Dictionary Based Secure Provenance Compression for Wireless Sensor Networks

Changda Wang*, Syed Rafiul Hussain*, *Member, IEEE*, and Elisa Bertino, *Fellow, IEEE*

Abstract—Due to energy and bandwidth limitations of wireless sensor networks (WSNs), it is crucial that data provenance for these networks be as compact as possible. Even if lossy compression techniques are used for encoding provenance information, the size of the provenance increases with the number of nodes traversed by the network packets. To address such issues, we propose a dictionary based provenance scheme. In our approach, each sensor node in the network stores a packet path dictionary. With the support of this dictionary, a path index instead of the path itself is enclosed with each packet. Since the packet path index is a code word of a dictionary, its size is independent of the number of nodes present in the packet's path. Furthermore, as our scheme binds the packet and its provenance through an AM-FM sketch and uses a secure packet sequence number generation technique, it can defend against most of the known provenance attacks. Through simulation and experimental results, we show that our scheme outperforms other compact provenance schemes with respect to provenance size, robustness, and energy consumption.

Index Terms—Provenance, dictionary based compression, sensor network

1 Introduction

Large-scale sensor networks are deployed in numerous application domains, including medical monitoring, environmental monitoring, surveillance, home security, military operations, industrial machine monitoring, etc. In these application domains, sensors vary from miniature, body-worn sensors to external sensors such as video cameras or positioning devices. The diversity of such network environments requires to adopt techniques that can ensure the trustworthiness of data across the network [1].

Since provenance [2] records the history of both data acquisition and transmission, it is considered as an effective mechanism to evaluate the trustworthiness of data. It also provides the information about the operations performed on data. However, reducing the size of the provenance is crucial in large-scale sensor networks. Sensor nodes in these networks may not be able to record and manipulate very large provenance data due to storage and computational resource constraints. Besides, transmission channels may not have sufficient capacity for transmitting large provenance data. Although most of the recent approaches [3], [4] focus mainly on provenance modeling, collection, and querying, a few of them [2], [5], [6], [7], [8] address the size and trustworthiness of provenance in sensor networks. In this paper, we investigate the problem of efficient and secure compression of provenance information in wireless sensor networks (WSNs). The problem imposes a set of challenges:

- The compression of provenance should be as compact as possible so that for large-scale WSNs the provenance size does not increase with the number of nodes traversed by the network packets.
- The compression or encoding should ensure that the system does not lose any provenance information after decoding.
- The trustworthiness of the provenance must be assured.

To reduce the provenance size for large-scale WSNs, earlier approaches [2], [5], [9] use lossy compression techniques. For example, some provenance schemes [2], [5] only record the data processing or routing nodes, but discard the order in which they are traversed by the network packets. Hence, these approaches fail to provide accurate path provenance of data packets. On the other hand, if a model randomly generates a message, according to Shannon's theory the message cannot be encoded into a smaller number of bits (on average) than the entropy of that model. Entropy is a measure of the uncertainty in a random variable which quantifies the expected value of the information contained in a message [10]. So, the theorem sets a lower bound on the size of provenance for an entropy based model.

In order to address the drawbacks of lossy compression techniques and to address the limitation of entropy lower bound, we propose a dictionary based approach to encode the sensor data provenance. Our proposed technique compresses the packets' paths and represents them using distinct *indexes*. These *indexes* are stored in a dictionary. With the support of this dictionary, a fixed size path index can be used to represent a path of arbitrary length [11], [12]. Therefore, the use of dictionary based method allows one to keep the size of

*These authors contributed equally to this work.

- C. Wang is with the School of Computer Science and Communication Engineering, Jiangsu University, China. At the time this work was done, he was with the Department of Computer Science, Purdue University. E-mail: changda@ujs.edu.cn.
- S. R. Hussain is with the Department of Computer Science, Purdue University. E-mail: hussain1@purdue.edu.
- E. Bertino is with the Cyber Center and the Department of Computer Science, Purdue University. E-mail: bertino@purdue.edu.

Manuscript received 19 Apr. 2014; revised 17 Sept. 2014; accepted 26 Dec. 2014. Date of publication 9 Feb. 2015; date of current version 20 Jan. 2016.
Recommended for acceptance by H. Frey.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2402156

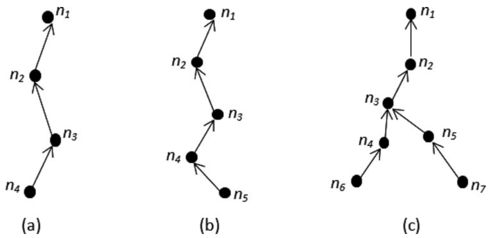


Fig. 1. Provenance graphs of sensor networks.

a compressed path smaller than the path's entropy at the cost of additional storage space for dictionaries.

As opposed to existing approaches, our proposed mechanism compresses the provenance into a much smaller size which is smaller than the entropy. Hence, transmission channels do not experience difficulties with handling provenance data in large scale WSNs. Moreover, our compression scheme is lossless in contrast to other lightweight mechanisms which are lossy in nature. The simulation and experimental results of our dictionary based lossless compression approach show that it generates provenance of much smaller size than that of existing lossy provenance schemes.

Our specific contributions are:

- We propose a dictionary based provenance scheme which is the most compact and lossless scheme up to date.
- We design an efficient, and distributed algorithm for encoding the provenance information as well as a centralized approach for its decoding.
- We introduce a secure packet sequence number generation mechanism and use the AM-FM sketch technique to secure the provenance.
- We perform a formal security analysis and an extensive performance evaluation of our proposed provenance scheme.

