Authenticated Data Compression in Delay Tolerant Wireless Sensor Networks

Young-Hyun Oh, Peng Ning, Yao Liu
North Carolina State University
Raleigh, NC 27695
Email: {yoh4, pning, yliu20}@ncsu.edu

Michael K. Reiter
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599
Email: {reiter}@cs.unc.edu

***Abstract*— Delay Tolerant Wireless Sensor Networks (DTWSNs) are sensor networks where continuous connectivity between the sensor nodes and their final destinations (e.g., the base station) cannot be guaranteed. Storage constraints are particularly a concern in DTWSNs, since each node may have to store sensed data for a long period of time due to intermittent communication while continuously collecting data. In this paper, we develop novel techniques to seamlessly authenticate and compress sensed data, addressing the storage concerns in DTWSNs in hostile environments by harnessing both temporal and spatial correlation among sensed data. By appropriately using Message Authentication Code (MAC) and one-way hash functions, our techniques allow a node (including intermediate, non-source nodes) to efficiently compress sensed data without regenerating MACs. We explore the trade-off between data distortion and storage reduction provided by compression so that we can reach the maximum data reduction while maintaining desired data quality.**

I. Introduction

Wireless sensor networks (WSNs) have recently received a lot of attention due to a wide range of potential applications such as warehouse inventory, object tracking, and environment monitoring. A typical WSN is expected to consist of a large number of sensor nodes deployed in a large scale, where the sensor nodes have limited power, storage, communication, and processing capabilities.

In certain sensing applications, it is not feasible to provide real-time transmission of sensed data between WSNs and their final destinations. Such WSNs are referred to as Delay Tolerant Wireless Sensor Networks (DTWSNs) [7]. In such networks, continuous connectivity between the sensor nodes and the data collectors (e.g., mobile sinks) cannot be guaranteed due to intermittent communication. Examples of DTWSNs in practice include ZebraNet [9] and DataMules [15].

Storage constraints are particularly more concerns than energy constraints in DTWSNs, since each sensor node may have to store sensed data for a long period of time due to intermittent communication while continuously collecting data. Data compression is certainly a good candidate to reduce the storage requirement by removing redundancy in stored data while maintaining desired data quality. However, when a DTWSN is deployed in a hostile environment, data integrity must be protected from unauthorized modifications by potential attackers.

In this paper, we address both compression and authentication issues for DTWSNs. Specifically, we develop novel techniques to seamlessly integrate data authentication and compression at the same time. Our contributions in this paper are three-fold:

1) We develop authenticated data compression schemes to allow each node not only to protect data integrity by using Message Authentication Code (MAC), but also to efficiently compress sensed data without regenerating MACs. An important property of the authentication scheme is that an intermediate node (other than the data generator) can also compress the sensed data while reusing the same MAC as long as it follows the correct compression procedure.
2) We take advantage of the redundancy due to temporal and spatial correlation among sensed data, and develop both *authenticated temporal* and *spatial* compression schemes.
3) We also investigate the trade-off between data quality and storage reduction provided by compression. Our schemes give a compression threshold so that each node can reach the maximum data reduction while maintaining desired data quality.

Our security analysis indicates that these schemes can properly defend against external attacks. Moreover, we study the performance of these schemes through simulation using both real and simulated data sets. Our results demonstrate that these schemes can effectively take advantage of the temporal and spatial correlation among the sensed data so that the required storage can be reduced without greatly sacrificing the data quality.

The rest of the paper is organized as follows. Section II discusses the assumptions and threat model. Section III presents the proposed authenticated data compression schemes and their security analysis. Section IV reports the simulation evaluation of the proposed techniques. Section V discusses related work, and Section VI concludes this paper and points out some future research directions.

II. Assumptions and Threat Model

Assumptions: We assume a DTWSN consists of a large number of resource constrained *(regular) sensor nodes* and a few *mobile sinks*, as illustrated in Figure 1. The sensor nodes collect data from the physical environment, while the mobile sinks travel to the network periodically to retrieve the collected data and deliver them to one or a few off-site *data consumers*. Optionally, there may be a few *storage nodes* evenly deployed in the network, and a sensor node may transmit its sensed data to a nearby storage node for intermediate storage via a reliable channel. Both

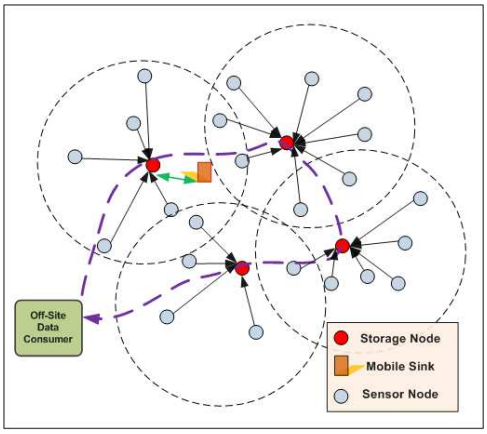

Fig. 1. System Model

regular sensor nodes and storage nodes are stationary. A mobile sink is a powerful device with high computational capacity, no storage limit, long-lasting power, and mobility. Data consumers are applications that will process the data retrieved by mobile sinks.

