance in order to compare across sets. Therefore, as shown in equation 3, we measure the normalized error by summing up the Euclidean distances at each data point and dividing the sum of the Euclidean distances by n, the number of values in the time series:

$$ts\_norm\_error\{T\_1, T\_2\} = \frac{\sqrt{\sum_{i=1}^{win\_len}(T\_1_i - T\_2_i)^2}}{n} \quad (3)$$

T_1_i is the data point in time series T_1 and T_2_i is the data point in time series T_2. This result is the average error per data point.

The ts_norm_error for the time series in Fig. 2 is 0.61.

Privacy Statement 1: No raw time series data is sent over the network. Without knowledge of the linear transform used, actual data cannot be calculated. The amount of error introduced is the ts_norm_error between E_data_i and E_coef_i.

Even though the uninformed network intruder will not capture or see actual data values, if the informed network intruder captures the values being transmitted, he can reconstruct the original local time series. Therefore, we need to consider additional privacy measures. The first approach is to set win_len smaller than the length of the time series. If $win\_len = n$, then the entire time series is represented using one set of coefficients. Time series reconstruction is straightforward for the network intruder. If there are multiple windows, the intruder will need to determine the length of the window in order to reconstruct the original signal. So we must ask the question, with what probability can the intruder discover the window length?

Because wavelets are being considered, one may assume that the informed network intruder would predict the window size to be a power of two. Therefore, the probability of discovering or guessing the length of the window is $P(win\_len) = 1 \div log(n)$. To ensure that bursts of desired width are searched for, win_len must be at least the width of the maximum burst size determined in step one of the general algorithm. Therefore, if the informed network intruder has some knowledge about the mapping between window size and maximum burst size, the probability may be reduced by some constant factor.

Privacy Statement 2: Privacy is increased if the time series is divided into multiple windows of pre-determined length.

While partitioning the data into windows may be a sufficient level of privacy against the uninformed network intruder, the informed network intruder can still detect the original local time series with a fairly high probability. Therefore, we need to consider different approaches for introducing error into the linear transform coefficients. Some options include: removing subsets of coefficients, perturbing smaller coefficient values by zeroing them, and adding random noise.

Our first option is the removal of coefficients. Suppose that we remove the Level 0 coefficient. Doing so adjusts the location of the time series along the y-axis. Fig. 3 shows an example for the reconstructed time series where the Level 0 coefficient is adjusted and all the other coefficients remain the same. We see a minimal impact on the shape of the time series based on adjusting the Level 0 coefficient for the Haar transform (Fig. 3(a)). A much greater impact is seen in Fig. 3(b). As the value of the remainder increases, it also has more of an impact on the overall shape of the series because the value represents the difference (deviation) between the actual time series and the mother wavelet function. In the worst case, if the Level 0 coefficient is not transmitted and the other coefficients are determined, the envelope of the time series can be determined.

To quantify the chance of an informed network intruder determining this coefficient, we will assume that the real-valued coefficients are bucketed into integer bins. If there are x equally likely bins for the Level 0 (L0) coefficient, then the probability of guessing the Level 0 coefficient right is $P(L0) = 1 \div x$. (Note that without binning $P(L0)$ decreases.) Therefore, if we have multiple windows each with a different Level 0 coefficient, we reduce the probably of reconstructing the original time series to

$$P\{E\_data\} = P(win\_len) \times (nbr\_of\_windows \times P(L0)) \tag{4}$$

We also recall that the burst detection algorithm did not use the Level 0 coefficient. Therefore, we are increasing the level of privacy and not affecting the burst detection accuracy, a rare win, win situation.

Building on the Level 0 result, we consider removing different resolutions of coefficients. Fig. 4 shows an example of this and the resulting time series after reconstruction. It is clear that the specific amount of error introduced is highly dependent upon the time series itself. If the coefficients removed are large values, the impact on the shape of the original time series is much more significant than if the magnitude of the coefficients is smaller. Therefore, we propose a *coefficient removal heuristic* that always removes the Level 0 coefficient and then removes additional levels of coefficients based on the minimum and maximum burst widths specified. Recall our example with 16 coefficients. If the minimum burst width specified is 4 and the maximum is 8, then we can remove the Level 0, 1 and 4 coefficients. Only the Level 2 and 3 coefficients for each window are transmitted.