The rest of the paper is organized as follows: Section 2 provides an overview of our method. Section 3 introduces the system model. Section 4 describes the provenance encoding, binding, and decoding algorithms. Section 5 discusses a detailed case study. Section 6 presents a recursive provenance encoding and decoding scheme. Section 7 discusses the security and performance of our approach. Sections 8 and 9 present the simulation and experimental results, respectively. Section 10 surveys related work, and finally, Section 11 concludes the paper.

2 Overview of Our Approach

Before introducing the underlying concepts and formal algorithms, we give an overview of our dictionary based provenance scheme. Here, we discuss (i) how our scheme works, (ii) how it ensures the security of provenance, and (iii) what are the implementation challenges of our scheme.

In our approach, we consider a multi-hop WSN consisting of a number of sensor nodes and a base station (BS). The BS collects data packets and also their provenance information such as source nodes, traversed paths etc. Therefore, provenance of data in a sensor network can be represented as a directed graph, referred to as *provenance graph*, where vertexes represent the provenance records at sensor nodes and directed edges represent the transmissions of data packets from one node to another.

TABLE 1
Dictionary of Indexes

Linear Path	Index
$\{n_4, n_3, n_2, n_1\}$	$\langle n_4, n_1\rangle$
$\{n_5, n_4, n_3, n_2, n_1\}$	$\langle n_5, n_1\rangle$
$\{n_6, n_4, n_3\}$	$\langle n_6, n_3\rangle$
$\{n_7, n_5, n_3\}$	$\langle n_7, n_3\rangle$
$\{n_3, n_2, n_1\}$	$\langle n_3, n_1\rangle$

To understand how our scheme works, consider the *provenance graphs* in Figs. 1a and 1b having four and five nodes respectively. Without any compression, paths in such *provenance graphs* can be encoded as $\langle n_4, n_3, n_2, n_1\rangle$ and $\langle n_5, n_4, n_3, n_2, n_1\rangle$ where n_i is the ID of ith sensor node. Clearly, the provenance with five nodes has larger size than that with four nodes.

However, in our approach, paths in the *provenance graphs* of Figs. 1a and 1b are encoded as $\langle n_4, n_1\rangle$ and $\langle n_5, n_1\rangle$ respectively which are of equal size. Here, $\langle n_4, n_1\rangle$ and $\langle n_5, n_1\rangle$ are called *indexes* of the paths $\{n_4, n_3, n_2, n_1\}$ and $\{n_5, n_4, n_3, n_2, n_1\}$ respectively. These linear paths and their corresponding *indexes* are stored in a *dictionary* as shown in Table 1. Hence, without loss of generality, if the graph is a linear path $\{n_M, n_{M-1}, \ldots, n_2, n_1\}$, it can be simply represented by the index $\langle n_M, n_1\rangle$. We name such graphs as *linear topology graphs*. Note that the *indexes* can be built in different ways. The way it is presented in Table 1 is just one of them.

In more general graphs, as shown in Fig. 1c, multiple paths are connected as branches in a tree. We denote such graphs as *tree topology graphs*. While dealing with these graphs, we use semicolon as a delimiter to separate the branches of the tree. For example, we encode the graph in Fig. 1c as $\langle\langle n_6, n_3\rangle; \langle n_7, n_3\rangle; \langle n_3, n_1\rangle\rangle$. For simplicity, we represent it as $\langle n_6, n_3; n_7, n_3; n_3, n_1\rangle$.

To ensure the security of provenance, we use the AM-FM sketch scheme [13] which binds the packet content and its provenance together. The AM-FM sketch is a distributed, node-level digital signature scheme. Using this scheme, whenever a sensor node generates or forwards a data packet, it creates the digest of the data and signs this digest prior to sending the packet to the next node.

The most challenging issues in the design of our dictionary based provenance scheme are how to build the dictionary and share it across the network. Existing approaches, such as [11] and [12], build dictionary for the substrings that appear multiple times in a message and assign each substring a unique index of shorter length. For instance, given the message *"abcdxabcd"*, the substring *"abcd"* shows up two times. Therefore, the dictionary based method builds an index for *"abcd"* at its first appearance and then uses that index to represent all subsequent *"abcd"*. Thus it compresses a large string to a smaller one.

However, existing dictionary based compression techniques [11], [12] cannot be applied directly to compress provenance, because these methods build dictionary based on recurring substrings in the same message. But in sensor networks, *provenance graphs* are acyclic. In other words, each node appears at most once in a packet path. So, if we consider the sequence of nodes' IDs along a packet path as a message, no recurring substrings can be found.

Hence, to build the dictionary for a packet's path, we propose to find the common substrings between earlier paths and the current path. In practice, some packets' paths or part of them are reused with high probability due to better channel quality or lower energy consumption. As a result, recurring subsequences of nodes' IDs can be found frequently in those packets' paths. Therefore, we can compress all successive packets' paths using the *indexes* of these recurring subsequences.

3 System Model

In this section, we introduce the network model, and the data model that we consider for our proposed dictionary based provenance scheme. We also present the provenance model along with some useful concepts that are used in our proposed scheme. Finally, we describe the adversary model which summarizes the security objectives we aim to achieve.

3.1 Network Model

The network is modeled as a directed graph $G(N, L)$, where $N = \{n_i \mid 1 \leq i \leq \mid N \mid\}$ is the set of network nodes and $L = \{l_{ij} \mid 1 \leq i, j \leq \mid N \mid\}$ is the set of links. The cardinality of set N is denoted by $|N|$. A directed edge from node n_i to n_j, denoted as l_{ij}, indicates a one hop communication from node n_i to n_j.