We assume that sensor nodes and storage nodes can communicate with each other when necessary, possibly with short delays. However, continuous connectivity between nodes deployed in the network and the mobile sinks cannot be guaranteed. Thus, sensor nodes (and storage nodes when they are present) may have to store sensed data for a long period of time while continuously collecting data. Finally, we assume that the location information of sensor nodes is known when it is needed.

We assume that each sensor node shares a master secret key with the data consumer. Furthermore, we assume that nodes in the network can establish a pairwise key using existing pairwise key establishment schemes (e.g., [12], [13], [21]). Thus, node-to-node communication in the network can be authenticated.

Threat Model: We assume that the adversary has powerful devices such as laptops and workstations. The adversary can eavesdrop, inject, drop, and modify packets in the network, since wireless communication is broadcast-based. Under these assumptions, the adversary can manipulate the data to its own advantage and send it to storage nodes or mobile sinks. However, we assume that the adversary hasn't compromised the sensor nodes that report the sensed data. In other words, the adversary doesn't know the authentication key to forge malicious but authenticated data.

Design Goal: Our goal is to develop efficient mechanisms for integrated data authentication and compression, so that regular sensor nodes, and storage nodes when they are present, can authenticate the sensed data, possibly after compression, to the data consumer. Moreover, compression of sensed data should be controlled so that the data after compression retains enough information for the data consumer. Detecting compromised nodes that provide malicious data is a related but orthogonal problem. We assume there are other mechanisms for this (e.g., [20]), but do not address it in this paper.

III. Proposed Schemes

We consider two practical scenarios, where there are (1) temporal correlation in data sensed by individual sensors over time, and (2) spatial correlation in data sensed by sensors physically close to each other. Accordingly, we develop *authenticated temporal* and *authenticated spatial* compression schemes for these scenarios, respectively. Due to temporal and spatial correlation among sensed data, each scheme achieves great storage reduction by removing redundancy, while at the same time maintains a pre-defined data quality. We also prove that our schemes protect integrity of data from the adversary through security analysis.

We note that these two schemes can be integrated when there are both temporal and spatial correlations. Since the integration is fairly straightforward, we do not explicitly discuss it in this paper.

A. Authenticated Temporal Compression

A sensor node is supposed to continuously sense the physical environment, collect data, and store them until transmitting them to a receiving node. However, in a DTWSN, there is no guarantee for the sensor node to establish a communication channel to transmit the sensed data. Thus, in certain cases, the sensor node may reach the storage limit before it can transmit the sensed data, though it is still necessary for it to collect data continuously.

Data compression is an ideal tool to address this problem. In lossy compression, a sequence of data is typically transformed into the frequency domain through, for example, Discrete Cosine Transformations (DCT) or wavelet transformations. Due to the correlation among the data, not all the transformed data in the frequency domain retain the same amount of information about the original data. Indeed, some less important data items can be dropped without significantly affecting the overall data quality. More importantly, these transformed data items can be ordered and gradually dropped to allow a gradual reduction of storage requirement and at the same time a graceful degradation in data quality. This is ideal for nodes in a DTWSN; a node can gradually compress the stored data to give space to newly collected data. When it can establish a communication channel to the intended receiver, the sensor node can transmit the best quality data it can afford.

Our basic idea is to provide an efficient authentication method for the transformed data items. We would like to have an authentication method that can be computed once, and reused for gradual compression with little additional overhead, even by intermediate nodes that do not know the authentication key. This capability is not available in a direct application of Message Authentication Code (MAC), since any change in the authenticated data will lead to a change in its MAC.

Our authenticated temporal compression scheme takes advantage of the temporal correlation among sensed data. We divide the time line into time intervals, and process the data items sensed in each time interval individually. Our scheme has five steps: transformation, MAC generation,

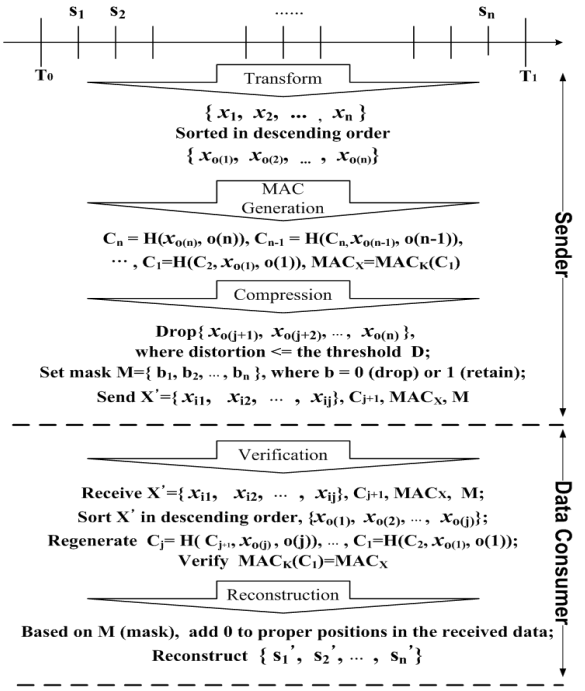

Fig. 2. Authenticated temporal compression

compression, verification, and reconstruction. The first two steps take place on sensor nodes, the compression step may happen either on the original sensor or an intermediate storage node, while the last two steps occur on the data consumer. Figure 2 illustrates these steps.

