Secure Provenance Transmission for Streaming Data

Salmin Sultana, Mohamed Shehab, *Member*, *IEEE*, and Elisa Bertino, *Fellow*, *IEEE*

Abstract—Many application domains, such as real-time financial analysis, e-healthcare systems, sensor networks, are characterized by continuous data streaming from multiple sources and through intermediate processing by multiple aggregators. Keeping track of data provenance in such highly dynamic context is an important requirement, since data provenance is a key factor in assessing data trustworthiness which is crucial for many applications. Provenance management for streaming data requires addressing several challenges, including the assurance of high processing throughput, low bandwidth consumption, storage efficiency and secure transmission. In this paper, we propose a novel approach to securely transmit provenance for streaming data (focusing on sensor network) by embedding provenance into the interpacket timing domain while addressing the above mentioned issues. As provenance is hidden in another host-medium, our solution can be conceptualized as watermarking technique. However, unlike traditional watermarking approaches, we embed provenance over the interpacket delays (IPDs) rather than in the sensor data themselves, hence avoiding the problem of data degradation due to watermarking. Provenance is extracted by the data receiver utilizing an optimal threshold-based mechanism which minimizes the probability of provenance decoding errors. The resiliency of the scheme against outside and inside attackers is established through an extensive security analysis. Experiments show that our technique can recover provenance up to a certain level against perturbations to inter-packet timing characteristics.

Index Terms—Data stream, sensor network, secure provenance, watermarking, transmission

1 Introduction

The proliferation of the Internet, embedded systems, and sensor networks has greatly contributed to the wide development of streaming applications. Examples include real-time financial analysis, location-based services, transaction logs, sensor networks, control of automated systems. The data that drives such systems is produced by a variety of sources, ranging from other systems down to individual sensors and processed by multiple intermediate agents. This diversity of data sources accelerates the importance of data provenance to ensure secure and predictable operation of the streaming applications. Data **provenance** is considered as an effective tool for evaluating data trustworthiness, since it summarizes the **history of the ownership and the actions performed** on the data. Recent research works on the provenance-based evaluation of the trustworthiness of sensor data [18], location data [8], and multihop network [29] manifest the key contribution of provenance in data streams. As an example consider a battlefield surveillance system that gathers enemy locations from various sensors deployed in vehicles, air-crafts, satellites, etc., and manages queries over these data. Mission critical applications in such a system must access only high confidence data in order to guarantee accurate decisions. Thus, the assurance of data trustworthiness is crucial here, which prioritizes the secure management of provenance. Likewise, provenance plays a key role in process control tasks (i.e., SCADA systems) that analyze the real-time data collected from different sensors. Provenance facilitates such systems by leveraging high trustworthy data, thus, preventing wrong control decisions. The significance of provenance for streaming data is also emphasized in the *Research and Development Challenges for National Cyber Security* report [3] which recommends research initiatives on efficient and secure implementation of provenance for real-time systems.

Past research on provenance mainly focused on modeling, collecting, and querying, leaving security unexplored. Moreover, although the provenance of workflows and curated databases [11], [20] has been investigated extensively, very few approaches have been reported for data streams. In this paper, we introduce and study the problem of secure and efficient transmission of provenance in an aggregation supportive streaming environment (focusing on sensor networks). The unique nature of streaming environment imposes a set of challenges to the provenance solution.

- It must conform with the *fast arriving data nature* and *high processing throughput* of the infrastructure. Hence, the solution cannot introduce significant processing overhead to individual data element.
- Provenance, even of increasing size, should be efficiently managed and transmitted so to *minimize the additional bandwidth consumption*.
- The provenance management system must transmit provenance in a *secure manner* as well as *quickly react* to malicious attacks.

- *S. Sultana is with the Department of Electrical and Computer Engineering, Purdue University, Apt. 3, 133 Nimitz Drive, West Lafayette, IN 47906. E-mail: ssultana@purdue.edu.*
- *M. Shehab is with the Department of Software and Information Systems, University of North Carolina, Charlotte, NC 28223-0001. E-mail: mshehab@uncc.edu.*
- *E. Bertino is with the Department of Computer Sciences, Purdue University, LWSN Building, 305 N. University Street, West Lafayette, IN 47906. E-mail: bertino@cs.purdue.edu.*

Manuscript received 4 Apr. 2011; revised 28 July 2011; accepted 20 Dec. 2011; published online 13 Feb. 2012.
Recommended for acceptance by K.-L. Tan.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-04-0175.
Digital Object Identifier no. 10.1109/TKDE.2012.31.

While past research has focused on the management of streaming and provenance data through separate channels [25] as well as on how to trace the source of the stream long after the process has been completed [27], our solution is the first that addresses all of the challenges mentioned above. We propose a framework that transmits provenance along with the sensor data by hiding it over the interpacket delays (IPDs) (i.e., the delay between sensor data packets). The provenance of a data packet includes the identities of nodes in the data flow path. The provenance transmission requires a number of data packets (i.e., the associated IPDs). Each node in the path encodes one bit of provenance information over each IPD. Hence, the provenance can be decoded by processing the IPDs required to encode all the provenance bits. The embedding of provenance within a host medium makes our technique reminiscent of watermarking [7]. However, since the *interpacket delays* are used as watermark carrier, there is no data degradation due to watermarking. *Scalability* is another concern as the size of provenance increases proportionally to the number of participating nodes. We address this issue by adopting a *spread spectrum* based technique which supports multiuser communication over the same medium [9]. Hence, our proposed framework provides scalability, imperceptibility and robustness to attacks. The contributions of this paper include:

- introducing the problem of secure provenance transmission for streaming data;
- design of a watermark based approach for embedding provenance in the inter-packet timing domain;
- an efficient technique for provenance retrieval based on an optimal threshold;
- a security analysis of our scheme; and
- an experimental evaluation using synthetic data that reflects the timing characteristics of real-world data.

Why not traditional security mechanisms?: A possible approach to the problem of secure provenance for streaming could be based on traditional security solutions like encryption, digital signature, and message authentication code (MAC). In a digital signature (or MAC)-based mechanism, each party involved in the data processing would append its information to data and sign it (or compute and attach the MAC) to ensure authenticity. In addition, encryption and an incremental chained signature based approach for secure document provenance [12] could be adapted for use in sensor networks. However, such approaches are not applicable in resource constrained sensor networks, because provenance information tends to grow very fast, often becoming several magnitudes in size larger than the original data [12]. Such a characteristic thus would force the transmission of a vast amount of provenance information along with data. Encryption/signature/MAC-based mechanisms cannot help in reducing such size even after compaction. Hence, traditional security means incur significant bandwidth overhead and impact efficiency and scalability. On the contrary, watermarking allows one to embed provenance within the data itself; hence it makes efficient usage of bandwidth. Moreover, watermarking reduces power and processing requirements.

Another reason that could motivate the adoption of existing security mechanisms is that in our context each data source typically generates a lot of packets; thus there are large groups of packets that have the same provenance. In this context, the expensive encryption/MAC/digital signature mechanisms can be used with low frequency to send provenance in some selected packets. However, such an approach has the drawback that the attackers would be able to identify the provenance containing packets by observing and analyzing all the data packets. Upon detection, the attacker could then drop such packets and block the provenance transmission. Even if such attacks could be detected, there would be no way to recover the destroyed provenance. On the other hand, a careful design of the watermarking scheme can make the provenance invisible to the attacker, which enhances the reliability and robustness of the provenance. Moreover, our scheme is able to recover the provenance even in the face of several attacks.

The rest of the paper is organized as follows. Section 2 introduces the system model and preliminaries. The proposed scheme for encoding and decoding provenance is summarized in Section 3. Sections 4, 5, and 6 explain the algorithms associated with the different stages of provenance embedding. Sections 7 and 8 discuss the decoding threshold evaluation, and the provenance retrieval scheme, respectively. Section 9 analyzes the security of our scheme. The experimental results are presented in Section 10. Section 12 discusses related work and Section 13 concludes the paper.

2 System Model and Background

2.1 Network Model

We consider a typical deployment of wireless sensor networks, consisting of a large number of nodes. Sensor nodes are stationary after deployment, though the routing paths may change due to node failure, resource optimization, etc. The network is modeled as a graph G (N,E) where

- $N = \{n_i : n_i$ is a network node with identifier $i\}$: a set of network nodes;
- $E = \{e_{i,j} : e_{i,j}$ is an edge connecting nodes n_i and $n_j\}$: the set of edges between the nodes in N.

There exists a base station (BS) that acts as sink/root and connects the network to outside infrastructures such as the Internet. All nodes form a tree rooted at the BS and report the tree topology to BS once after the deployment or whenever a change in the topology occurs. Since the network does not change frequently, such a communication will not incur significant overhead. The network is organized into a cluster structure [13]. Sensory data from the children nodes are aggregated at the cluster-head a.k.a. aggregator, and routed to the applications through the routing tree rooted at the BS.

The BS cannot be compromised and it has a secure mechanism (e.g., μTESLA [23]) to broadcast authentic messages in the network. A low-level communication protocol will ensure reliable and in order message delivery. Each sensor node shares a secret key K_i with the BS and is assigned a unique pseudonoise (PN) sequence, denoted as $\mathbf{pn}_i = pn_i[1]\, pn_i[2] \cdots pn_i[L_p]$, where L_p, an integer greater than 0, is the length of the PN sequence.