We consider the following details for our network model:

- The BS assigns each node a unique identifier n_i, a counter $count_i$, and an encryption key k_i that is shared between the BS and that particular node. All this information is embedded in a node before its deployment. Also, the BS itself maintains a counter, $count'_i$, for each node n_i. Initially, $count_i$ and $count'_i$ are both set to 0. When a node n_i generates a packet p, it increments the value of $count_i$ by 1. Upon receiving the packet p, the BS increments the n_i specific counter $count'_i$ by 1.
- Routing paths may change over time due to node failure, mobility, link quality degradation, etc. Hence, our model supports both stationary and dynamic natures of sensor networks.
- The BS collects data from nodes in rounds. The round counter, $rc = 0, 1, \ldots$ is known to all nodes.
- The BS has no constraints with respect to energy, storage space, security, and computational capability.

3.2 Data Model

We assume that the sensor nodes in a network generate data periodically. We name these nodes as *data source nodes*. A node may also receive data from another node in order to forward such data towards the BS. We call this node a *forwarder node*. While being routed towards the BS, packets that fulfil the aggregation requirements are aggregated according to some hierarchical dissemination scheme [14]. So, if a node receives such packets from multiple nodes, it forwards the aggregated packet to the next node. We refer to such a node as *aggregator node*.

Fig. 1c shows a *provenance graph* where nodes n_6 and n_7 generate data. So, these nodes are *data source nodes*. The generated data are then received by nodes n_4 and n_5 respectively. Both these nodes forward the respective data to node n_3. Hence, n_4 and n_5 are *forwarder nodes*. As node n_3 aggregates data received simultaneously from multiple nodes, it is called an *aggregator node*.

Each packet contains: (i) packet's sequence number, (ii) its source node's ID i.e., the ID of the node that generates the packet, (iii) data value, (iv) provenance record, and (v) message authentication code (MAC).

In our model, a packet p generated at node n_i has a sequence number seq computed as $E_{k_i}(count_i||rc)$, where E is an encryption function, and $||$ denotes concatenation. The other parameters k_i, $count_i$, and rc are as defined in Section 3.1.

3.3 Provenance Model

Our proposed dictionary based scheme encodes provenance records at nodes that are involved at each step of data processing and transmission. In this section, we introduce the following concepts that are used in our provenance scheme.

Network data provenance. The provenance of a data packet p is represented as a graph $T(V_p, E_p)$, referred to as *provenance graph*. Each vertex $v_i \in V_p$ represented as (n_i, seq, agr) is a provenance record of p at node n_i. Here, n_i is the host node of v_i, i.e., $Host(v_i) = n_i$, seq is the sequence number of p, and agr is the aggregation record of packets $\{seq_{i1}, \ldots, seq_{iM}\}$ at node n_i. If no packets are aggregated at node n_i, agr is set to $\emptyset$ and the sequence number is the same as that of p. But, if agr is not empty, i.e., n_i is an *aggregator node*, it generates a new packet with aggregated data value and gives it a new sequence number. An edge $e \in E_p$ represents one hop connection between two vertices in V_p.

A *provenance graph* $T(V_p, E_p)$ satisfies the following properties:

- $T(Host(V_p), E_p) \subset G(N, L)$ is a tree rooted at the BS.
- If node n_i is on the path of a packet p, there is a $v \in V_p$ for which $Host(v) = n_i$.

We illustrate *data provenance* with the following examples. If node n_4 (in Fig. 1a) generates a packet p with sequence number seq_p, the *provenance graph* of p is represented as $T(V_p, E_p)$ where $V_p = \{v_4, v_3, v_2, v_1\}$. As there is no aggregation at node n_4, v_4 is $(n_4, seq_p, \emptyset)$. Again if nodes n_6 and n_7 (in Fig. 1c) generate packets q and r separately with sequence numbers seq_q and seq_r, the *provenance graphs* of q and r are represented as $T(V_q, E_q)$ and $T(V_r, E_r)$ respectively. Here, $V_q = \{v_6, v_4, v_3, v_2, v_1\}$ and $V_r = \{v_7, v_5, v_3, v_2, v_1\}$. If node n_3 aggregates data packets q and r, $v_3 = (n_3, seq_t, \{seq_q, seq_r\})$ where seq_t is the sequence number of the aggregated packet.

Dictionary index. Dictionary index (*dicIndex*) is used to represent the compression of a linear path. We define *dicIndex* as $\langle n_b, n_e \rangle$, where n_b and n_e respectively represents the beginning and ending nodes on a linear path $\{n_b, ..., n_e\}$. For example, the *dicIndex* of path $\{n_4, n_3, n_2, n_1\}$ in Fig. 1a is represented as $\langle n_4, n_1 \rangle$.

Packet path index. Assume that a packet p traverses the path $\{n_M, n_{M-1}, \ldots, n_1\}$ to reach the BS. If there is no *dicIndex* that can compress the path, the path index of p includes the entire path $\langle n_M, n_{M-1}, \ldots, n_1 \rangle$. Now, if such *dicIndex* exists, it replaces the corresponding path snippet that it compresses in the path and thus the *pathIndex* of

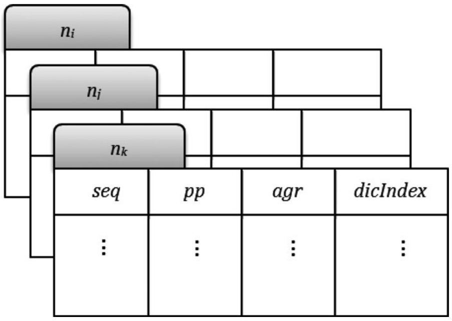


Fig. 2. Packet path dictionary (*PPD*).

packet p is formed. In the case of aggregated packets, we separate the paths in $pathIndex$ by using semicolons as delimiter.

For example, in Fig. 1a, if the $dicIndex = \langle n_4, n_1\rangle$ is built before a packet p traverses the path $\{n_4, n_3, n_2, n_1\}$, the $pathIndex$ of p at node n_1 is formed as $\langle n_4, n_1\rangle$. And in Fig. 1c, assuming that the $dicIndexes$ $\langle n_6, n_3\rangle$, and $\langle n_7, n_3\rangle$ are previously built, the $pathIndex$ of the aggregated packet at node n_3 is $\langle n_6, n_3; n_7, n_3\rangle$. Here, the semicolons are used to separate the branches of the tree.