1) Transformation: Each sensor continuously collects data from the physical environment. Assume the sensor node has collected a sequence of data for one sensing period (e.g., from T_0 to T_1), represented as a vector $S = (s_1, s_2, \cdots, s_n)^T$. The sensor node then applies a transformation W (e.g., DCT, wavelet transformations) to S. (Note that the choice of transformation is independent of our approach.) Thus, the node gets a sequence of transformed data items, represented as a vector

$$X = W(S) = (x_1, x_2, \ldots, x_n)^T.$$

Note that in typical transformations used in compression, such as DCT and wavelet transformations, $X = W(S)$ is simply $X = WS$, where W is an $n \times n$ matrix.

2) MAC Generation: The key requirement of the MAC generation step is that the MAC computed for each time interval should remain the same even if some transformed data items are discarded during compression. This requirement is critical in reducing the overhead introduced by authentication.

We propose a chaining technique using a cryptographic hash function H (e.g., SHA-1). In the compression step, we follow the way in which typical compression schemes use to determine what data to drop during compression. That is, the item with the smallest absolute value in X contributes the least to the quality of compressed data, and gets discarded first. Accordingly, the sensor node sorts the absolute values of the transformed data items in X in descending order before applying the chaining technique. Let $x_{o(i)}$ be the i-th element in the sorted list. Given the transformed data $X = (x_1, x_2, ..., x_n)^T$, the sensor node performs the following operations:

$$\begin{aligned} C_n &= H(x_{o(n)}, o(n)), \\ C_{n-1} &= H(C_n, x_{o(n-1)}, o(n-1)), \\ &\ldots \\ C_1 &= H(C_2, x_{o(1)}, o(1)). \end{aligned}$$

The sensor node then computes the MAC of the sequence of data by $MAC_X = MAC_K(C_1)$, where K is the secret key shared between the sensor node and the data consumer. Note that the position of x_i in X is implicitly represented in X but explicitly included in the computation of MAC_X.

3) Compression: The compression step can be performed by either the original sensor node or an intermediate storage node. The compression node discards the items with the smallest absolute values in the transformed data vector X. We use a masking vector M, which is a vector with 1 and 0 representing the corresponding value in X retained and discarded, respectively. This masking vector may be stored using a lossless compression algorithm such as Run-length Encoding (RLE) compression.

There are two major issues in the compression step: (1) how to avoid recomputing the MAC while discarding non-critical data, and (2) how to avoid compressing too much to provide good enough data to the data consumer.

The first issue is easy to address due to the chaining technique used in MAC generation. When there is no compression, the data presented by the sensor node is $X = (x_1, x_2, \cdots, x_n)^T, MAC_X$. Verification can be easily performed by repeating the MAC generation process and comparing the result with MAC_X. When it is necessary to drop a few data items, for example, $x_{o(j+1)}, \cdots, x_{o(n)}$, the compression node simply computes

$$\begin{aligned} C_n &= H(x_{o(n)}, o(n)), \\ C_{n-1} &= H(C_n, x_{o(n-1)}, o(n-1)), \\ &\ldots \\ C_{j+1} &= H(C_{j+2}, x_{o(j+1)}, o(j+1)). \end{aligned}$$

The compression node discards $x_{o(j+1)}, \cdots, x_{o(n)}$, and sets 0 at their corresponding positions in the masking vector M. Thus, the data presented by the compression node is

$$X' = (x_{i_1}, x_{i_2}, \cdots, x_{i_j})^T, C_{j+1}, MAC_X, M.$$

The compression can continue if more data items need to be discarded. This allows compression to be performed gradually when there is a need to further reduce the storage. Note that a summary of $x_{o(j+1)}, \cdots, x_{o(n)}$ has already been contained in C_{j+1}. With the above data items, a receiver can easily verify MAC_X. For example, suppose the transformed data vector by a sensor node is $X = (5, 3, 1, 4, 0)$ and the compression node can drop the items $(0, 1)$ in X while maintaining the pre-defined data quality. Then the transformed data vector is $X' = (5, 3, 4)$ with the masking vector $M = (1\ 1\ 0\ 1\ 0)$. The original position of each remaining data item in X can be easily derived using its current position in X' and the masking vector M.

This compression process avoids recomputing the MAC from scratch. It requires light computation; it takes approximately two hash operations for each data item dropped from X, which can be efficiently done on sensor nodes.

Now let us address the second issue, how to avoid compressing too much. We use a distortion threshold to represent the data quality desired by the data consumer. In other words, given a distortion threshold D, the distortion between the compressed data and the original data should not exceed D for the compressed data to be useful for the data consumer. Formally, given the original sequence of data S, the sequence of compressed data S', and the distortion threshold D, the data recovered through decompressing S' should satisfy the following condition:

$$\|S - S'\| = \delta \leq D, \tag{1}$$

where $\|Z\| = \sqrt{z_1^2 + z_2^2 \ldots + z_n^2}$ is the L_2 norm of an n-dimension vector $Z = (z_1, z_2, \cdots, z_n)$.