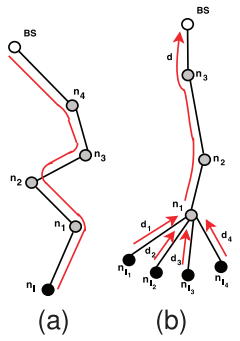

Fig. 1. Provenance graph for a sensor network. (a) Simple provenance. (b) Aggregate provenance.

2.2 Data Model

The sensor network supports multiple distinguishable data flows where source nodes generate data **periodically**. A node may also receive data from other nodes in order to forward them towards the BS. For the rest of the paper, we will use the term *data arrival* with the meaning of data generation or receipt at a node. While transmitting, a node may send the sensed data or pass an aggregated data value computed from multiple sensors' readings, or act as a routing node. Each data packet contains an attribute value and provenance for this attribute. The packet is also timestamped by the source node with the generation time. As we see later, the packet timestamp is crucial for provenance embedding and decoding processes. Hence we use a message authentication code to maintain its integrity and authenticity.

However in case of aggregation, the cluster head creates a new packet with aggregated data which makes it difficult to preserve the packet timestamps received from all of its children. Hence, we assume that at the beginning of each aggregation round, all of the cluster nodes synchronize their time and agree upon a timestamp to associate with their data packets for that round. Then the cluster head creates a new packet with the aggregated data and authenticated timestamp from one of its children. Since time synchronization is performed in sensor networks for various purposes [10], it will not add extra overhead to our protocol.

2.3 Data Provenance

The notion of data provenance is well established in many scientific domains; however, the definition of provenance varies depending on the specific application domain [21]. In the context of sensor networks, we use the definition of data provenance as information about the source node and the nodes that processed/forwarded the data throughout its transmission towards the BS [18]. Provenance is formally defined as:

Definition. *The provenance p_d for a data item d is a rooted tree satisfying the properties: 1) p_d is a subgraph of the sensor network G(N,E); 2) the root node of p_d is the BS; 3) for any $n_i, n_j \in p_d$, n_i is a child of n_j if and only if n_i participated in the processing of d and/or passed data information to n_j.*

Fig. 1 shows two different kinds of provenance. In Fig. 1a, data item d is generated at leaf node n_{l_1} and the internal nodes simply pass it to BS. We call such internal nodes *simple node*s and this kind of provenance *simple provenance*. The simple provenance can be represented as a path. In Fig. 1b, the aggregator n_1 generates data item d by aggregating data $d_1, \ldots, d_4$ from $n_{l_1}, \ldots, n_{l_4}$ and passes d toward BS. Here, the provenance is called *aggregate provenance* and represented as a tree.

We propose a distributed strategy to securely embed provenance as a set of nodes over the interpacket delays. In our scheme, **the unique PN sequence is used as the identity of a node and thus encoded in the provenance**. Upon extracting the list of nodes, the BS can easily determine their order with the knowledge of network topology and construct the provenance tree.

2.4 Adversary Model

We assume that the source and the destination node (i.e., the BS) on the path being monitored are honest. Any other arbitrary node may be malicious. An adversary can eavesdrop and perform traffic analysis anywhere on the path. Beyond that, the adversary is able to deploy a few malicious nodes as well as compromise few legitimate nodes by capturing them and physically overwriting their memory. Thus, the attacker might have control of more than one node, and these malicious nodes might collude to attack the system. If an adversary compromises a node, (she)he can extract all key materials, data, and codes stored on that node. The adversary may drop, or inject packets on the links that are under its control.

From the perspective of network access capabilities, the described adversaries can be typed into two categories: *outside* and *inside* attackers. The outside attacker is an external entity which cannot access any secret of the legitimate nodes but can eavesdrop, inject or replay data. The inside attacker is actually a legitimate node which is compromised and attacks the network by running malicious code or exploiting the stolen secrets and data.

The goal of an attacker is to make undetected changes to the provenance to carry out mischievous activities, such as to make an innocuous node untrustworthy by making it responsible for false data, and so on. We do not consider denial of service attacks such as the complete removal of provenance, since a data packet with no provenance record at all will make the data highly suspicious [12] and hence generate an alarm at the BS. Instead, the primary concern is that an attacker might want to modify the provenance by perturbing interpacket timings. Hence, our objective is to achieve the following security properties:

Confidentiality. An adversary can observe the time between consecutive packet transmissions among neighboring nodes and get the IPDs of a particular data flow at associated nodes. By using the captured IPDs, an attacker must not be able to access or retrieve the provenance information of legitimate nodes. Thus, we aim to provide the following confidentiality assurances:

- **P1**: If an attacker does not know that provenance is being embedded over the IPDs, it cannot detect the presence of provenance by observing the data flow timing characteristics. Even if the attacker is aware of provenance embedding, it cannot retrieve the provenance consisting of legitimate nodes.

- **P2**: Only authorized parties (e.g., the BS) can access and check the integrity of the provenance.

Integrity. An inside attacker may try to modify/destroy the provenance of data packets passed through it. A naive attack is to change the interpacket timings randomly to make the provenance worthless. More intelligent attempts include adding legitimate nodes to the provenance of fake data, adding compromised nodes to or removing legitimate nodes from valid provenance.

- **P3**: An adversary, acting alone or colluding with others, cannot successfully add legitimate nodes to the provenance of fake data.
- **P4**: An attacker (or a set of colluding attackers) cannot undetectably add or remove nodes from the provenance of data generated by benign nodes.

In addition, we want to prevent provenance forgery attack and to ensure the freshness of provenance.

- **P5**: (Unforgeability) An adversary cannot claim that a valid provenance for a data packet belongs to a different data packet.

- **P6**: (Freshness) An adversary cannot replay captured provenance, avoiding detection at the BS.

However, an adversary may increase network jitter in a way that the recorded IPD at the BS is much larger than the desired value. Such an attack is intended to destroy the embedded provenance. As we discuss later, our scheme can recover provenance if the IPD is altered within a certain limit. In any case, the BS can detect such malicious activity and may utilize some auxiliary mechanism to identify the attacker and take necessary actions. Moreover, the attacker can inject or drop data packets which also alters the IPDs and interfere with the embedded provenance. We successfully recover provenance against the *insertion attack* but survive the *deletion attack* to a certain extent.

2.5 Digital Watermarking

The key idea of digital watermarking is to hide a secret information (watermark) related to a digital content within the content itself thereby ensuring the movement of the watermark along with the content. Thus, digital watermarking involves the selection of a watermark carrier domain and the design of two complementary processes.

1. An *embedding process* E that utilizes the watermark carrier A, the watermark message w, and, possibly, a key K to generate the watermarked data AW as $E(A, w, K) = AW$
2. A *detector process* that determines the existence of a watermark within the received signal (with the key, if applicable) and extracts it.

Though our proposed scheme resembles a watermarking technique, the detection process in our scheme is more powerful since it can extract individual node identities from the aggregated data watermarked in time domain.

2.6 Spread Spectrum Watermarking

Spread spectrum is a transmission technique by which a narrowband data signal is *spread* over a much larger bandwidth so that the signal energy present in any single frequency is undetectable [9]. In our context, the *sequence of interpacket delays* is the *communication channel* and the *provenance* is the *signal* transmitted through it. Provenance is spread over many IPDs such that the information present in one IPD (i.e., container of information) is small. Consequently, an attacker needs to add high amplitude noise to all of the containers in order to destroy the provenance. Thus, the use of the spread spectrum technique for watermarking provides strong security against different attacks. We have adopted the *direct sequence spread spectrum* (DSSS) technique which is widely used for enabling multiple users to transmit simultaneously on the same frequency range by utilizing distinct pseudonoise sequences [9]. The intended receiver can extract the desired user's signal by regarding the other signals as noise-like interferences. The components of a DSSS system are as follows:

Input:

- The original data signal $d(t)$, as a series of $+1, -1$.
- A PN sequence $px(t)$, encoded like the data signal. N_c is the number of bits per symbol and is called PN length.

Spreading. The transmitter multiplies the data with the PN code to produce spreaded signal as $s(t) = d(t)px(t)$

Despreading. The received signal $r(t)$ is a combination of the transmitted signal and noise in the communication channel. Thus $r(t) = s(t) + n(t)$, where $n(t)$ is a white Gaussian noise. To retrieve the original signal, the correlation between $r(t)$ and the PN sequence $pr(t)$ at the receiver is computed as $R(\tau) = \frac{1}{N_c}\sum_{t=T}^{T+N_c} r(t)\ pr(t+\tau)$. If $px(t) = pr(t)$ and $\tau = 0$, i.e., $px(t)$ is synchronized with $pr(t)$, then the original signal can be retrieved. Otherwise, the data signal cannot be recovered. So, a receiver without having the PN sequence of the transmitter cannot reproduce the originally transmitted data. This fact is the basis for allowing multiple transmitters to share a channel. In this paper, we refer to $R(0)$ as *cross correlation*.