TABLE IV
COEFFICIENT REMOVAL BURST DETECTION RESULTS

Data Set Id	Actual	Haar	Daubechies	Mean
Stock	160	87	160	160
Sensor	243	242	242	238
Synthetic	132	132	132	122
Mix1	235	234	234	233
Walk	128	128	128	127
Mix2	133	132	132	132

Privacy Statement 3: Removing coefficients introduces error throughout the series. However, in the worst case, the time series envelope can still be determined if the number of windows is one and the magnitude of the coefficients removed is small compared to the ones being maintained. In other words, if replacing the removed coefficients with zero is a reasonable approximation, the shape of the time series can still be discovered.

One may conjecture that calculating a set of means for each window equal to the number of coefficients maintained for each window may be a reasonable approach for burst detection. Table IV compares the burst detection results for our coefficient removal heuristic and mean binning. In this set of experiments, we maintained the same number of coefficients as means. For $win\_len = 128$, we maintained 8 coefficients or 8 means for each window. We see that the burst detection accuracy results are very similar with the exception of the stock data. Daubechies performed well throughout, but Haar did not do as well on the slow changing stock ticker time series as coefficients were removed. The problem was that some large coefficients were removed and since a good percentage of the coefficients for that time series were smaller, the impact on burst detection was an issue. For time series with more variation, the impact is less. It was our hypothesis that when using means, shorter bursts would not be detected and the longer ones would. In reality, the longer bursts were detected, but most of the shorter ones were also. However, the privacy guarantees for the mean binning approach are weaker because an uninformed network intruder can identify the envelope of the time series directly from the values transmitted. All the approaches we have proposed have a higher level of privacy than the mean binning heuristic.

We consider one other perturbation strategy we refer to as our *percentage thresholding heuristic*. This strategy has two parts. First, the Level 0 coefficient is removed. Second, we set a certain percentage of the smallest coefficients in each

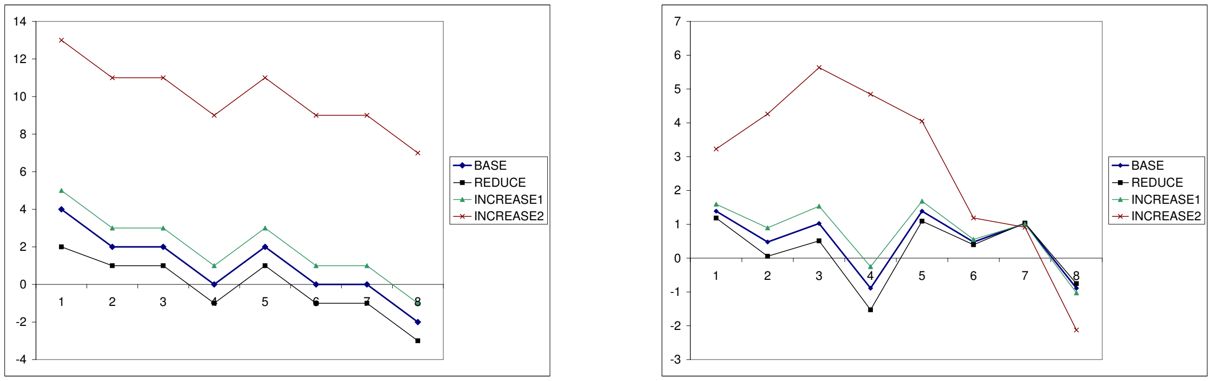

Fig. 3. Adjustment of Level 0 Coefficient (Left: Haar) (Right: Daubechies)