In general, to see if a compression action will exceed the distortion threshold, one will have to reconstruct the data from the compressed data and verify if the result has exceeded the threshold. Specifically, the compression node needs to compute

$$\delta = \|S - S'\| = \|W^{-1}(X) - W^{-1}(X')\|,$$

where X and X' are the transformed data vectors before and after the compression, and W^{-1} is the inverse transformation used to reconstruct data vector S' from X'. The node then needs to compare δ with the distortion threshold D to determine if the compression is acceptable or not.

Fortunately, in a typical transformation used in compression (e.g., DCT and wavelet transformations), the transformation function can be represented as $W(S) = WS$, where W is an $n \times n$ orthogonal matrix. As Lemma 1 shows below, the distortion due to compression can be computed without performing the inverse transformation.

Lemma 1: Suppose $X = WS$ and $X' = WS'$, where W is an $n \times n$ transformation matrix. If W is an orthogonal matrix (i.e., $W^T W = I$), then $\|X - X'\| = \|S - S'\|$.

Proof:

$$\begin{aligned} \|\mathbf{X} - \mathbf{X'}\| &= \|\mathbf{WS} - \mathbf{WS'}\| = \|\mathbf{W}(\mathbf{S} - \mathbf{S'})\| \\ &= \{\mathbf{W}(\mathbf{S} - \mathbf{S'})\mathbf{W}(\mathbf{S} - \mathbf{S'})\}^{\mathbf{1/2}} \\ &= \{(\mathbf{S} - \mathbf{S'})\mathbf{W^T W}(\mathbf{S} - \mathbf{S'})\}^{\mathbf{1/2}} \\ &= \{(\mathbf{S} - \mathbf{S'})(\mathbf{S} - \mathbf{S'})\}^{\mathbf{1/2}} = \|\mathbf{S} - \mathbf{S'}\|. \end{aligned}$$

■

According to Lemma 1, the compression node can simply compute the distortion due to discarding certain transformed data items using the transformed data directly. This removes the need for inverse transformation. Moreover, the compression node can incrementally compute the distortion due to only keeping the first x transformed data items, where x runs from 1 to n, and identify the threshold at which the distortion will exceed D.

4) Verification: Assume that the data consumer receives the data items

$$X' = (x_{i_1}, x_{i_2}, \cdots, x_{i_j})^T, C_{j+1}, MAC_X, M$$

through the mobile sink. The data consumer first recovers the position of each data item in the original X by using its current position in X' and the masking vector M. The data consumer then sorts the absolute values of the data items in X' in the descending order, where $x_{o(i)}$ is the i-th data item in X' and $o(i)$ is the position of $x_{o(i)}$ in the original X. The data consumer computes

$$\begin{aligned} C_j &= H(C_{j+1}, x_{o(j)}, o(j)), \\ C_{j-1} &= H(C_j, x_{o(j-1)}, o(j-1)), \\ &\ldots \\ C_1 &= H(C_2, x_{o(1)}, o(1)). \end{aligned}$$

as well as $MAC_K(C_1)$. If $MAC_K(C_1)$ is the same as the received MAC_X, the data items are authenticated.

5) Reconstruction: Based on the masking vector M, the data consumer adds 0 to proper positions in X'. Then, the data consumer performs a reverse transformation $S' = W^{-1}(X')$ to reconstruct the sensed data in the time domain.

6) Security Analysis: In the authenticated temporal compression scheme, the final MAC_X is computed from C_1 using a key K shared between the sensor node and the data consumer. Thus, without the knowledge of K, the adversary will not be able to use a different C_1 to forge a valid MAC_X. Moreover, it is easy to see that the hash value C_1 is computed from all the data items $x_1, \cdots, x_n$ and their respective positions in X. Any changes in $x_1, \cdots, x_n$, including re-arranging their order, will cause a change in C_1. Similarly, any change in the masking vector M will lead to the change in the positions of the data items, thus leading to a change in C_1. Due to the pre-image resistance property of a cryptographic hash function, it is computationally infeasible for the adversary to forge any data item and still be able to obtain the same C_1. Thus, the adversary will not be able to generate a MAC_X for forged data. The adversary may over-compress the data, but the effect will not be worse than simply corrupting the data. The adversary may also attempt to replay previous messages. However, a standard solution such as a sequence number can defeat such attacks.

B. Authenticated Spatial Compression

The authenticated spatial compression scheme is intended for a storage node to authenticate and compress data collected by the nearby sensor nodes by exploiting the spatial correlation among them.

We first consider a special situation in which sensor nodes are nicely arranged in regular grid intersections. This assumption allows us to view the sensed data by these nodes as different pixels in an image. Thus, we can apply transformations used by 2D compression algorithms (e.g., 2D DCT or 2D wavelet transformations) at the storage node. We then extend the result to randomly deployed sensor networks by using interpolation. Specifically, the storage node first uses interpolation to transform data

sensed at random locations to values that could have been sensed at artificial, regular grid locations, and then applies techniques developed for the special situation to perform authenticated compression. We assume the sensor nodes know their locations, and provide and authenticate their locations with the sensed data. In both cases, the storage node finds a compression threshold that can reach the maximum data reduction while maintaining desired data quality. Note that a reliable channel between a sensor node and the storage node is important due to data loss could affect the data distortion for both cases.