Packet Path Dictionary. Packet path dictionary (*PPD*) at some node n_i is a database that keeps provenance information of the packets generated, forwarded or aggregated by node n_i. When a packet p traverses a path from node n_k to n_i $(k \neq i)$, the *PPD* at n_i stores information regarding this path. It also keeps packets' sequence numbers, dictionary indexes of their paths, and aggregation records at node n_i.

Provenance index. Provenance index, denoted as $prIndex = (v, pathIndex)$, represents the provenance encoding at vertex v of the provenance graph.

3.4 Adversary Model

During the network operations, any node but the BS may be compromised by adversaries, i.e., only the BS is assumed to be trusted. An adversary can eavesdrop in the network and collect confidential information by means of packet sniffing, traffic analysis etc. It can compromise legitimate nodes and extract critical information such as keys, codes, or data. It may also use these nodes to perform attacks cooperatively, e.g., providing false route information, dropping, injecting or altering packets. The BS cannot distinguish these compromised nodes from the benign ones. However, the BS needs to verify the integrity, authenticity, and availability of provenance which we consider to be the most important security objectives in our scheme. Therefore, we do not emphasize the encryption of provenance data. If confidentiality is required, we can simply encrypt the provenance by the secret keys of the respective nodes. The security properties that we aim to achieve in our scheme are summarized below:

- Any unauthorized change in packet content or provenance information is detected.
- An adversary cannot inject counterfeit information in the provenance or remove benign nodes from it.
- A malicious node cannot drop packets without being identified.
- Packet replay attacks are detectable.

TABLE 2
Table for Node n_6 in the *PPD* of n_6

Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_6\}$	$\emptyset$	$\langle n_6, \emptyset\rangle$
⋮	⋮	⋮	⋮

4 Provenance Scheme

We develop a distributed mechanism to encode provenance at sensor nodes and a centralized approach to decode provenance at the BS. With the support of the *PPD* at each node, our proposed approach includes a $pathIndex$ in the provenance record instead of the path itself and thus compresses provenance to a much smaller size than that of the original one. Also, to ensure the protection against any unauthorized modification, the packet and its provenance are bound together by a secure message authentication code. While decoding, the BS verifies the MAC and then extracts the *provenance graph* for the packet it receives.

Before going into the details of our encoding and decoding mechanism, we present the structure of a *PPD*. Let a node n_i has a dictionary as shown in Fig. 2. n_i keeps a table for each node n_k $(k \neq i)$ which appears before n_i along a packet's path towards the BS. Each row in such a table has four fields *seq*, *pp*, *agr*, *dicIndex*, where *seq* is the packet sequence number, *pp* is the packet's path $\{n_k, \ldots, n_i\}$ from node n_k to n_i, *agr* is the aggregation record at node n_i, and $dicIndex$ is $\langle n_k, n_i\rangle$ is described in Section 3.3. Querying $dicIndex = \langle n_k, n_i\rangle$ from the *PPD* retrieves the packet path $\{n_k, \ldots, n_i\}$ if such a *pp* exists in the *PPD*. Node n_i also stores the information about the packets it generates in its *PPD*.

4.1 Provenance Encoding

For a data packet, provenance encoding refers to the creation of compressed provenance at each node on the packet's path. Our proposed mechanism uses the *PPD*s of those nodes to efficiently encode provenance records.

During the process of provenance encoding, each node along a packet's path is assumed to be one of these three: *data source node*, *forwarder node*, and *aggregator node*. The functionalities of these nodes are to generate, forward, and aggregate packets, respectively. We start encoding the provenance from *data source node* and then update it in subsequent nodes along the packets' paths. We now describe how we encode provenance at different types of nodes. Initially, each node's *PPD* is set to $\emptyset$.

Encoding at a data source node. When a *data source node* n_i generates a packet p, it creates a row in its own table located in its *PPD*. The *seq* field of that row is set as the sequence number of the generated packet. Since the newly generated packet has not yet traversed any other nodes in the network, *pp*, *agr*, and $dicIndex$ are set as $\{n_i\}$, $\emptyset$, and $\langle n_i, \emptyset\rangle$, respectively. The *data source node* then forwards the packet to the next node towards the BS. For instance, in Fig. 1c, if node n_6 generates a packet with sequence number seq_1, a row is created in its own table as shown in Table 2. Then this packet is sent to the next node n_4 with provenance record $prIndex = (v_6, \langle n_6, \emptyset\rangle)$, where $v_6 \in V_{seq1}$ is $(n_6, seq_1, \emptyset)$.

TABLE 3
Table for Node n_6 in the *PPD* of n_4

Sequence Number	Packet Path	Aggregation	Dictionary Index
(seq)	(pp)	(agr)	$(dicIndex)$
seq_1	$\{n_6, n_4\}$	$\emptyset$	$\langle n_6, n_4\rangle$
⋮	⋮	⋮	⋮

Encoding at a forwarder node. When a node n_i receives a packet from some node n_k, it creates a new row in the table for n_k in its *PPD*. It appends its ID to the current path of the received packet and sets this as pp of that new row. If the packet is simply forwarded without any aggregation, *agr* is $\emptyset$. Finally, the $dicIndex$ is set as $\langle n_k, n_i\rangle$. The provenance record of the received packet is then updated accordingly and forwarded to the next node.

Fig. 1c shows that a packet generated by n_6 is received by node n_4 and then forwarded to the next node n_3. Upon receiving the packet, n_4 creates a row in the table for n_6 in its *PPD* as shown in Table 3. Then the provenance record $(v_4, \langle n_6, n_4\rangle)$ is sent to n_3 where $v_4 \in V_{seq1}$ is $(n_4, seq_1, \emptyset)$.