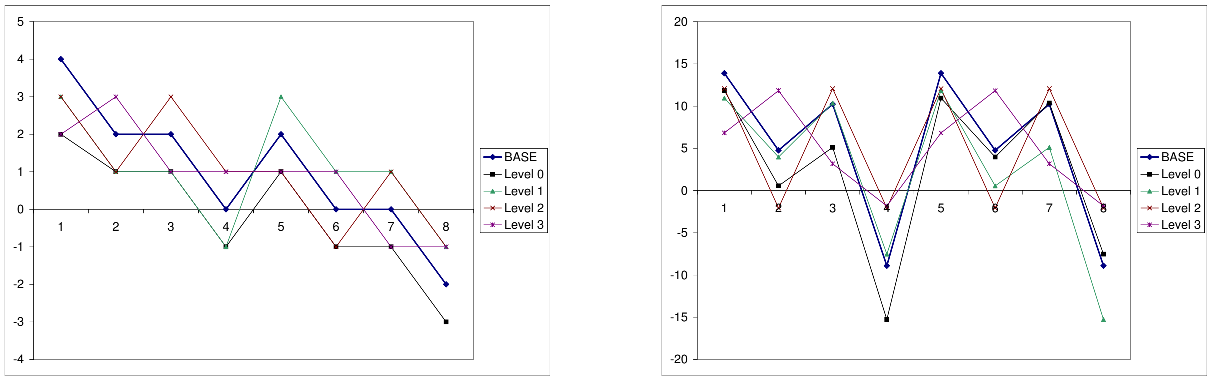

Fig. 4. Remove A Single Level of Coefficients (Left: Haar) (Right: Daubechies)

window to 0. By doing so, we further obscure the original data and in many cases, enhance the burst detection results as illustrated in Table V. In this table, each column represents the percentage of coefficients removed and each cell has the number of bursts detected using Haar coefficients and the number of bursts detected using Daubechies coefficients, respectively. We see that the burst accuracy remains extremely high until 95% of the coefficients are removed. Fig. 5 illustrates an example of network packet data from the UCR Repository. In this example, the smallest 50% of the coefficients at each level were set to zero. Fig. 5(a) is the actual time series and Fig. 5(b) is the time series reconstructed after using percentage thresholding on Daubechies coefficients. We see that the bursts are flattened to some extent, but in general are still visible when compared to the rest of the time series. Because the Level 0 coefficient is missing and a large percentage of actual coefficient values are set to 0, the amount of error introduced is very large in most cases. In the worst case, the Level 0 coefficient is zero and all the coefficients set to zero were already zero. However, in that case, the informed network intruder would have no way of knowing that the signal reconstructed from those values is the actual signal.

TABLE V
PERCENTAGE THRESHOLDING BURST DETECTION RESULTS

Data Set Id	Actual	50%	80%	95%
Stock	160	159, 160	160,159	158, 81
Sensor	243	242, 242	241, 242	240, 197
Synthetic	132	132, 132	132, 132	132, 132
Mix1	235	234,234	234, 234	234, 234
Walk	128	128,128	128, 128	114, 77
Mix2	133	132, 132	132, 132	132, 108

Privacy Statement 4: The amount of privacy is highest using the percentage threshold heuristic and bursts continue to be accurately detected for thresholding percentages above 50%.

The last strategy for coefficient perturbation is the addition of random noise. We address this idea in the next section.

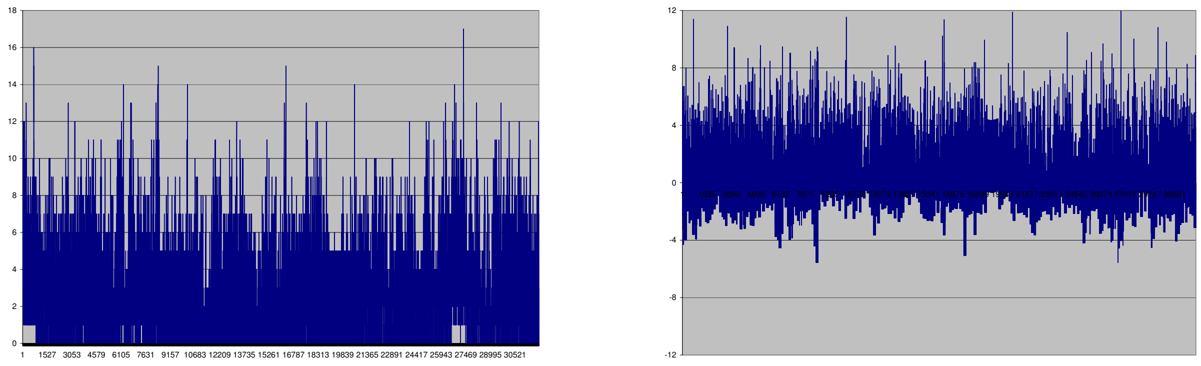

Fig. 5. Percentage Thresholding Comparison (Left: Actual Time Series) (Right: Reconstructed Time Series After Percentage Thresholding)

C. *Secure Aggregation of Distributed Time Series*

Secure Multi-party Computation (SMC) was introduced in [16]. SMC is a cryptography based technique that enables three or more parties to securely compute a function. A computation is considered secure if each party knows only its input and the final result. We will now go through the details of the SMC based on discussions presented in [12] [13]. For proof of correctness and a more detailed discussion refer to [26].