In case of multiuser communication in DSSS, spreaded signals produced by multiple users are added and transmitted over the channel. To retrieve the signal for jth user, the cross-correlation between $r(t)$ and $px_j(t)$ is computed. Multi-user communication introduces noise to the signal of interest and interfere with the desired signal in proportion to the number of users. The condition for error free communication in DSSS can be derived from Shannon's channel-capacity theorem

$$C = B\log_2\left(1 + \frac{S}{N}\right),$$

where C is the amount of information allowed by the communication channel, B is the channel bandwidth, and S/N is the signal-to-noise ratio. As S/N is usually $\ll 1$ for spread-spectrum applications, the expression becomes

$$\frac{C}{B} \approx \frac{S}{N}.$$

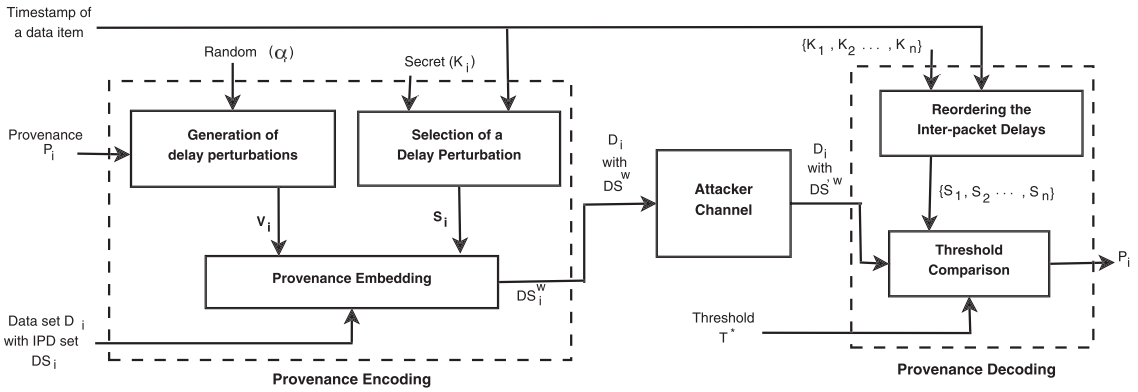

Fig. 2. Stages of provenance encoding at a sensor node and decoding at the base station.

Thus to propagate error-free information for a given noise-to-signal ratio in the channel, the bandwidth should be increased to an appropriate level.

3 Overview of Our Approach

We propose a distributed approach to watermark provenance over the delay between consecutive data packets. Provenance of a data packet includes the nodes in the data flow path. The PN sequence (of L_p bits) of a node is used to uniquely represent its identity in the provenance. Due to the adoption of DSSS based watermarking, all nodes in the provenance use the same medium for transmitting their PN sequences. Hence, only L_p bits of digital information are required for watermarking the provenance. Since we utilize the IPDs, L_p IPDs (in other words, a sequence of $L_p + 1$ packets) are required for embedding and transmitting the provenance of a data packet. We assume that, at least for such number of packets, the provenance (i.e., data flow path) of the packets generated by a source node would be the same. The assumption is reasonable since a routing path does not change too often. Once the path is constructed, it is stable for a good amount of time until there is any link or node failure.

After generating a data packet, the source node marks it with the generation time and ensures the integrity of the timestamp with a MAC. The MAC is computed using the node specific secret key K_i. The next L_p data packets generated by the node, more specifically, the sequence of L_p IPDs is the medium where we hide the provenance of the packets. We denote the set of IPDs by $\mathcal{DS} = \{\ \Delta[1]\ , \Delta[2], \ldots, \Delta[L_p]\ \}$, where $\Delta[j]$ represents the IPD between jth and $(j+1)$th data packet. The data source encodes a bit of its PN sequence over each IPD. Throughout the transmission of a packet towards the BS, each intermediate node also encodes 1-bit of provenance information over the associated IPD. Hence, an IPD recorded at the BS carries the sum of 1-bit information from each node in the path. The process also uses the secret K_i and a locally generated random number α_i (known as impact factor). The BS only knows the distribution of the α_i's.

Fig. 2 presents an overview of our approach for provenance encoding at a sensor node in the data path and decoding at the BS. The process a node n_i follows to encode a bit of PN sequence over an IPD is summarized below:

- **Step E1 (Generation of Delay Perturbations).** n_i generates a set of delay perturbations by using the PN sequence $\mathbf{pn_i}$ and impact factor α_i. The set is represented by $\mathcal{V}_i = \{v_i[1], v_i[2], \ldots, v_i[L_p]\}$, where each element $v_i[j]$ is a real number. Note that, $v_i[j]$ corresponds to the provenance bit $pn_i[j]$. However, the node may perform the computation offline since it is independent of any packet specific information.
- **Step E2 (Selection of a Delay Perturbation).** On the arrival of any (j+1)th data packet, n_i records the IPD $\Delta[j]$ and assigns a delay perturbation $v_i[k_j] \in \mathcal{V}$ to it. To ensure the robustness of the scheme, the delay perturbations are not assigned sequentially to the IPDs, i.e., $v_i[j]$ is not assigned to $\Delta[j]$. Instead, a delay perturbation $v_i[k_j]$ is selected using the secret K_i and the packet timestamp.
- **Step E3 (Provenance Embedding).** In this step, n_i delays the packet transmission by $v_i[k_j]$ time unit. As $v_i[k_j]$ corresponds to the provenance bit $pn_i[k_j]$, through this step a provenance bit is embedded over an IPD. This notion makes our scheme reminiscent of watermarking.

By following the above procedure, each node in the flow path encodes its 1-bit information. Consequently, the provenance bits are watermarked over the L_p IPDs by manipulating them with corresponding delay perturbations, termed as *watermark delay*. This way, $\mathcal{DS}$ is transformed into the watermarked version $\mathcal{DS}^w$. However, data packets may also experience different propagation delays or attacks aimed at destroying the provenance information. At the end, the BS receives the data set along with watermarked IPDs $\mathcal{DS}^w$, which can be interpreted as the sum of delays imposed by the intermediate nodes, the attackers, and the difference between consecutive propagation delays along the data path. Thus, $\mathcal{DS}^w$ represents the DSSS encoded signal in our context. The provenance retrieval process at the BS approximates the provenance from this DSSS signal based on an optimal threshold T^*. The threshold, corresponding to the network diameter and PN length, is calculated once after the deployment of the network. For retrieval purposes, the BS also requires the secret keys $\{K_1, \ldots, K_n\}$ and PN sequences $\{\mathbf{pn_1}, \ldots, \mathbf{pn_n}\}$. The retrieval process follows two steps:

- **Step R1 (Reordering the IPDs).** The IPDs for incoming packets are recorded at the BS. For each

node, the IPDs are reordered according to the algorithm used in **E2**, which produces a node specific permutation of the IPDs. We denote this sequence as $\mathbf{CS_i}$.

- **Step R2 (Threshold-Based Decoding).** For any node n_i, the BS computes the cross-correlation between $\mathbf{CS_i}$ and the PN sequence $\mathbf{pn_i}$. If the correlation result exceeds the threshold T^*, the BS decides that $\mathbf{pn_i}$ was embedded as a part of the provenance.

As the BS does not know which nodes participated in the data flow, it performs the *Bit selection* and *Threshold Comparison* for all nodes. Based on the threshold comparison result, it identifies the nodes in a data flow. In next sections, we discuss these steps in detail.

4 Generation of Delay Perturbations

As the first step to embed provenance, a node n_i generates a delay sequence that is used for watermarking. The PN sequence $\mathbf{pn_i}$ and impact factor α_i are used for this purpose. The PN sequence, consisting of a sequence of $+1$ and -1's, is characterized by a zero mean. The zero mean property is required to ensure successful information decoding at the BS in the context of DSSS-supported *multiuser communication*. α_i is a random (real) number generated according to a normal distribution $N(\mu, \sigma)$. μ and σ are predetermined and known to the BS and all the nodes. Thus, the BS only knows the distribution of α_i's, but not their exact values. However, n_i generates the set of delay perturbations $\mathcal{V}_i$ as a sequence of real numbers as follows:

$$\begin{aligned}\mathcal{V}_i &= \alpha_i \times \mathbf{pn_i} \\ &= \alpha_i \times \{pn_i[1],\ pn_i[2], \ldots, pn_i[L_p]\} \\ &= \{(\alpha_i \times pn_i[1]), \ldots, (\alpha_i \times pn_i[L_p])\} \\ &= \{v_i[1],\ v_i[2], \ldots, v_i[L_p]\}.\end{aligned}$$

5 Selection of a Delay Perturbation

In this section, we present the algorithm that a node applies to select the delay perturbation (from $\mathcal{V}_i$) corresponding to an IPD. If we sequentially assign the delays to the IPDs (which implies that the provenance bits are embedded sequentially), it will be much easier for the attackers to infer information about the provenance or to corrupt the provenance. Hence, we randomize the embedding positions using a different permutation of the elements in $\mathcal{V}_i$. On the arrival of any $(j+1)$th data packet, the jth IPD $\Delta[j]$ is considered for watermarking and the information to watermark is picked out from $\mathcal{V}_i$ using a selection algorithm. Thus, instead of watermarking $v_i[j]$ over the IPD $\Delta[j]$, we select a delay $v_i[k_j]$ for this purpose, where k_j is an index within $[0, L_p - 1]$. The algorithm uses the secret K_i and the packet timestamp, and selects a delay perturbation for the IPD according to the following formula:

$$selection(\Delta[j]) = H(ts[j+1] \parallel K_i)\ \ mod\ \ L_p.$$

Here, H is a lightweight, secure hash function, $\parallel$ is the concatenation operator, and $ts[j+1]$ represents the packet timestamp. Since secure hash functions generate uniformly distributed message digests, each execution of the selection mechanism will result in a unique integer in the range $[0, L_p - 1]$. The resulting integer can be used to index a distinct element in $\mathcal{V}_i$.