When n_3 receives the packet from n_4, it updates its *PPD* according to Table 4. It then simply forwards the packet to n_2 with provenance record $(v_3, \langle n_6, n_4, n_3\rangle)$ where v_3 is $(n_3, seq_1, \emptyset)$.

Encoding at an aggregator node. If an *aggregator node* n_i simultaneously receives M packets $seq_{i1}, seq_{i2}, \ldots, seq_{iM}$, it aggregates them into a single packet seq_i. It then creates a new row in its own table located in its *PPD*. It sets *seq*, *pp*, *agr*, and $dicIndex$ for the new entry as seq_i, $\{n_i\}$, $\{seq_{i1}, seq_{i2}, \ldots, seq_{iM}\}$ and $\langle n_i, \emptyset\rangle$ respectively. If the received packets $seq_{i1}, seq_{i2}, \ldots, seq_{iM}$ are generated or aggregated at nodes $n_{b1}, n_{b2}, \ldots, n_{bM}$ respectively, the $pathIndex$ that is sent to the next node is $\langle n_{b1}, n_i; \ldots; n_{bM}, n_i\rangle$.

To illustrate the above scenario, we assume that node n_3 in Fig. 1c receives two packets with sequence numbers seq_1 and seq_2 generated by nodes n_6 and n_7 respectively. It then aggregates these packets into a new one with sequence number seq_3. The entry of a new row in the table of n_3 in its *PPD* is presented in Table 5. Finally, the packet with provenance record $(v_3, \langle n_6, n_3; n_7, n_3\rangle)$ is sent to node n_2 where v_3 is $(n_3, seq_3, \{seq_1, seq_2\})$.

Upon receiving the packet, node n_2 stores the path information from n_6, n_4, and n_3 in its *PPD*. The tables for these nodes are shown in Table 6. In this way, the *PPD*s at all nodes along the packet's path are updated. If this path is reused by other subsequent packets, these *PPD*s are used to compress the provenance records to a much smaller size. For example, if node n_3 receives another packet from n_6 with sequence number seq_4, it does not send the entire path $\{n_6, n_4, n_3\}$ to n_2. It is sufficient to send only the $dicIndex$ $\langle n_6, n_3\rangle$ that is stored in the *PPD* of node n_3 for the path $\{n_6, n_4, n_3\}$.

TABLE 4
Table for Nodes n_4 and n_6 in the *PPD* of n_3

Table for node n_4 in the *PPD* of n_3

Sequence Number	Packet Path	Aggregation	Dictionary Index
(seq)	(pp)	(agr)	$(dicIndex)$
seq_1	$\{n_4, n_3\}$	$\emptyset$	$\langle n_4, n_3\rangle$
⋮	⋮	⋮	⋮

Table for node n_6 in the *PPD* of n_3

Sequence Number	Packet Path	Aggregation	Dictionary Index
(seq)	(pp)	(agr)	$(dicIndex)$
seq_1	$\{n_6, n_4, n_3\}$	$\emptyset$	$\langle n_6, n_3\rangle$
⋮	⋮	⋮	⋮

TABLE 5
Table for Node n_3 in the *PPD* of n_3

Sequence Number	Packet Path	Aggregation	Dictionary Index
(seq)	(pp)	(agr)	$(dicIndex)$
seq_3	$\{n_3\}$	$\{seq_1, seq_2\}$	$\langle n_3, \emptyset\rangle$
⋮	⋮	⋮	⋮

When node n_2 receives the packet with $pathIndex = \langle n_6, n_3\rangle$, it replaces n_3 with its own ID and looks up the $dicIndex$ $\langle n_6, n_2\rangle$ in the table it created for node n_6. It then retrieves the entire path $\{n_6, n_4, n_3, n_2\}$ which is stored for $\langle n_6, n_2\rangle$. In the next step, n_2 forwards the packet to n_1 with $pathIndex = \langle n_6, n_2\rangle$ instead of the entire path traversed by the packet. Thus, the size of the provenance is compressed using the dictionaries stored at nodes. The steps of the provenance encoding are presented in Algorithm 1.

Algorithm 1. Provenance Encoding

```
Input: $(n_i, seq_i)$
Output: $prIndex = (v, pathIndex)$
if $n_i$ is a data source node then
    $prIndex.v = v_i$
    $pp = n_i$
    $agr = \emptyset$
    $prIndex.pathIndex = \langle n_i, \emptyset\rangle$
end if
if $n_i$ is a forwarder node then
    $prIndex.v = v_i$
    $pp = pp \bigcup n_i$
    $agr = \emptyset$
    $prIndex.pathIndex = \langle n_k, n_i\rangle$
end if
if $n_i$ is an aggregator node then
    $prIndex.v = v_i$
    $pp = n_i$
    $agr = \{seq_{i1}, seq_{i2}, \ldots, seq_{iM}\}$
    $prIndex.pathIndex = \langle n_{b1}, n_i; \ldots; n_{bM}, n_i\rangle$
end if
```

Note that each row in the *PPD* contains the path information and hence the corresponding dictionary index can be built accordingly whenever it is required. However, precomputing and storing dictionary index speed up the retrieval process of path information and incur less delay in packet transmission.

4.2 Provenance Binding

We enclose only the $prIndex$ as provenance record along with the data packet to save energy and bandwidth. In order to protect against attacks, any unauthorized modification of packet content or its associated $prIndex$ needs to be

TABLE 6
Table for Nodes n_3, n_4 and n_6 in the *PPD* of n_2

Table for node n_3 in the *PPD* of n_2			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_3, n_2\}$	$\emptyset$	$\langle n_3, n_2\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Table for node n_4 in the *PPD* of n_2			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_4, n_3, n_2\}$	$\emptyset$	$\langle n_4, n_2\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Table for node n_6 in the *PPD* of n_2			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_6, n_4, n_3, n_2\}$	$\emptyset$	$\langle n_6, n_2\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

detected. Simply encrypting the packet content or its provenance is not feasible due to the high cost of encryption. Another approach is to bind together the packet and its $prIndex$ by MAC and then encrypt the MAC with the private key of each node along the packet's path.