Without a trusted third party, pieces of the computation must take place at each site. During communication, raw data values cannot be sent since knowledge of this can lead to determining the original time series. The data values must be encrypted with a random key. For our problem, we are interested in aggregating the linear transform coefficients. We can accomplish this by using SMC for calculating the sum of the coefficients - this operation is referred to as secure sum [16] [26]. Notice that by aggregating perturbed coefficients instead of the original time series, we add an additional layer of privacy protection. The lossy linear transformation and the secure sum computation are two different privacy preservation operations. If the privacy attained from the lossy linear transformation is sufficient, SMC can be removed from the algorithm and the lossy signals can be sent directly to the leader site. We choose to maintain this step to ensure that the local bursts cannot be detected by semi-honest participants.

A sum is computed securely using the following method for vectors. First, the leader, designated site 1, generates a vector of random numbers, R. R is chosen uniformly from the range $[0 \ldots x]$ such that each element in the aggregate coefficient vector c

$$c = \sum_{k=1}^{t} c_k \tag{5}$$

is known to lie in the range from $[0 \ldots x]$ and c_k is a vector of transformation coefficients for site k. Site 1 adds each element of R to each coefficient and sends vector $(R + c_1 mod x)$ to participant 2. Because the values in R are randomly chosen from the range $[0 \ldots x]$, participant 2 does not learn the actual coefficients, c_1. The remaining sites add their coefficients to the received set of values. Once the final site completes its computation, the leader receives the following result:

$$c = R + \sum_{j=1}^{t} c_j mod x \tag{6}$$

Since the leader knows R, it can subtract the vector R from c, the vector of aggregated coefficients, to get the actual sum. Using secure sum is our approach for adding random noise to the aggregation process. The secure sum algorithm has been extended to consider malicious participants. We refer you to [27] for those details.

A nice feature of linear transforms with orthogonal basis vectors is that the operations of sum and scalar product are preserved and translate directly across different basis vectors. Therefore, if we sum the coefficients of two Haar wavelets and construct a time series from the vector of summed coefficients, the final time series is the same as if the summation occurred on the original time series. This is one of the main properties that enables our aggregate time series bursts to be detected with such high accuracy.

Because this secure computation is not the final burst detection, but rather an intermediate calculation needed for determining the bursts, we must verify that sharing an aggregate sum does not violate privacy. Can one participant derive another participant's time series from this aggregate sum or from intermediate pieces of the aggregate sum calculation?

As previously shown, if the coefficient removal heuristic or the percentage thresholding heuristic are applied, then the probability of the original time series being determined is low even if the coefficients are recovered. If not, then we consider the scenario with the highest probability of privacy violation by a semi-honest participant.

As an example of the worst case scenario, suppose we have 3 parties participating in the computation. Let us also suppose that the coefficients are all positive integers. If each site is given c, the aggregate sum, then each site can remove its coefficients and have the aggregate sum of the remaining two sites. Suppose site 1 does this and also divides the aggregate sum by two to obtain an average value for each coefficient in the vector. Let us also assume that the coefficients of site 1 are higher than the average coefficients of the other two sites and the Level 0 coefficient is close to 0. The probability of determining the coefficients (no privacy) is then

$$P(np) = \frac{1}{\prod_{i=1}^{n}((t-1)!) \times \mu(t-1)_i} \tag{7}$$

The above probability equation has three factors: the number of coefficients (n), the number of sites minus the site attempting to breech the privacy ($t-1$), and the average coefficient value of the other sites ($\mu(t-1)$) assuming site 1 is doing the computation. We can see that even with a relatively small number of coefficients and sites, the probability of determining the coefficients is very small. Also, if the coefficients were discovered for the other sites involved in the computation, without external information, one could not be certain of which site the coefficients belong to.

Privacy Statement 5: Using secure sum adds a layer of random noise to the aggregation process. It can be used in conjunction with any of the heuristics presented.