As part of the provenance encoding process, each node executes the algorithm once for each of the L_p IPDs returning a set of indices as the permutation of integers from 0 to $L_p - 1$. The indices are used to point the elements in $\mathcal{V}_i$. Thus, the order according to which each node embeds the delays from $\mathcal{V}_i$ over the IPDs forms a permutation of the elements different from the sequential order. This sequence is denoted as $S_i = \{s_i[1],\ s_i[2],\ \ldots\ s_i[L_p]\} = \{v_i[k_1], v_i[k_2],\ \ldots, v_i[k_{L_p}]\}$. Note that, given an IPD, the algorithm will select differently indexed delays for different nodes based on the key K_i. Thus, an attacker cannot predict the *IPD-to-Delay Perturbation* assignment without the knowledge of secrets K_i and L_p. Keeping the provenance length secret is not a requirement but keeping it secret makes it harder for an attacker to regenerate the selections.

6 Provenance Embedding

In this section, we present the provenance embedding algorithm into two steps:

6.1 Simple Provenance Embedding

As shown in Fig. 1a, the simple provenance is represented as a simple path. Each node in the path watermarks its PN sequence over a set of L_p IPDs, i.e., $(L_p + 1)$ packets are utilized. Intuitively, the first packet in a data flow does not experience any delay due to provenance embedding. For any other $(j+1)$th data packet (sent/forwarded), each node in the path hides a provenance bit over the associated IPD $\Delta[j]$. Interchangeably, a node n_i uses the IPD $\Delta[j]$ to accommodate a delay perturbation $v_i[k_j](= s_i[j])$. Using $s_i[j]$, the delay to be added to $\Delta[j]$ is computed as

$$\lambda_i[j] = s_i[j] \times T,$$

where T is the value of a time unit. If $s_i[j] > 0$, the resulting $\lambda_i[j] > 0$ and then we can perform watermarking by simply adding $\lambda_i[j]$ to $\Delta[j]$. But if $s_i[j] < 0$, the delay to be added to an IPD is negative. To avoid this situation, we introduce a constant offset when calculating $\lambda_i[j]$, which ensures that $\lambda_i[j]$ is always positive. The offset may be any constant leading to $\lambda_i[j] > 0$. We use $(\mu\ + const * \ \sigma)$ in our scheme, where *const* is any constant that makes $\lambda_i[j]$ greater than 0, i.e.,

$$const\ >\ \frac{-(s_i[j] + \mu)}{\sigma}.$$

Thus, the final equation is

$$\lambda_i[j] = (s_i[j]\ +\ (\mu\ + const * \ \sigma)) \times T, \tag{1}$$

n_i then performs watermarking by adding $\lambda_i[j]$ to $\Delta[j]$, i.e., delaying the packet transmission by $\lambda_i[j]$ time. Thus, n_i formulates the watermarked IPD $\Delta^w[j]$ and transmission time of the (j+1)th packet $t'_i[j+1]$ as follows:

$$\begin{aligned}\Delta^w[j] &= \Delta[j] + \lambda_i[j], \\ t'_i[j+1] &= t_i[j+1] + \lambda_i[j] + c.\end{aligned}$$

where c is a constant > 0 corresponding to the delay added by a sensor node, including processing and any other delay.

After watermarking, n_i sends the $(j+1)$th packet towards the BS at instant $t_i'[j+1]$. Throughout the transmission, all other nodes in the provenance embed one bit of provenance information over the IPD following the same procedure.

6.2 Aggregate Provenance Embedding

Fig. 1b shows the aggregate provenance, represented as a tree. Assume that in an aggregate provenance tree, n_a is the aggregator possessing U children $n_{l_1}, n_{l_2}, \ldots, n_{l_U}$. At any $(j+1)$th sensing interval $(1 \leq j \leq L_p)$, the child nodes send data to n_a embedding their provenance information over the locally managed IPDs. Watermark delays of the children are denoted by $\lambda_{l_1}[j], \ldots, \lambda_{l_U}[j]$, respectively. n_a computes the aggregated data, attaches authenticated timestamp from one of its children, and also maintains the corresponding IPD in such a way that this delay represents the provenance embedding for the aggregator and its children. Intuitively, we could accomplish this by adding a delay of

$$\lambda_A[j] = \sum_{i=1}^{U} \lambda_{l_i}[j] + \lambda_a[j]$$

to the unwatermarked IPD, where $\lambda_a[j]$ represents the watermark delay computed by the aggregator. The $\lambda_{l_i}[j]$'s can be approximated by the aggregator from the IPD observations while data is being received from the corresponding child. However, this scheme would impose a major delay to the aggregated data which would abruptly reduce data throughput. To address this problem, we propose a different solution based on some mathematical tricks.

Let the watermark delays for a child node n_{l_i} average to Λ_{l_i}. Utilizing the Λ_{l_i}'s of child nodes, n_a computes the watermark delay (denoted as $\lambda_A[j]$) for aggregated provenance as follows:

$$\lambda_A[j] = \lambda_a[j] + \sum_i (\lambda_{l_i}[j] - \Lambda_{l_i}). \tag{2}$$

$\lambda_A[j]$ in (2) may also be negative. So, we also add the constant offset to make $\lambda_A[j]$ always positive. The reason why this solution works is explained in Section 8.

7 Decoding Threshold Evaluation

This section presents the evaluation of an optimal threshold T^* that minimizes the *probability of decoding error* which is defined as the probability of retrieving provenance incorrectly. Let P_{err}, P_1, and P_0 represent the probability of decoding error, probability that a node embeds its identity (i.e., PN sequence) in the provenance and probability of not embedding, respectively. Variables p_e and p_r denote the probability of embedding and retrieval of a node's PN sequence, respectively ($p_e = 1$ implies that the PN sequence of a node was embedded, $p_r = 1$ implies the PN sequence was retrieved). $f(r)$ is the probability density function of cross correlation. P_{err} is calculated as

$$\begin{aligned} P_{err} &= P(p_r = 0, p_e = 1) + P(p_r = 1, p_e = 0) \\ &= P(p_r = 0|p_e = 1)P_1 + P(p_r = 1|p_e = 0)P_0 \\ &= P(r < T|p_e = 1)P_1 + P(r > T|p_e = 0)P_0 \\ &= P_1 \int_{-\infty}^{T} f(r|p_e = 1)dr + P_0 \int_{T}^{\infty} f(r|p_e = 0)dr. \end{aligned}$$

To minimize the probability of decoding errors (P_{err}), we take the first order derivative of P_{err} with respect to T to locate the optimal threshold T^* as follows:

$$\begin{aligned} \frac{\partial P_{err}}{\partial T} &= P_1 \frac{\partial}{\partial T} \int_{-\infty}^{T} f(r|p_e = 1)dr + P_0 \frac{\partial}{\partial T} \int_{T}^{\infty} f(r|p_e = 0)dr \\ &= P_1 f(T|p_e = 1) - P_0 f(T|p_e = 0). \end{aligned} \tag{3}$$

The distributions $f(r|p_e = 0)$ and $f(r|p_e = 1)$ are estimated from the statistics of sets R_e' and R_e, respectively. The experimental observations of cross correlation for the nodes present in the provenance are stored in a set R_e and for those that are not present are stored in R_e'. The values of R_e' and R_e show that the distributions $f(r|p_e = 0)$ and $f(r|p_e = 1)$ can be estimated as Gaussian distributions $N(\mu_0, \sigma_0)$ and $N(\mu_1, \sigma_1)$ respectively. However, the following analysis can still be performed with other types of distributions. P_0 could be estimated by

$$\frac{|R_e|}{|R_e| + |R_{e'}|}$$

and $P_1 = 1 - P_0$. Substituting the Gaussian expressions for $f(r|p_e = 0)$ and $f(r|p_e = 1)$ in (3) and equating it to zero we get the following quadratic equation:

$$\begin{aligned} \frac{\sigma_0^2 - \sigma_1^2}{2\sigma_0^2\sigma_1^2} T^{*2} + \frac{\mu_0\sigma_1^2 - \mu_1\sigma_0^2}{\sigma_0^2\sigma_1^2} T^* + ln\left(\frac{P_0\sigma_1}{P_1\sigma_0}\right) \\ + \frac{\mu_1^2\sigma_0^2 - \mu_0^2\sigma_1^2}{2\sigma_0^2\sigma_1^2} = 0. \end{aligned}$$

The roots of this equation give the optimal threshold T^* that minimizes P_{err}. The second order derivative of P_{err} is evaluated at T^* to ensure that the second order necessary condition $(\frac{\partial^2 P_{err}(T^*)}{\partial T^2} > 0)$ is met. To show the high dependence of the probability of decoding errors on the choice of decoding threshold T^*, we conducted experiments with a sensor network of diameter 12 and PN Length = 240 bits. The histograms and the Gaussian estimates of R_e and R_e' obtained from the experiment are reported in Fig. 3a. The optimal computed threshold T^* is indicated by the dotted vertical line. As we can see from Fig. 3a, the two distributions are far apart which is a direct result of using the competing objects for b_i equal to 1 and 0. Fig. 3b shows the probability of decoding error for different values of the threshold, which in turn shows the presence of an optimal threshold that minimizes the probability of decoding error.