Traditional MAC schemes, e.g., MD5 or SHA-1, are designed for centralized scenarios. In such cases, each node in the packet's path generates an independent MD5 or SHA-1. If we chain all the MACs together, the combined size of the resulting MAC increases in proportion to the number of nodes on the path. Such a MAC chain incurs additional energy and channel bandwidth.

To address this issue, we adopt a provenance binding technique which is a direct application of the AM-FM sketch mechanism [13]. It is a distributed message digest mechanism through which we bind a packet and its provenance together at each node along its path. Generally, this approach can restrain the MAC size from growing beyond a range $[(1-\varepsilon)k, (1+\varepsilon)2k]$ with probability $(1-\delta)$ where k is the sample size of the provenance record, $0 < \delta < 1$ and ε ($\varepsilon < 1$) is the false positive and false negative rates related to k, if $k \geq O(\frac{log(2/\delta)}{\varepsilon^2})$. However, with a certain statistical confidence we assume that any unauthorized packet content or provenance modification can be detected by using the AM-FM sketch if the false positive and false negative rates are insignificant and controllable.

4.3 Provenance Decoding

When the BS receives a packet, it verifies the integrity of the protected data through the AM-FM evaluation process. If the verification confirms that the protected data are trustworthy, the BS accepts the packet, otherwise, the packet is dropped. For an accepted packet p, the *provenance graph* represented as $T(V_p, E_p)$ is retrieved by looking up its $pathIndex$ in the *PPD* of the BS.

To explain the decoding process, we again make reference to Fig. 1c. We assume that node n_1 is the BS in the network and it previously received a packet with sequence number seq_1 generated by node n_6. Hence, it has all the information of the path from n_6 to itself stored in its *PPD* as shown in Table 7. Now assume that node n_6 generates another packet p with sequence number seq_4 and the BS receives that from n_2. While forwarding the packet to the BS, n_2 embeds the provenance record $prIndex = (v_2, \langle n_6, n_2\rangle)$ in the packet as described in Section 4.1. Here, $v_2 = (n_2, seq_4, \emptyset)$. The BS then decodes the provenance of p using the *PPD* stored in it.

So, when the BS receives $prIndex = (v_2, \langle n_6, n_2\rangle)$ from node n_2, it forms $\langle n_6, n_1\rangle$ by replacing n_2 with its own ID and looks up in the table it has for node n_6 in its *PPD*. The *pp* it retrieves from its *PPD* is $\{n_6, n_4, n_3, n_2, n_1\}$. Thus the BS decodes the provenance information and retrieves the path of the received packet using the dictionary *PPD* stored in it.

If the BS receives a packet with a non-empty aggregation record (agr), i.e., the $pathIndex$ of the packet's provenance record has semicolon(s), the BS finds the *aggregator node(s)* by locating the semicolons in the *pathIndex*. Then each path separated by the semicolon(s) is considered individually and looked up in the *PPD* in order to decode.

The steps of provenance decoding are presented in Algorithm 2.

5 Case Studies

In this section, we discuss how our provenance scheme works for different topologies. To illustrate our examples we assume that the BS in Fig. 3a previously received two data packets generated by nodes I and J respectively. Thus the paths $\{J, H, F, D, B, BS\}$ and $\{I, G, C, A, BS\}$ are already built and the $PPDs$ of the nodes on these paths are updated accordingly.

5.1 Dictionary Initialization

Without loss of generality, we analyze one of these two paths, i.e., $\{J, H, F, D, B, BS\}$. At the beginning, node J

TABLE 7
Table for Nodes n_2, n_3, n_4 and n_6 in the *PPD* of n_1

Table for node n_2 in the *PPD* of n_1			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_2, n_1\}$	$\emptyset$	$\langle n_2, n_1\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Table for node n_3 in the *PPD* of n_1			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_3, n_2, n_1\}$	$\emptyset$	$\langle n_3, n_1\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Table for node n_4 in the *PPD* of n_1			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_4, n_3, n_2, n_1\}$	$\emptyset$	$\langle n_4, n_1\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Table for node n_6 in the *PPD* of n_1			
Sequence Number (seq)	Packet Path (pp)	Aggregation (agr)	Dictionary Index ($dicIndex$)
seq_1	$\{n_6, n_4, n_3, n_2, n_1\}$	$\emptyset$	$\langle n_6, n_1\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

generates a packet with sequence number seq_{J_1}. It creates $v_J = \langle J, seq_{J_1}, \emptyset\rangle$ according to the provenance encoding algorithm and then transmits $prIndex = (v_J, \langle J, \emptyset\rangle)$ to the next node.

Algorithm 2. Provenance Decoding

```
Input: $prIndex = (v, pathIndex)$
Output: $T(V_p, E_p)$
if the AM-FM verification fails then
  drop the received packet
else
  if $v.agr = \emptyset$ then
  $T(V_p, E_p)$ = Query $pathIndex$ to $PPD$.
else
  $\psi$ = number of ';' in $pathIndex$ +1
  for i = 1 to $\psi$ do
    $path_i$= Query $branch_i$ of $pathIndex$ to $PPD$.
  end for
  $T(V_p, E_p) = (path_1; \ldots; path_\psi)$
end if
end if
```

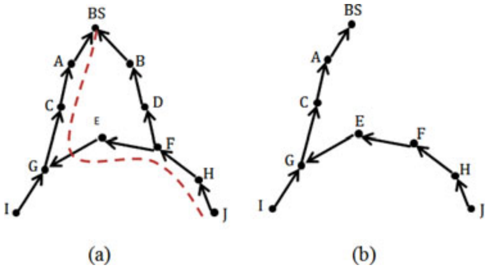


Fig. 3. Network topology for case studies.