1) Grid-based Deployment: In a grid-based deployment, sensor nodes are deployed at regular points and all sensor nodes form a grid layout. A storage node is also deployed to store data received from these nodes.

Assume that the storage node has received a sequence of data items S sensed by these nodes at a certain time. The storage node then applies a 2D transformation W (e.g., 2D DCT, 2D wavelet transformations) to S and obtains $\mathbf{X} = W(\mathbf{S})$.

As discussed earlier, the data item with the smallest absolute value in $\mathbf{X}$ contributes the least to the quality of the compressed data, and thus gets discarded first. We use a masking matrix, which is also an $m \times m$ matrix with 1 and 0 representing the corresponding value in $\mathbf{X}$ retained and discarded, respectively. This masking matrix may be stored using a lossless compression algorithm such as RLE compression.

Based on the values of elements in $\mathbf{X}$, we can sort them and generate the MAC in the same way as in authenticated temporal compression with the key shared between the storage node and the data consumer. The verification of the compressed data and the determination of the compression threshold can be done in the same way as well. We omit the details here.

An important issue is that the locations of the sensors must be included and authenticated along with the compressed data. Keeping the initial deployment locations is not sufficient to deal with potential physical attacks, in which the adversary may move the sensor nodes to different locations. Thus, only keeping the initial locations will lead to incorrect conclusions when there are physical attacks.

Note that this requirement is not unique to our authenticated spatial compression scheme. In many sensor network applications, locations are critical in the interpretation of sensed data, and any sensor network application that requires sensor locations will have to keep and authenticate sensors' locations to deal with the above physical attacks.

2) Random Deployment: In practice, it is often not possible to uniformly deploy sensor nodes at regular grid points due to challenges faced by the deployment process such as terrain conditions. In the following, we extend the result from the grid-based sensor deployment to randomly deployed sensor networks.

Figure 3 illustrates the treatment of randomly deployed sensor networks. Assume that the storage node receives a sequence of data items sensed by n nearby sensor nodes, denoted $\mathbf{S} = (s(x_1, y_1), s(x_2, y_2), \cdots, s(x_n, y_n))$,

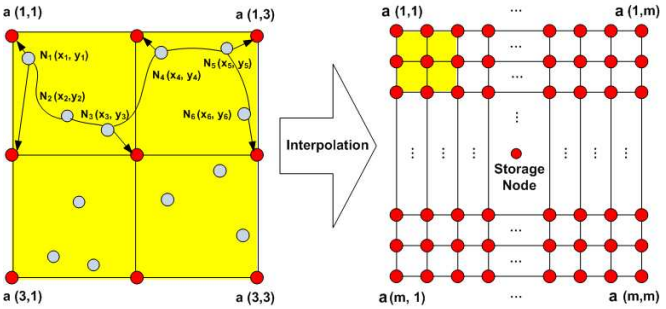

Fig. 3. Random sensor deployment and interpolation

where $s(x_i, y_j)$ represents the data sensed by the node at location (x_i, y_j). The storage node then uses interpolation to transform $\mathbf{S}$ to values that could have been sensed at artificial, regular grid locations. Specifically, we assume a simple interpolation function $f(x, y) = c_1 x + c_2 y + c_3$. For each artificial location, the storage node finds three real sensor nodes that are closest to the artificial location but not on the same line. (Note that having three nodes on the same line will not lead to a successful interpolation.) The storage node then uses the locations and the sensed values at the three real sensor nodes and the function $f(x, y)$ to calculate the value that could have been sensed at the artificial grid location. Alternatively, the storage node may take more than three sensor nodes and use a Minimum Mean Squared Error (MMSE) estimator to get a better estimate of the artificial data values.

Using the interpolation function, the storage node creates an artificial data matrix $\mathbf{A}$ for regular grid points in the grid layout. The storage node then applies a 2D transformation W (e.g., 2D DCT or wavelet transformations) to $\mathbf{A}$ and obtains $\mathbf{X} = W(\mathbf{A})$. As discussed earlier, the storage node sorts $\mathbf{X}$, generates the MAC, and uses the masking matrix M in the same way as in the grid-based sensor deployment.

For determining the stopping point of compression, the storage node additionally needs to reconstruct the original data from the interpolated values at artificial grid points. As in Equation (1), we use a distortion threshold δ to represent the data quality D desired by the data consumer. The threshold δ is the difference between the original sequence of data S and the sequence of the compressed data S'. Since the storage node compresses the values at the artificial points rather than the original values, the storage node needs to reconstruct the original data before it computes the threshold δ. Thus, the storage node computes

$$\delta = \|S - S'\| = \|f^{-1}(W^{-1}(X)) - f^{-1}(W^{-1}(X'))\|,$$

where X and X' are the transformed data vectors before and after the compression, W^{-1} is the inverse transformation used to reconstruct artificial data matrix A' from X', and f^{-1} is the inverse interpolation function used to reconstruct data S' from A'. Through these steps, the storage node determines the compression threshold that can reach the maximum data reduction while maintaining desired data quality.