8 Provenance Retrieval

The provenance retrieval algorithm recovers provenance using the secret parameters including the keys $\{K_1, K_2, \ldots, K_n\}$, the PN length L_p, and the optimal threshold T^*. The BS records the watermarked IPDs and executes the retrieval

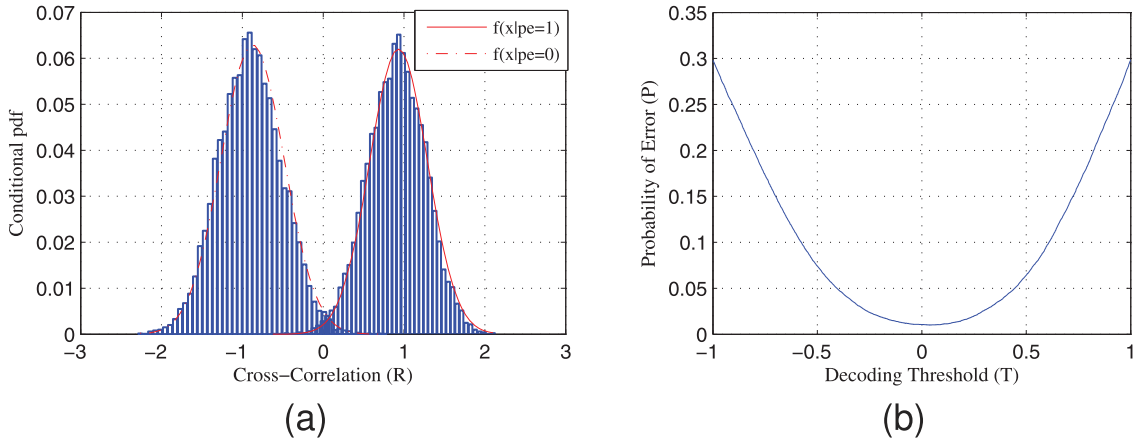

Fig. 3. (a) Shows $f(x|p_e = 0)$, $f(x|p_e = 1)$ and the computed $T^* = 0.0303$. (b) Experimental values of P_{err} for different decoding threshold (T) values.

process whenever it collects a number of L_p IPDs denoted by the set $\mathcal{DS}^w$. Since the BS does not know which nodes embedded their identities in the provenance, it executes the process for all of the nodes in the network and tries to identify the desired nodes. For each node, the BS generates a node specific sequence of real numbers by reordering the IPDs in $\mathcal{DS}^w$ according to the bit selection algorithm. We denote such a sequence by $\mathbf{CS_i} = \{cs_i[1], cs_i[2], \ldots, cs[L_p]\}$. Any element (i.e., IPD) in this sequence can be interpreted as the sum of delays added by the nodes in provenance, the difference of propagation delay between two consecutive data packets, and possibly any delay added due to malicious attacks. Thus,

$$cs_i[j] = \sum_{k,m} \lambda_k[m] \; + \; \sum_{k,m} \Delta tr_{(k\,,\,k+1)}[m] + D[m],$$

where $\Delta tr_{k\,,\,k+1}$ is the difference between the propagation delays of two consecutive data packets from kth intermediate node to $(k+1)$th node and $D[m]$ is any delay added due to attacks. We can expand the equation as

$$cs_i[j] = \sum_{k,m} s_k[m] \times T \; + \; \sum_{k} (\mu \; + const * \; \sigma) \times T + \sum_{k} \Delta tr_{(k\,,\,k+1)}[m] + D[m].$$

As $(\mu \; + const * \; \sigma) \times T$ is a constant, the sum over this constant can be denoted as another constant Tc. To determine whether a node contributes to a data flow, the cross correlation between $\mathbf{CS_i}$ and provenance information $\mathbf{pn_i}$ is computed as follows:

$$\begin{aligned} R_i &= \mathbf{CS_i} \; . \; \mathbf{pn_i} = \sum_j cs_i[j] \times pn_i[j] \\ &= \sum_j \sum_{k,m} (s_k[m] \times T) \times pn_i[j] + \sum_j Tc \times pn_i[j] \\ &\quad + \sum_j \sum_{k,m} \Delta tr_{(k\,,\,k+1)}[m] \times pn_i[j] + \sum_j D[m] \times pn_i[j]. \end{aligned}$$

As $\mathbf{pn_i}$ has an equal number of 1's and -1's, $\sum_j pn_i[j]$ becomes 0 resulting in $\sum_j Tc \times pn_i[j] = 0$. Due to this special property of $\mathbf{pn_i}$, any constant delay added during watermarking will contribute a 0 to the cross correlation. For the same reason, adding a delay of $(\lambda_{l_i}[j] - \Lambda_{l_i})$ during aggregation instead of $\lambda_{l_i}[j]$ has the same effect. The constant Λ_{l_i}, if added, would have been eliminated from the cross correlation.

Note that, the last two terms, representing difference in propagation delays and attacker induced delays, are negligible compared to the first term, i.e., ideal cross correlation value. So, the inclusion of the node in provenance can be decided correctly by a comparison of R_i with T^*. If $R_i \geq T^*$, the identity of this node was embedded, i.e., the node contributed to data flow. Otherwise, the node did not participate. After successfully retrieving the provenance information, the BS resets $\mathcal{DS}^w$ and starts collecting IPDs for future provenance retrievals.

The decoding error can be reduced further by embedding the provenance, i.e., each $v[j] \in \mathcal{V}$, multiple times. The number of repetitions is called *redundancy factor*. At the BS, the provenance is extracted multiple times and the decision about the presence of a node in the provenance is taken based on a *majority voting technique*. Thus, the effect of any unusual propagation delay or malicious attacks is mitigated. Besides, the knowledge of diameter H of the sensor network can be used to determine the nodes in the data flow path more accurately by selecting H nodes with the highest cross-correlation values.

Complexity. Upon the receipt of every L_p data packets from a source/cluster, the BS retrieves provenance by computing the cross-correlation between $\mathbf{CS_i}$ and $\mathbf{pn_i}$ of each node in the network. Thus, the provenance retrieval complexity is $O(N * L_p)$, where N is the number of nodes in the network. The complexity can be improved with topology knowledge by which the BS can estimate a set of probable nodes in the data path and compute cross correlations only for this set of nodes rather than all the nodes in the network.

9 Security Analysis

In this section, we discuss possible attacks that can be performed to corrupt the embedded provenance and show how our scheme defeats them. We discuss from the perspectives of both the outside and inside attackers.

9.1 Outside Attacker

With the capability of capturing data packets and interpacket timing characteristics, an outside attacker may try to disrupt provenance security in different ways.

9.1.1 Provenance Detection and Retrieval

An attacker might want to identify and extract the provenance embedded by a node. Several attacks have been devised to detect and corrupt the active timing-based watermark in network flows. Cabuk implements a covert network timing channel which transmits one packet in a time interval to encode the bit "1" and stays silent for a "-1" bit [5]. Such a static encoding of messages leads to a highly regular behavior in the interpacket delays, whereas overt traffic arrives anytime, resulting in an irregular pattern. Cabuk shows how to detect the covert channel by identifying a regular pattern in the IPDs. Peng et al. develop an attack technique [22] to detect, recover, duplicate or remove a message, watermarked in a flow according to the scheme proposed in [30]. The attacker tries to infer important watermarking parameters (such as quantization step used to compute watermark delay, proportion of watermarked IPDs, etc.) using packet timestamps at each intermediate host and achieves the attack goals utilizing these parameters. Luo et al. propose an approach to detect and autonomously remove spread spectrum flow watermarks (SSFW) [19]. Since the encoder needs to throttle the flow's throughput to a low value for a given period T_c for embedding a "-1" and spreading the watermark using PN codes increases the number of such low-throughput periods significantly, the SSFW causes an abnormal sequence of low-throughput periods (large delays) in the flow. Hence, the attacker can detect the SSFW by identifying the presence of anomalous sequences of low-throughput periods. Kiyavash et al. have devised a multiflow attack to detect the SSFW [17] based on the observation of long low-throughput period on several flows compared to a trained model.

It is important to notice that these attacks mainly focus on detecting whether a data flow has a secretely embedded watermark and, if present, then on recovering/removing it. On the contrary, the attacker in our context might have prior knowledge about the fact that a timing-based provenance watermarking scheme is applied in the sensor network. Also we are not considering the complete removal of provenance as well. Therefore, attacks conducted to only detect the existence of provenance will not help the attacker anyway, unless the attacker can retrieve the provenance information of a node. In addition, most of these attacks are addressed to specific watermarking techniques and hence cannot be generalized to disrupt any watermarking scheme. However, the following claim shows that our scheme can evade such detection and retrieval attacks

Claim 1. *By observing the data flow timing characteristics: 1) An attacker, being unaware of provenance transmission over the interpacket delays, cannot detect the presence of provenance by: i) Regularity testing traffic, ii) Inferring the watermark parameters, and iii) Carrying a multi-flow attack. 2) An attacker with prior knowledge about provenance embedding technique cannot retrieve the provenance information of legitimate nodes. (P1)*

Justification. We justify the claim by showing the robustness of our watermarking scheme against the above mentioned attacks. In our scheme, the watermarked IPDs do not follow any regular pattern (unlike [5], where a time interval is used to encode a "$+1$" by transmitting a packet in the interval and a "-1" by remaining silent for the period). Though the latency is increased in our case, the IPDs appear random in nature and it is hard to distinguish the patterns generated by the watermarking from natural variation in traffic rates. Hence, our scheme can evade detection based on regularities in data traffic [5]. Also our watermarking technique differs from the scheme proposed in [14] as we formulate the watermark delay in a completely different way and use every IPD for watermarking purpose. As a result, the watermark recovery process depending on the inference of secret watermarking parameters [22] does not help an attacker in extracting the provenance, embedded according to our scheme. Moreover, the lack of clock synchronization between nodes will weaken this attack. Compared to existing SSFW techniques, our scheme uses low amplitude watermarks, i.e., much smaller delays (on the order of few milliseconds) that appear close to natural network jitter. It makes the provenance invisible to attackers and thus prevents the attackers from detecting and removing the provenance [15]. Regarding the multiflow attack, Kiyavash et al. [14] showed that this attack can be thwarted if one applies different PN codes or watermarks to different flows and change the position of watermark in a flow [17], [14]. In our system, each node possesses unique provenance information that is watermarked in the flow and also the embedding position of the provenance bits is changed continuously. Thus, a multiflow attack cannot defeat our scheme.