Initially, *PPD* is set to $\emptyset$ for nodes J, H, F, D, B and BS. So, when the packet with sequence number seq_{J_1} arrives at the BS, the $pathIndex$ for that packet contains the entire path, i.e., $\langle J, H, F, D, B\rangle$. By parsing $\langle J, H, F, D, B\rangle$, the BS retrieves the packet's *provenance graph*. Table 8 lists the changes in the *PPDs* after the BS receives the packets from both nodes I and J.

5.2 Path Index Generation

To illustrate how path indexes are generated, we consider the following three scenarios:

1) Linear path reuse. Now consider the case in which node J generates a new packet with sequence number seq_{J_2} and transmits it along the same path as that of seq_{J_1}. Since the

TABLE 8
Entries in the *PPDs* at Different Nodes

Node ID	Sequence number	pp	$dicIndex$
J	seq_{J_1}	$\{J\}$	$\langle J, \emptyset\rangle$
H	seq_{J_1}	$\{J, H\}$	$\langle J, H\rangle$
F	seq_{J_1}	$\{J, H, F\}$	$\langle J, F\rangle$
D	seq_{J_1}	$\{J, H, F, D\}$	$\langle J, D\rangle$
B	seq_{J_1}	$\{J, H, F, D, B\}$	$\langle J, B\rangle$
E	$\emptyset$	$\emptyset$	$\emptyset$
I	seq_{I_1}	$\{I\}$	$\langle I, \emptyset\rangle$
G	seq_{I_1}	$\{I, G\}$	$\langle I, G\rangle$
C	seq_{I_1}	$\{I, G, C\}$	$\langle I, C\rangle$
A	seq_{I_1}	$\{I, G, C, A\}$	$\langle I, A\rangle$
BS	seq_{J_1}	$\{J, H, F, D, B, BS\}$	$\langle J, BS\rangle$
	seq_{I_1}	$\{I, G, C, A, BS\}$	$\langle I, BS\rangle$

PPDs of nodes along this path have been updated in the earlier round, the $pathIndex$ is $\langle J, B\rangle$ instead of $\langle J, H, F, D, B\rangle$ when the packet arrives at the BS.

2) Linear path combination. Next, consider the case in which the node J generates the third packet with sequence number seq_{J_3} which traverses a different path $\{J, H, F, E, G, C, A\}$ due to some change in the network topology (e.g., link quality degradation or mobility). Each node in this modified path updates its *PPD* according to Algorithm 1 while forwarding the packet seq_{J_3} towards the BS. As shown in Fig. 3a, this new path can be decomposed into three linear path snippets which are $\{J, H, F\}$, $\{F, E, G\}$, and $\{G, C, A\}$. Here, $\{J, H, F\}$ and $\{G, C, A\}$ are linear path snippets that are traversed by previous packets. So, these path snippets are encoded as $\langle J, F\rangle$ and $\langle G, A\rangle$ and thus the path $\{J, H, F, E, G, C, A\}$ can be compressed as $\langle J, F, E, G, A\rangle$.

To decode $pathIndex = \langle J, F, E, G, A\rangle$, the BS first parses it to see if there is any semicolon. Since there is no semicolon, the BS learns that J is the only *data source node*. Then the BS looks up $\langle G, A\rangle$, $\langle E, G\rangle$, $\langle F, E\rangle$ and $\langle J, F\rangle$ in the tables of nodes A, G, E, and F in its *PPD*. While the BS looks up A's table, $\langle G, A\rangle$ returns $\{G, C, A\}$. Then the BS looks up $\langle E, G\rangle$ in G's table. As this path snippet is not traversed by previous packets, the index is not found. So, the BS prepends E to $\{G, C, A\}$ that results in $\{E, G, C, A\}$. The BS continues to look up $\langle F, E\rangle$ in E's table. As the *PPD* of E is $\emptyset$ at that time, the BS prepends F to $\{E, G, C, A\}$ and thus gets $\{F, E, G, C, A\}$. Finally, the BS looks up $\langle J, F\rangle$ in F's table that returns $\{J, H, F\}$. The BS then prepends it to the previously decoded path $\{F, E, G, C, A\}$ and obtains the complete path $\{J, H, F, E, G, C, A\}$.

Now if a new packet with sequence number seq_{J_4} is generated and sent along the same path as that of seq_{J_3}, the $pathIndex$ will be compressed as $\langle J, A\rangle$. Hence, only two node IDs are enclosed in the provenance of the packet for each one hop transmission.

3) Tree Composition. We consider the tree topology in Fig. 3b, which is a subgraph of Fig. 3a. If node G simultaneously receives two packets that are generated by nodes I and J, it aggregates them into a single packet and sends this packet to the BS. Using the *PPD*s, the path snippets $\{J, H, F, E, G\}$ and $\{G, C, A\}$ can be compressed as $\langle J, G\rangle$ and $\langle G, A\rangle$ respectively. Hence, the *pathIndex* of the packet received by the BS is $\langle I, G; J, G; G, A\rangle$.

6 Recursive Provenance Scheme

To mitigate the provenance expansion caused by the *aggregator nodes*, we propose a recursive provenance scheme. For instance, as semicolons are used to represent tree topologies, the *pathIndex* $\langle n_6, n_3; n_7, n_3; n_3, n_1\rangle$ for the *provenance graph* shown in Fig. 1c contains node n_3 for three times.