The verification of the compressed data X' and the masking matrix M by the data consumer can be done in

the same way as in the grid-based deployment. If they are authenticated, the data consumer adds 0 to the proper positions in X'. Then, it performs the inverse transformation $A' = W^{-1}(X')$ to reconstruct the values for artificial grid points. Finally, the data consumer can reconstruct the values at the original locations by using interpolation again. This time, for each real sensor location, the data consumer uses three or more artificial sensors that are closest to the real sensor location.

3) Security Analysis: In the authenticated spatial compression, sensor nodes and storage nodes can establish a pairwise key using existing pairwise key establishment schemes (e.g., [12], [13], [21]). Thus, the communication between sensor nodes and storage nodes in the network can be authenticated. Following the same argument for authenticated temporal compression, without knowing the authentication key, the adversary will not be able to compute a valid MAC, and any modification to the compressed data will be detected. Moreover, replay attacks can be defeated by using a standard solution such as a sequence number.

Another threat is the potential physical attacks in which the adversary moves sensor nodes to different locations. However, in the authenticated spatial compression scheme, the locations of sensors are provided and authenticated along with the compressed data. Thus, such physical attacks will not be effective.

Our authenticated spatial scheme cannot deal with the situation where the storage nodes are compromised. However, as we assumed in our threat model (see Section II), we assume that there are other mechanisms to detect compromised nodes (e.g., [20]).

IV. Simulation Results

We evaluate our authenticated temporal and spatial compression schemes through simulation. We are interested in understanding the trade-off between data distortion and storage reduction provided by compression. Specifically, our simulation is focused on demonstrating that our schemes give a compression threshold so that each sensor node can reach the maximum storage reduction while maintaining desired data quality.

In our simulation, we use Haar wavelet transformation [17] and DCT transformation [3] to transform sensor data in the time (or space) domain to the frequency domain. As discussed earlier, both transformation functions have the orthogonal property, and as a result, the distortion due to compression can be computed without performing the inverse transformation. This greatly saves the computational overheads on sensor nodes.

We use the run-length encoding (RLE) algorithm to compress the masking vector (or matrix), which consists of consecutive zeros or ones. RLE is a lossless and simple data compression algorithm, which is good for many consecutive data elements in a vector.

A. Simulation Data Sets

In our simulation experiments, we use both real and simulated data sets to examine the proposed schemes.

Real Data Set: To examine how our authenticated data compression schemes perform in a real-world deployment, we test them using the data set from the GoMOOS project [1]. GoMOOS is a working prototype for a regional ocean observing system. Special sensor nodes, called *buoys*, were deployed along with the coastline of the Gulf of Maine to sample meteorological data. In our simulation, we use the underwater temperature data collected by ten buoys from 2007-08-13 17:00:00 UTC to 2007-09-04 00:00:00 UTC. This data set exhibit temporal locality, but they do not display spatial locality due to the locations of buoys. Thus, we only use this data set for the authenticated temporal compression

Simulated Data Set: In addition to the real data set, we also use an energy model [8] to simulate values possibly collected by sensors. We assume that an energy source at location (x_s, y_s) emits a signal, which is measured by sensor nodes deployed in the network. The signal strength emitted by the energy source decays as the distance to sensor nodes increases. If the constant decay coefficient is K, the signal strength measured by the sensor node i is $E_i = \frac{C}{(d_i)^K} + r_t$, where C is the energy emitted by the source node, d_i is the distance from the node i at location (x_i, y_i) to the energy source (i.e., $\sqrt{(x_s - x_i)^2 + (y_s - y_i)^2}$), and r_t is the random variation of the signal over time, which follows a normal distribution $N(\mu, \sigma^2)$ [8]. The value K is typically from 2.0 to 5.0 based on the environment. In our simulation, we follow the suggestion in [5] and use a value of $K = 3.0$.

B. Authenticated Temporal Compression

We evaluate the authenticated temporal compression using both the real data set and the simulated data set. For the real data set, we extract 1,024 data items from each buoy, which samples the hourly temperature 1 meter underwater. For the simulated data set, we generate 1,024 data items by using the energy model for one time period.

We explore the relationship between data distortion and storage reduction on the authenticated temporal compression. We use the *storage reduction ratio* to measure the saving in storage. Specifically, we first compute the storage overhead by $H = \frac{New}{Original} = \frac{(O_n - N_m)Bytes + RLE_n Bytes}{O_n Bytes}$, where O_n is the number of transformed data items for each item, N_m is the number of transformed data after data compression, and RLE_n is the number of bytes to encode the masking vector using RLE compression. Then we compute the storage reduction ratio by $R = 1 - H$.

We use *distortion ratio* to measure the degree of distortion between the compressed and the original data. Specifically, the distortion ratio is measured by $d = \frac{\|S - S'\|}{\|S\|}$, where S and S' are the vectors representing the original and the compressed data values, respectively.