However, a statistical test, based on the assumption that IPDs of covert traffic center on limited numbers of distinct values instead of being randomly distributed [4], can detect the presence of provenance in the time domain. The reason is that the mean of watermark delays for "$+1$" and "-1" bits converges to two separate values in our scheme. Still, an attacker cannot retrieve the provenance information of a node by observing the IPDs of flows from/to that node. The embedding positions of provenance bits are changed in every round of embedding based on the packet timestamp and they also differ from node to node. Hence, given a sequence of L_p IPDs, the attacker has to try all combinations of these numbers to get the order of bits in the provenance information. For example, given 120 delays for a 120 bit provenance information (with equal number of 1's and -1's), the attacker has to try $\binom{120}{60}$ combinations to get the original sequence of provenance bits.

Thus our watermarking scheme makes the embedded provenance invisible to most of the attacks.

9.1.2 Replay Attack

An adversary may fraudulently transmit previously heard data packets (transmitted by legitimate nodes) to give a false idea about the sensed environment [24]. For an IPD-based provenance transmission system (like ours), the attacker also observes the timing characteristics in order to maintain them during packet replay. To make the replayed data appear as fresh, the attacker will update the packet timestamp to a recent value. Nevertheless, we claim that

Claim 2. *Replay attack cannot be performed successfully. (P6)*

Justification. An old data with fabricated timestamp and replayed provenance will be discarded at the BS. The selection of provenance bit for any jth IPD depends on the timestamp of $(j+1)$th packet and thus changes with the varying timestamp. So, sustaining the old time observations while marking the packet with a new timestamp does not allow the BS to extract provenance successfully. Consequently, the provenance integrity check fails.

9.2 Inside Attacker

As discussed earlier, the inside attacker is a more powerful attacker which will try to disrupt the provenance security more intelligently. Obviously, such an attacker can maliciously modify or disable the code of the provenance module on the compromised node. However, we leave out this attack as it will alter the provenance for the node in a way that it will be detected at the BS. Our scheme defends against the following integrity and forgery attacks:

Claim 3. *An attacker, acting alone or colluding with others, cannot add legitimate nodes to the provenance of data generated by the compromised nodes. (P3)*

Justification. The attacker wants to generate fake data and construct the provenance including some innocent nodes $\{n_{i_1}, n_{i_2}, \ldots, n_{i_U}\}$ to mark them as untrustworthy by making them responsible for false data. However, this attack will fail since the provenance embedding process requires node-specific secrets, like the PN code, the secret key, and the impact factor, and the attacker does not know these for the uncompromised nodes.

Claim 4. *An attacker, acting alone or colluding with others, cannot successfully add or remove nodes from the provenance of data generated by benign nodes. (P4)*

Justification. Assume that n_e and n_m are compromised nodes and collude to execute the attack. A benign data item d, with provenance $p_d = \{n_{i_1}, n_{i_2}, \ldots, n_{i_U}\}$, is routed through n_e which wants to remove n_{i_2} from p_d and replace it with n_m. To remove n_{i_2} from provenance, n_e has to remove the delays added by n_{i_2} from IPDs. Since negative delays cannot be added, n_e will adjust the jth IPD by delaying the jth packet which decreases the delay introduced to the $(j+1)$th packet for provenance embedding. The amount of delay to be added can be found by observing the timing characteristics of packets to and from n_{i_2}. Note that n_e has to adjust the IPDs in reverse order, from $j = L_p$ to 1. To achieve this, n_e has to accumulate all the (L_p+1) packets, adjust their IPDs, and then transmit these packets towards the BS maintaining the adjusted timings. Such an attack scheme will add too much delay to the packets, which will definitely be detected at the BS. Regarding the addition of a node, n_e can easily add n_m in the provenance if they collude. However, the provenance integrity check at the BS will fail and detect an attack.

Claim 5. *Provenance forgery can be detected, i.e., given the valid provenance for a data packet, the attacker cannot associate this provenance with a data packet with a difference provenance. (P5)*

Justification. A malicious routing node n_e can perform two types of attack.

Forgery attack 1. Suppose that the data packet d belongs to a data flow generated by a benign node n_s. The provenance of d is $p_d = \{n_s, n_{i_1}, \ldots, n_{i_U}\}$. n_e might want to associate p_d with a fake packet d_e. To achieve this, n_e tries to insert d_e in the flow while maintaining the observed timing characteristics. However, to certify that d_e is a part of the flow generated by n_s, n_e must generate the MAC of the data value and timestamp by using the secret key of n_s. Being unaware of the secrets of n_s, n_e cannot generate the MAC.

Forgery attack 2. d_1 and d_2 belong to two different data flows generated by n_{s_1} and n_{s_2}, respectively. n_e swaps the data value of these packets. Hence, the BS will now identify n_{s_2} as a part of the provenance for d_1 whereas d_1 contains the MAC generated by n_{s_1}.

Hence, the data integrity check will detect the provenance forgery in both cases.

Claim 6. *Only authorized parties can access and check the integrity of provenance. (P2)*

Justification. This follows from the provenance decoding process which requires the PN sequences and secret keys of all nodes in the network (at least for nodes in the provenance). Only the authorized party (the BS in our case) that has access to these information can retrieve the provenance and thereafter check the integrity.

From the above security analysis it follows that an adversary cannot access or modify the provenance without being detected. Nevertheless, modifications may destroy the provenance and impact the robustness of the scheme. The strength of our scheme is that, with some redundancy and detection mechanism, it can recover the provenance up to a great extent. Here, we consider the following attacks by compromised nodes:

Deletion attack. A compromised node can destroy the information carried out by the IPDs by dropping data packets routed through it. Dropping any jth data packet consumes the (j-1)th and jth IPD. However, we can mitigate this attack by embedding the provenance multiple times and employing the *majority voting technique* when retrieving the provenance, as discussed in Section 8. The impact and effectiveness of the *redundancy factor* (i.e., how many times the provenance is embedded) on provenance recovery is evaluated and reported in Fig. 4c.

Alteration attack. This attack perturbs the IPDs with the goal of moving the cross-correlation values from above the threshold T^* to below the threshold T^* and vice versa, leading the erroneous retrieval of provenance. As in the deletion attack, embedding provenance multiple times will reduce the impact of this attack. However, the attacker may try to change the IPDs within a safe range, since an alteration to an IPD beyond a certain limit would be recognized by the BS as an attack. Such a modification, however, would affect the cross-correlation value negligibly, thus leaving the provenance decoding process undisturbed.

Insertion attack. A malicious routing node may insert fake data in the data flow generated by a legitimate node. Through the MAC verification as discussed in **Claim 5** or using some standard detection mechanism, the BS can detect such false data packets and discard them. Thus, an insertion attack will have almost no effect on the provenance decoding.

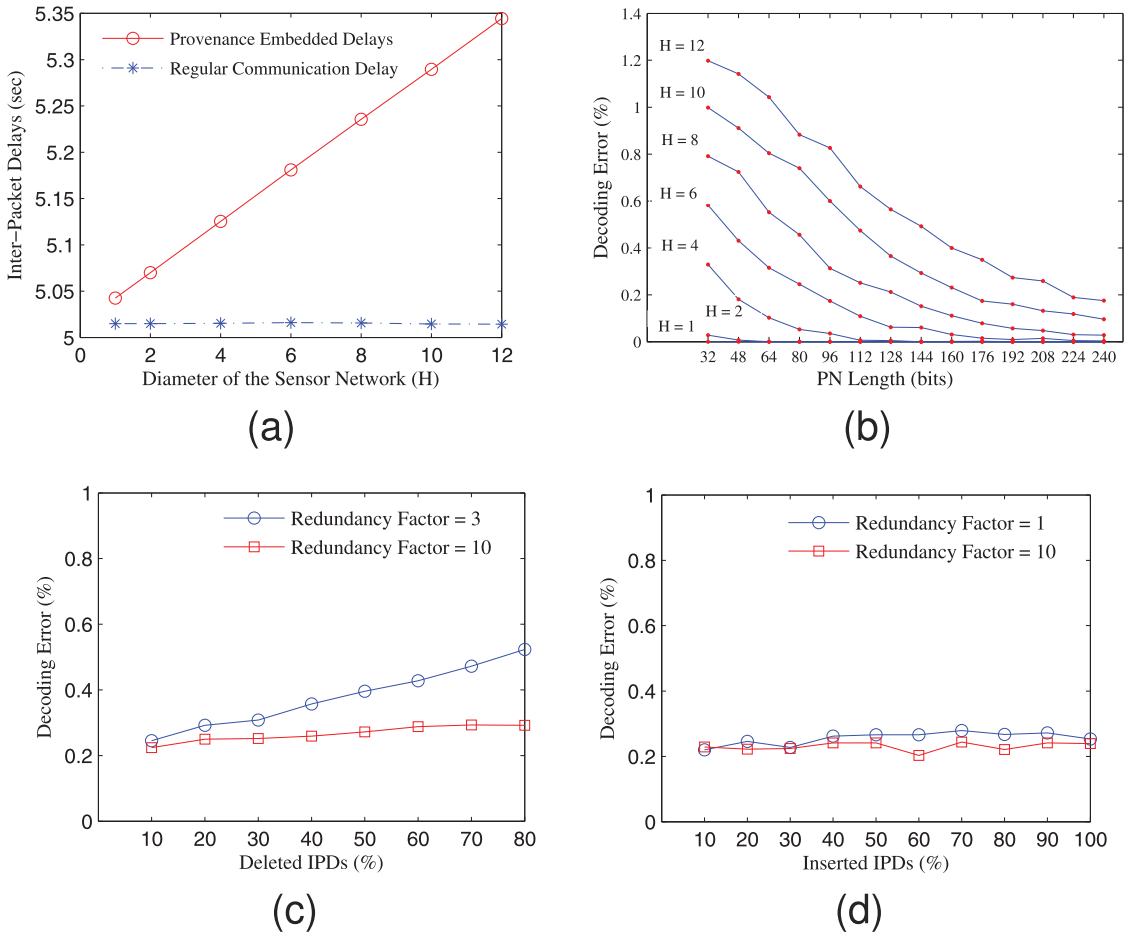

Fig. 4. (a) Delays incurred due to provenance embedding with varying diameters (b) Decoding error corresponding to various diameters and PN lengths. Resilience to (c) Deletion, and (d) Insertion attacks. Higher redundancy factor, i.e., embedding provenance multiple times mitigates the attacks.