V. Conclusions

This paper discusses a new area of privacy preserving data mining, burst detection. We have provided an explanation for different levels of privacy that can be attained using linear transforms. Our first contribution was to show that the coefficients from linear transformations could be used directly for burst detection; regeneration of time series values was not necessary. For our data sets, we found multiresolution wavelets were better than Fourier for burst detection from the coefficients themeselves. While the burst detection accuracy was extremely high, the privacy guarantees were not sufficient for informed network intruders and semi-honest participants.

Our next contribution was introducing two heuristics that perturbed the local coefficients: coefficient removal and percentage thresholding. In both cases, we were able to maintain a high burst accuracy rate. We showed, however, that the privacy guarantee was better for the percentage thresholding heuristic. Using this algorithm, small data values are perturbed, but large ones that may be involved with bursts are maintained. In general, Daubechies detected the largest number of bursts consistently across resolutions. As the number of coefficients decreases, Daubechies continues to detect bursts well. As a final layer of privacy for semi-honest participants, we add random noise via a secure sum operation on coefficients during data aggregation.

Overall, CUSUM was an effective choice and bursts were accurately detected directly from the perturbed transformation coefficients. Also, because we used wavelets, both wide and narrow bursts could be detected by searching for bursts at the appropriate resolutions. In the future we want to consider incorporating an aging function or a CUSUM restart function for time series with frequent bursts.

References

[1] T. Bailey, T. Sapatinas, K. Powell, and W. Krzanowski, "Signal detection in underwater sounds using wavelets," *Journal of Americal Statistical Association*, vol. 93, no. 441, pp. 73–83, 1998.

[2] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," in *ACM Symposium on Discrete Algorithms*, 2002.

[3] C. Shahabi, X. Tian, and W. Zhao, "Tsa-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data," *Statistical and Scientific Database Management,*, pp. 55–68, 2000.

[4] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM Press, 2003, pp. 336–345.

[5] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the ACM SIGMOD Conference on Management of Data*. ACM Press, May 2000, pp. 439–450.

[6] J. Vaidya and C. Clifton, "Privacy preserving naive bayes classifier for vertically partitioned data," in *SIAM International Conference on Data Mining*, 2004.

[7] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, July 2002.

[8] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," June 2002.

[9] X. Lin and C. Clifton, "Privacy-preserving clustering with distributed em mixture modeling," *Knowledge and Information Systems*, vol. 8, no. 1, pp. 68–81, 2005.

[10] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM Press, 2003, pp. 206–215.

[11] A. Karr, X. Lin, J. Reiter, and A. Sanil, "Secure regression on distributed databases," *Journal of Computational and Graphical Statistics*, 2004.

[12] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Exploration*, vol. 4, no. 2, pp. 1–7, 2003.

[13] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD Record*, vol. 33, no. 1, pp. 50–57, 2004.

[14] S. Merugu and J. Ghosh, "Privacy-preserving distributed clustering using generative models," in *In The IEEE International Conference on Data Mining (ICDM'03)*, November 2003.

[15] H. G. C. S. Burrus, R. Gopinath, *Introduction to Wavelets and Wavelet Transformations*. Prentice Hall, 1998.

[16] A. C. Yao, "How to generate and exchange secrets," in *IEEE Symposium on Foundations of Computer Science*, 1986.

[17] G. Barnard, "Control charts and stochastic processes," *Journal of he Royal Statistical Society*, vol. B21, pp. 239–271, 1959.

[18] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day, 1976.

[19] W. Shewhart, *Economic control of quality of manufactured product*. New York: D. Van Nostrand, 1931.

[20] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, pp. 100–115, 1954.

[21] E. O. Brigham, *The Fast Fourier Transform and Its Applications*. Upper Saddle, NJ: Prentice Hall, 1988.

[22] A. Jensen and Cour-Harbo, *Ripples in Mathematics*. Berlin: Springer, 2000.

[23] E. Keogh, "The ucr time series data mining archive." University of California, Riverside Computer Science & Engineering Department, http://www.cs.ucr.edu/˜eamonn/TSDMA/.

[24] L. Singh and M. Sayal, "Time series data sets," Georgetown University, Computer Science Department, http://www.cs.georgetown.edu/˜singh/research/data/time_series.html.

[25] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs: Prentice-Hall, 1975.

[26] O. Goldreich, A. Micali, and A. Wigderson, "How to play any mental game," in *ACM Symposium on Theory of Computing*, 1987, pp. 218–229.

[27] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," *SIGKDD Exploration*, vol. 4, no. 2, pp. 12–19, 2002.