Figure 4 illustrates the relationship between the average data distortion and storage reduction for the real and the simulated data sets, respectively. Overall, the graphs for

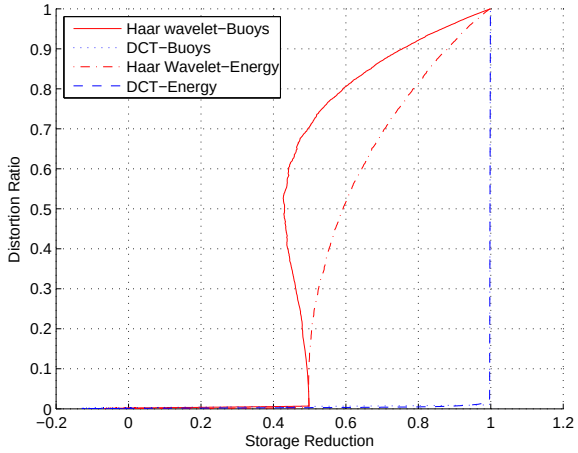

Fig. 4. Temporal distortion degree and storage reduction ratio

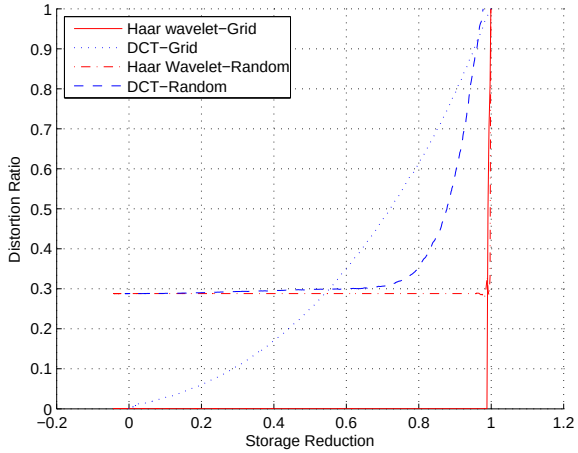

Fig. 5. Spatial distortion degree and storage reduction

both data sets are very similar. This demonstrates that our simulated data set has similar temporal correlation to the real data set.

Figure 4 shows that the distortion ratio for the Haar transformation is close to zero until the storage reduction is close to 0.5 in both data sets. After that, the distortion ratio for the Haar transformation starts to drastically increase. However, the distortion ratio for the DCT transformation remains less than 0.02 until the storage reduction reaches almost 1 in both data sets. This is because the DCT transformed data is almost all concentrated on low frequencies.

Our simulation results demonstrate that if the sensed data is highly correlated with each other in the time domain, both the Haar and the DCT transformations can lead to good storage reduction without hurting the data quality much. Moreover, the DCT transformation can lead to better storage reduction than the Haar transformation for the same expected data distortion.

C. Authenticated Spatial Compression

We evaluate the authenticated spatial compression using the simulated data set. The real sensor data cannot be used due to the lack of spatial coverage. We apply our schemes to two deployment scenarios. First, we generate 1,024 simulated data items for the grid-based sensor deployment using the energy model function. The simulated data set represents the signal strength measured by sensor nodes at 1,024 grid intersections in the area. The storage node receives sensed data from these nodes at a certain time.

For random sensor deployment, we simulate 1,024 values sensed by sensor nodes at random locations using the energy model. The storage node deployed in the network stores data items received from these sensor nodes. The storage node then uses interpolation to generate 1,024 values at the artificial grid locations.

The storage node uses both 2D Haar and 2D DCT transformations to generate the transformed data matrix. We use the same metrics, storage reduction ratio and distortion ratio, as in the evaluation of temporal compression.

Figure 5 illustrates the relationship between the distortion ratio and the storage reduction ratio for both grid-based and random deployments. For the grid-based deployment, the distortion degree for the 2D Haar transformation remains close to zero until the storage reduction ratio is almost 1. The distortion ratio for the 2D DCT transformation is almost proportional to the storage reduction ratio.

For the random deployment, both the 2D Haar and the 2D DCT transformations lead to a distortion ratio at about 0.3 even with small storage reductions. This is mainly due to the error introduced by the interpolation process. Fortunately, the distortion ratio remains around 0.3 as the storage reduction ratio grows, until the storage reduction reaches 0.6 for the DCT transformation, and close to 1 for the Haar transformation. This result indicates that if certain data distortion is tolerable, both transformations can help reduce the storage requirement significantly. Moreover, the 2D Haar transformation demonstrates stronger ability to reduce the storage requirements for the same expected data distortion ratio. Thus, the 2D Haar transformation is a better choice for authenticated spatial compression than the 2D DCT transformation.

D. Computation, Memory, and Communication Overhead

We now briefly present computation, memory, and communication overhead of our schemes as follows:

Computational Overhead: Assume that N is the total number of data items sensed by a sensor node in a given time interval t_w, where each data item is n-bit length. The approximate total execution time T is as follows:

$$\mathbf{T} = T_{MAC} + N \times (T_H + T_D + T_{algo}/c)$$

where T_{MAC} is the time for generating MAC, T_H denotes for a cryptographic hash function (e.g., SHA-1), T_D is for computing data distortion, and T_{Algo} is for computing the transformation algorithm. T_{MAC} and T_H take $3,636\mu s$ for SHA-1 and $1,473\mu s$ for MD5 to process single data item with the input size of 512bits in ATmega128L on MICAz [2] [4]. T_D takes three clock cycles (i.e., about $0.406\mu s$) to process single data item (i.e., one clock cycle for an addition and two clock cycles for a multiplication with 7.3728 MHz clock frequency in ATmega128L [11]). T_{Algo} is small in WSNs, since performing fast DCT, which

does not require floating-point operations, on an 8×8 block (i.e., c is 64) takes $1,394.16\mu s$ (i.e.,11,153 cpu cycles) [11].