10 Experimental Evaluation

We now present simulation results assessing the scalability and robustness of our scheme. For the experiments, we simulate the sensor network as a tree with diameter H. The network consists of 1,000 nodes with default values of $H = 8$, and $L_p = 160$ bits. Other parameters include μ =5, $\sigma = 0.005$, $const = 100$, time unit $= 5$ ms, and redundancy factor $= 1$. We intended to perform experiments on real-life test data [1], [2]. Though the data sets obtained from these projects are timestamped and labeled with the data source, we cannot get the intermediate routing nodes. Note that, sensor data values are of no interest to us, rather we were interested in the reported IPDs. Sensor data is generated every 5 seconds. For each experiment, the simulations were run 100 times.

10.1 Scalability

The scalability of our solution is evaluated by quantizing the impact of H on the overall delay due to provenance embedding. The reason why we investigate the relationship of delay to the network diameter instead of the number of nodes is that the provenance length increases linearly with the diameter. In comparison, the effect of the total number of nodes is much lower. Fig. 4a shows a comparison of natural IPDs with the watermarked IPDs. The watermarked IPD increases from natural IPD by a maximum of 6 percent.The graph also shows that the watermarked IPD linearly increases with H though the increasing rate is not high.

As the diameter of sensor network has a direct influence on the PN length, one has to determine the optimal PN length for a particular H that ensures a low decoding error. Fig. 4b reports the percentage of the provenance decoding error for different PN lengths with varying network diameters. Predictably, an increase in the PN length results in a decrease of the decoding error for a particular diameter as well as the increase in diameter imposes a higher error rate for a particular PN length.

10.2 Provenance Recovery

These experimental results show how well the decoding process can recover the provenance against various attacks discussed in Section 9.

Deletion Attack: the adversary randomly drops α data packets (of a data flow) routed through it. The provenance is then decoded and the decoding error is measured for different α values as reported in Fig. 4c. We evaluated the performance of our scheme for various redundancy factors. The decoding error decreases with increasing values of the redundancy factor.

Alteration Attack: we evaluated the performance of our decoding technique against two types of alteration attacks

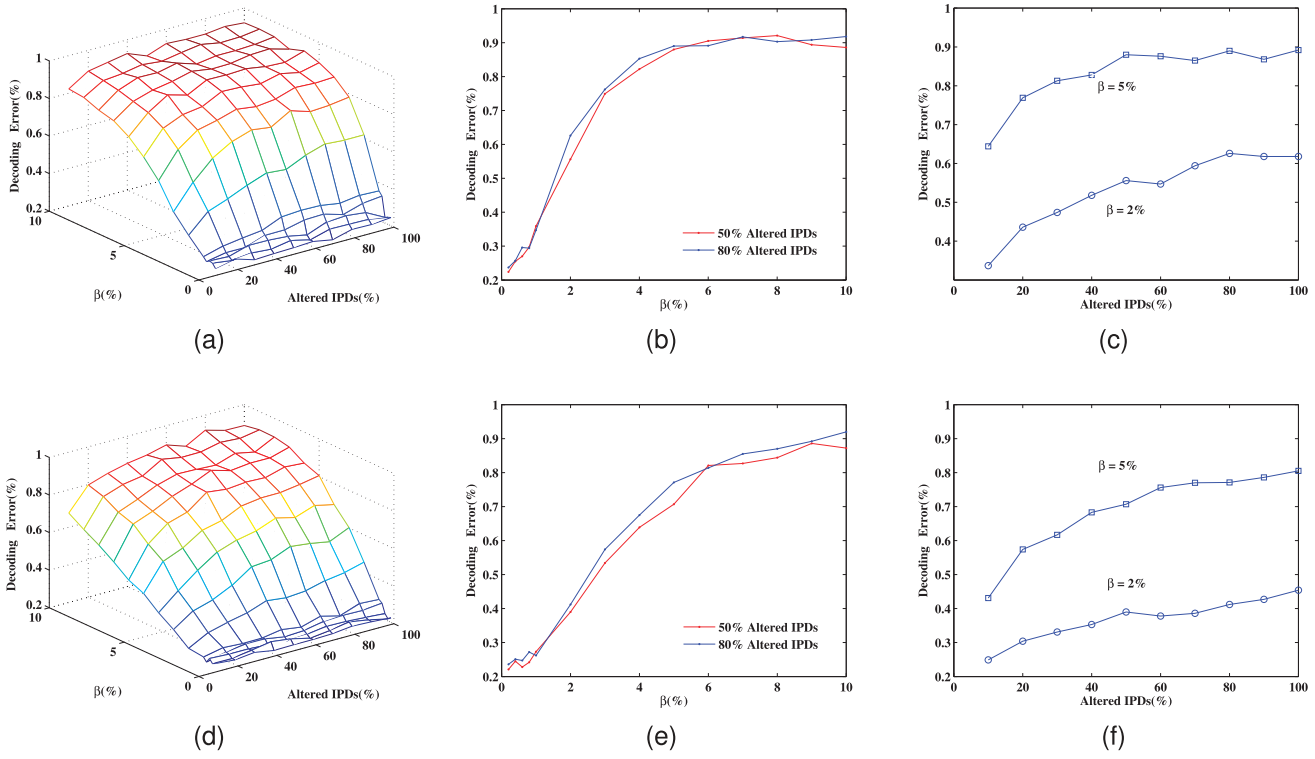

Fig. 5. (a), (b), and (c) Resilience to fixed-(α, β) alter attacks, and (d), (e), and (f) Resilience to random-(α, β) alter attacks. (a) Naturally, increasing rate of altered IPDs (α) and amount of alterations (β) result in higher decoding errors. (b) Provenance decoding error corresponding to increasing β. Results for different altered data percentages shown. (c) Decoding error against increasing rate of IPD alterations. Smaller β value shows more resiliency. Similar experiments were performed against random-(α, β) attacks and similar trend in results are shown in (d), (e), and (f), respectively.

namely, fixed and random (α, β) alteration attacks. In the fixed-(α, β) alteration attack, the attacker randomly selects $\frac{\alpha}{2}$ data packets and delays them by multiplying the corresponding IPDs by $(1 + \beta)$. Consequently, each following IPD (total $\frac{\alpha}{2}$) is decreased by $(1 - \beta)$. Here, β is a fixed value. In the random-(α, β) attack, the IPDs are multiplied by $(1 + x)$, where x is a uniform random variable $\in [0, \beta]$. Figs. 5a, 5b, and 5c show the behavior of our scheme against the fixed-(α, β) alteration attack. In Fig. 5a, as the percentage of IPDs altered and the alteration factor increases, so does the decoding error. However, our solution seems surprisingly resilient. The provenance decoding error shows low increases for increasing percentages of altered IPDs (Fig. 5b) or the alteration factor (Fig. 5c). Similar results were experienced for the random-(α, β) attack as shown in Figs. 5d, 5e, and 5f.

Insertion Attack: in this experiment, we insert α data packets in the flow, i.e., add α IPDs. Utilizing our detection mechanism, we can achieve an almost constant decoding error while varying the percentage of inserted IPDs. Fig. 4d shows the robustness of our solution against the insertion attack.

11 Discussions

11.1 Cost Analysis

For cost analysis, we compare our scheme with a MAC-based provenance scheme (MP) as shown in Table 1. In MP, a node transmits the nodeID and a MAC computed on it as the provenance record. The nodeID is assumed to be of 2 bytes. A sensor network specific CBC-MAC of 4 bytes is computed using TinySec library [16]. Hence, a provenance record is of 6 bytes in MP. As each node in the flow path embeds its provenance information, the provenance size linearly increases with the path length (i.e., number of hops) which also linearly increases the transmitted data packet size. Thus for a H-hop path, the provenance is of $6 \times H$ bytes in MP. On the contrary, our protocol only adds a 4-byte CBC-MAC computed on timestamp to the data. We also measure the per-node delay introduced by our watermarking mechanism and compare it with per-node data transmission time in MP. For this, we consider a CC2420 stack implementation in TinyOS with 250 Kbit/s transmission rate and data payload of 23 bytes. Since the data length in MP increases with hop count, the transmission time also increases. Though our time-based watermarking scheme introduces delay in the data flow, delays are of the same magnitude as natural network jitter. However, in resource constrained sensor networks, energy is mainly consumed for data transmissions. In this regard, we perform better than MP since data length does not grow linearly in our protocol. Regarding computational

TABLE 1
Cost Comparison between Our Protocol and a MAC-Based Provenance (MP) Solution

	Packet Length (bytes)	Delay per Node (ms)
Our Protocol	D + 4	~ 10-50
MP	D + 6×H	~ 5-10

D indicates data length and H represents path length. Delay in our protocol depends on parameter selections.

cost, our protocol requires a node to compute a hash function whereas MP requires a MAC computation for provenance embedding.