Memory Overhead: T_{MAC} and T_H require $n \times (N + 1)bits$ in memory. The memory requirement for T_D is $n \times Nbits$. And T_{Algo} takes $n \times (N/c)bits$. Thus, the total memory requirement of our schemes is $M = n \times (2N + 1 + N/c)bits$.

Communication Overhead: In the nature of data compression, our schemes reduce the communication overhead, which simultaneously implies the less energy consumption ([10] suggests that data communication is more expensive than computation).

V. Related Work

The delay tolerant network (DTN) architecture is originally designed for the interplanetary Internet [6]. However, it is also applicable to other types of networks that suffer from the lack of continuous connectivity. The concept of DTWSN was first proposed in [7], in which WSNs are deployed in mobile and extreme environments.

Several DTWSN applications have already been used. In ZebraNet [9], custom tracking collars (attached to zebras) collect their mobility pattern and report collected data when they pass within the radio communication range of a mobile base station. In DataMules [15], a mule periodically visits sensor nodes and collects data by using a non-interactive message store-and-forward service.

Data aggregation and data compression are different solutions to address the storage concern in DTWSNs. In secure data aggregation schemes [14], [18], [19], the base station can estimate the original data sensed by sensor nodes within a certain range. In these schemes, the base station cannot recover the original data for each sensor node due to the nature of aggregation algorithms (e.g., average function).

However, in our schemes, the base station can recover the raw data with a very high precision for each sensor node. Since our schemes provide lossy compression (e.g., Discrete Cosine Transform (DCT) [3] and wavelet compression [16], [17]), our schemes allow higher compression ratio with imperceptibly small inaccuracies in decompressed signals in WSNs. In this paper, we address the integration of authentication and lossy compression for DTWSNs.

VI. Conclusion

In this paper, we developed novel techniques to seamlessly authenticate and compress sensed data in DTWSNs. By taking advantage of the inherent temporal and spatial correlation among the sensed data, our techniques allow sensor nodes that face storage pressure to sacrifice the data quality within a tolerable range to exchange for space for continuous sensing. The novel authentication technique allows any node to compress the sensed data without knowing the authentication key. Our exploration of the tradeoff between data distortion and storage reduction gives a compression threshold so that our schemes can reach the maximum data reduction while maintaining desired data quality. We evaluate our schemes through simulation; our results demonstrate that these schemes provide significant storage reduction for typical sensor network applications.

In our future work, we will explore and compare other transformations, and hopefully identify the best transformation for different types of sensing applications.

Acknowledgment

This work is supported by the National Science Foundation under grants CNS-0721424 and CAREER-0447761.

References

[1] Gulf of maine ocean observing system. `http://www.gomoos.org/`.

[2] MICAz: Wireless measurement system. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transaction Computers*, pages 90–93, January 1974.

[4] Y. Choi, J. Kang, and D. Nyang. Proactive code verification protocol in wireless sensor network. In *International Conference on Computational Science and Its Applications (ICCSA 2007)*, pages 1085–1096, 2007.

[5] T. Clouqueur, P. Ramanathan, K. K. Saluja, and K. Wang. Value-fusion versus decision-fusion for fault-tolerance in collaborative target detection in sensor networks. In *Fusion 2001 Conference*, 2001.

[6] V. Cerf et. al. Interplanetary internet (IPN): Architectural definition. `http://www.ipnsig.org/reports/memo-ipnrg-arch-00.pdf`.

[7] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the SIGCOMM (SIGCOMM '04)*, August 2004.

[8] M. Hata. Empirical formula for propagation loss in land mobile radio services. *In IEEE Transactions on Vehicular Technology*, 29:317–325, 1980.

[9] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107, October 2002.

[10] N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, April 2005.

[11] D. Lee, H. Kim, S. Tu, M. H. Rahimi, D. Estrin, and J.D. Villasenor. Energy-optimized image communication on resource-constrained sensor platforms. In *Proceedings of 6th International Conference on Information Processing in Sensor Networks (IPSN 2007)*, pages 216–225, 2007.

[12] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 52–61, October 2003.

[13] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(1):41–77, February 2005.

[14] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, Nov 2003.

[15] R. Shah, S. Roy, S. Jain, , and W. Brunette. Datamules: Modeling a three-tier architecture for sparse sensor networks. *In Elsevier adhoc networks journal*, 1:215–233, 2003.

[16] R.S. Wagner, S. Du, and A. Cohen. An architecture for distributed wavelet analysis and processing in sensor networks. In *IPSN*, April 2006.

[17] J. Walker. A primer on wavelets and their scientific applications. Chapman and Hall, 1999.

[18] K. Wu, D. Dreef, B. Sun, and Y. Xiao. Secure data aggregation without persistent cryptographic operations in wireless sensor networks. *Ad Hoc Networks*, 5(1):100–111, 2007.

[19] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 11, July 2008.

[20] Q. Zhang, T. Yu, and P. Ning. A framework for identifying compromised nodes in sensor networks. *ACM Transactions on Information and Systems Security*, 11(3):1–37, 2008.

[21] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 62–72, October 2003.