11.2 Applicability in the Broader Range of Streaming Applications

As discussed in Section 12, *interpacket timing* based network flow watermarking has been widely used to identify the correlated traffic flows and to detect the source of attack behind the stepping stone(s). Hence, we can apply our timing based provenance watermarking scheme to a large class of streaming applications. To apply our scheme in streaming environments, a mechanism is required for distributing secret keys and PN sequences to the participating nodes. To retrieve provenance, the receiver needs to know the PN sequences of the possible nodes in the communication path.

12 Related Work

Work related to our approach falls into two classes: provenance security, and time-based flow watermarking.

Hasan et al. [12] propose a chain model of provenance and ensure integrity and confidentiality through encryption, checksum and incremental chained signature. Syalim et al. [26] extend this method by applying digital signature to a DAG model of provenance. However, these generic solutions are not aware of the sensor network characteristics. Since provenance tends to grow very fast, transmission of a large amount of provenance information along with data incurs significant bandwidth overhead and looses efficiency and scalability. The most relevant work by Chong et al. [6] proposes a scheme for embedding the provenance of data source within the data set. While it reflects the importance of issues we address, it is not intended as a security mechanism, hence, does not deal with malicious attacks. Besides, practical issues like scalability, data degradation have not been well addressed here.

There exists a lot of work regarding active-timing based watermarking for network flow [5], [14], [30], [28]. Our watermarking scheme significantly differs from these approaches in various aspects. 1) All of these schemes embed a single watermark message over the IPDs of a flow. On the contrary, we allow multiple nodes to watermark provenance over the same set of IPDs. 2) Our decoding process is completely different since it does not retrieve the embedded provenance by inferring bits from each IPD. Instead, we use a unique approach based on a cross-correlation and threshold based mechanism 3) Several mechanisms (e.g., [5]) watermark a bit by controlling the data throughput for a certain amount of time whereas we prolong the IPD by a small amount of time. Though Wang and Reeves [30], Kiyavash et al. [17] insert a watermark by delaying the transmission of some packets, the first scheme is subject to detection and recovery attack [22]. As described earlier, our scheme is resilient to this attack. While Kiyavash et al. use spread-spectrum technique to make watermark delays much smaller, their decoding process is nonblind and requires the unwatermarked IPDs to be stored in a database.

13 Conclusion

In this paper, we address the novel problem of securely transmitting provenance for data streams. We propose a spread-spectrum watermarking-based solution that embeds provenance over the interpacket delays. The security features of the scheme make it able to survive against various sensor network or flow watermarking attacks. The experimental results show that our scheme is scalable and extremely resilient in provenance retrieval against various attacks. In future, we will investigate the feasibility of this technique for large sized provenance.

Acknowledgments

This work has been partially supported by the Northrop Grumman Cybersecurity Research Consortium and by US National Science Foundation (NSF) under grants CNS-1111512 and CNS-0964294. Dr. Shehab's work was funded by NSF under grants CNS-0831360, CNS-1117411 and Google Research Award.

References

[1] Namos, http://robotics.usc.edu/namos, 2011.
[2] The Sensorscope Project, http://sensorscope.epfl.ch, 2012.
[3] National Cyber Security Research and Development Challenges, Related to Economics, Physical Infrastructure and Human Behavior, 2009.
[4] V. Berk, A. Giani, and G. Cybenko, "Detection of Covert Channel Encoding in Network Packet Delays," technical report, Dartmouth College, 2005.
[5] S. Cabuk, "IP Covert Timing Channels: Design and Detection," *Proc. ACM Conf. Computer and Comm. Security (CCS)*, pp. 178-187, 2004.
[6] S. Chong, C. Skalka, and J.A. Vaughan, "Self-Identifying Sensor Data," *Proc. Information Processing in Sensor Networks (IPSN)*, pp. 82-93, 2010.
[7] I. Cox and M. Miller, "Electronic Watermarking: The First 50 Years," *Proc. IEEE Workshop Multimedia Signal Processing* pp. 225-230, 2001.
[8] C. Dai, D. Lin, E. Bertino, and M. Kantarcioglu, "An Approach to Evaluate Data Trustworthiness Based on Data Provenance," *Proc. Fifth VLDB Workshop Secure Data Management (SDM)*, pp. 82-98, 2008.
[9] R.C. Dixon, *Spread Spectrum Systems: With Commercial Applications*, third ed. John Wiley and Sons, Inc., 1994.
[10] J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS)*, p. 186, 2001.
[11] I. Foster, J. Vockler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," *Proc. Conf. Scientific and Statistical Database Management*, pp. 37-46, 2002.
[12] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," *Proc. Conf. File and Storage Technologies (FAST)*, pp. 1-14, 2009.
[13] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. Ann. Hawaii Int'l Conf. System Sciences*, pp. 3005-3014, 2000.
[14] A. Houmansadr, N. Kiyavash, and N. Borisov, "Multi-Flow Attack Resistant Watermarks for Network Flows," *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing*, pp. 1497-1500, 2009.
[15] N.B.A. Houmansadr and N. Kiyavash, "Rainbow: A Robust and Invisible Non-Blind Watermark for Network Flows," *Proc. Network and Distributed System Security Symp. (NDSS)*, 2009.
[16] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A Link Layer Security Architecture for Wireless Sensor Networks," *Proc. Int'l Conf. Embedded Networked Sensor Systems*, pp. 162-175, 2004.

[17] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-Flow Attacks against Network Flow Watermarking Schemes," *Proc. USENIX Conf. Security Symp.,* pp. 307-320, 2008.
[18] H. Lim, Y. Moon, and E. Bertino, "Provenance-Based Trustworthiness Assessment in Sensor Networks," *Proc. Workshop Data Management for Sensor Networks,* pp. 2-7, 2010.
[19] X. Luo, J. Zhang, R. Perdisci, and W. Lee, "On the Secrecy of Spread-Spectrum Flow Watermarks," *Proc. European Conf. Research in Computer Security (ESORICS),* pp. 232-248, 2010.
[20] K.-K. Muniswamy-Reddy, D. Holland, U. Braun, and M. Seltzer, "Provenance-Aware Storage Systems," *Proc. USENIX Ann. Technical Conf.,* p. 4, 2006.
[21] U. Park and J. Heidemann, *Provenance and Annotation of Data and Processes,* pp. 280-292, Springer-Verlag, 2008.
[22] P. Peng, P. Ning, and D.S. Reeves, "On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques," *Proc. IEEE Symp. Security and Privacy (SP),* pp. 334-349, 2006.
[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "Spins: Security Protocols for Sensor Networks," *Wireless Networks,* vol. 8, pp. 521-534, Sept. 2002.
[24] T.G. Roosta, "Attacks and Defenses of Ubiquitous Sensor Networks," PhD thesis, Univ. of California, 2008.
[25] Y.L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in E-Science," *SIGMOD Record,* vol. 34, pp. 31-36, 2005.
[26] A. Syalim, T. Nishide, and K. Sakurai, "Preserving Integrity and Confidentiality of a Directed Acyclic Graph Model of Provenance," *Proc. Working Conf. Data and Applications Security and Privacy,* pp. 311-318, 2010.
[27] N. Vijayakumar and B. Plale, "Towards Low Overhead Provenance Tracking in Near Real-Time Stream Filtering," *Provenance and Annotation of Data,* vol. 4145, pp. 46-54, 2006.
[28] X. Wang, S. Chen, and S. Jajodia, "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems," *Proc. IEEE Symp. Security and Privacy (SP),* pp. 116-130, 2007.
[29] X. Wang and P. Mohapatra, "Provenance Based Information Trustworthiness Evaluation in Multi-Hop Networks," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM),* 2010.
[30] X. Wang and D.S. Reeves, "Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Interpacket Delays," *Proc. ACM Conf. Computer and Comm. Security (CCS),* pp. 20-29, 2003.

Salmin Sultana is working toward the PhD degree in computer engineering in the School of ECE, Purdue University. Her research interests include data provenance and security and fault tolerance of distributed systems, e.g., cloud computing, power grid, high-performance computing, etc. She is a member of the Center for Education and Research in Information Assurance and Security (CERIAS).

Mohamed Shehab is an assistant professor in the Department of Software and Information Systems, University of North Carolina at Charlotte. He leads the Lab of Information Integration Security and Privacy. His research interests include network and information security, especially in the design and implementation of distributed access-control protocols to cope with the requirements of emerging social networks, mobile applications, web services, and p2p environments. He is the recipient of the Google Research Award, in addition his research has been funded by US National Science Foundation (NSF), DOD, and Google. He is a member of the IEEE, the IEEE Computer Society, and the ACM.

Elisa Bertino is a professor of computer science at Purdue University and serves as a research director of CERIAS. Previously, she was a faculty member in the Department of Computer Science and Communication at the University of Milan. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She received the 2002 IEEE Computer Society Technical Achievement Award for outstanding contributions to database systems and database security and advanced data management systems and the 2005 IEEE Computer Society Tsutomu Kanai Award for pioneering and innovative research contributions to secure distributed systems. She is a fellow of the IEEE and the ACM.