Note that a vertex v_i is defined as (n_i, seq, agr), where agr is the data aggregation history at node n_i. Now, we define recursive provenance index as $prIndex_r = (v_i, pathIndex, flag)$, where *flag* is a binary variable to distinguish whether the first node in $pathIndex$ is a *data source node* or an *aggregator node*. When an *aggregator node* n_i receives M packets $seq_{i1}, seq_{i2}, \ldots, seq_{iM}$, it sets the sequence number, agr, $pathIndex$, and $flag$ as seq_i, $\{seq_{i1}, seq_{i2}, \ldots, seq_{iM}\}$, $\langle n_i, \emptyset\rangle$, and *TRUE*, respectively. Note that, in the recursive provenance scheme the *aggregator nodes* do not include the detailed aggregation record in the *pathIndex*. Hence, if the subsequent packets reuse the earlier paths, $prIndex_r$ can be compressed to a smaller size. The *aggregator node* then sets the pp, and $dicIndex$ as $\{n_i\}$ and $\langle n_i, \emptyset\rangle$ respectively in its *PPD*.

When the aggregated packet arrives at the BS, the BS checks the *flag* to see if the first node in the $pathIndex$ is an *aggregator node*. The BS then queries the agr in the *PPD* of that node to retrieve the paths which are aggregated. The same procedure continues until all the *aggregator nodes* in the packet's path are expanded.

For instance, consider the example of encoding at an *aggregator node* in Section 4.1 and assume node n_1 to be the BS. If n_1 receives a packet with $prIndex_r = (v_2, \langle n_3, n_2\rangle, TRUE)$, it replaces n_2 in $\langle n_3, n_2\rangle$ with its ID and looks up $\langle n_3, n_1\rangle$ in the table for node n_3 in its own *PPD*. Thus path snippet $\{n_3, n_2, n_1\}$ is retrieved for $\langle n_3, n_1\rangle$. Since the *flag* is set as *TRUE*, the BS learns that n_3 is an *aggregator node*. Therefore, the BS queries the $agr = \{seq_1, seq_2\}$ to the *PPD* of node n_3 and gets two $prIndex_r$ $(v_3, \langle n_6, n_3\rangle, FALSE)$ and $(v_3, \langle n_7, n_3\rangle, FALSE)$. As the *flags* are set as *FALSE*, the BS realizes that both n_6 and n_7 are *data source nodes*. Hence, the decoding process stops. The $pathIndexes$ $\langle n_6, n_3\rangle$ and $\langle n_7, n_3\rangle$ represent path snippets $\{n_6, n_4, n_3\}$ and $\{n_7, n_5, n_3\}$, respectively. Combining all three path snippets, the BS builds the complete provenance as $(n_6, n_4, n_3; n_7, n_5, n_3; n_3, n_2, n_1)$.

After a packet arrives at the BS, if the AM-FM sketch ensures that its data and provenance are consistent to each other and the $pathIndex$ matches with previously received packets, the BS does not send any query to the *aggregator nodes*.

7 Security and Performance

In this section, we analyze the security properties and evaluate the performance of our proposed provenance scheme.

7.1 Security Claims

We justify the following security properties of our scheme:

Claim 1. Without knowing the encryption key of a node, adversaries cannot inject counterfeit provenance at that node. Also, it cannot pass the AM-FM sketch verification at the BS.

Justification. Given a packet's path $\{n_1, n_2, \ldots, n_M\}$, its provenance record can be represented as $\{pr_1, pr_2 \ldots, pr_M\}$. Here, pr_i is the provenance of that packet at node n_i. To create the message authentication code, the AM-FM sketch binds every digested pr_i with the packet using the encryption key k_i. Thus, without knowing the key of n_i, pr_i cannot be injected. Even if pr_i is injected unauthentically, the AM-FM verification fails for the corresponding data and its provenance at the BS.

Claim 2. If n_i is a benign node in a packet path $\{n_1, n_2, \ldots, n_M\}$, an adversary node n_j $(i < j)$ cannot remove n_i from the provenance record without being detected.

Justification. Given a packet's path $\{n_1, n_2, \ldots, n_M\}$, the message authentication code received by n_j contains data from the benign node n_i. Since the AM-FM sketch uses a one-way function, an adversary n_j is not able to obtain the provenance from the digest. Thus n_j cannot remove n_i from the provenance without being detected by the AM-FM verification.

Claim 3. A malicious node cannot selectively drop packets generated by benign nodes without being detected.

Justification. According to our packet sequence number generation technique, the sequence number is computed as $E_{k_i}(count_i||rc)$ as discussed in Section 3.2. Initially, $count_i$ and $count'_i$ are both set to 0. When a node n_i generates a new packet p, $count_i$ is incremented by 1 at n_i and when p arrives at the BS, $count'_i$ is also incremented by 1.

Now assume that at some point the values of both $count_i$ and $count'_i$ are equal to K. Node n_i generates a new packet p with sequence number $E_{k_i}(K||rc)$ which results in $count_i$ to increase to $K+1$. Until the packet arrives at the BS, $count'_i$ holds the value K. Assume that the packet is dropped by a malicious node on its way to the BS. In the following round, n_i generates another packet p' and assigns it the sequence number $E_{k_i}((K+1)||rc)$. If p' successfully arrives at the BS, its sequence number does not match with the BS's expected sequence number $E_{k_i}(K||rc)$. Thus the BS detects that the packet p has been dropped by some malicious node. Hence, we can infer that if there is more than one path from a node n_i to the BS and at least one of them is trustable, the dropping of any packet generated or aggregated at node n_i is detectable.

It is important to note that the BS can query the *PPD* of each node on the packet's path to find out the last node that observes the packet with sequence number $E_{k_i}(K||rc)$. In this way, it can locate the malicious node at the next hop.

Claim 4. Packet replay attacks are detectable.

Justification. Assume that the current round count is set as rc and the node n_i generates a packet p with sequence number $E_{k_i}(count_i||rc)$. In the following round, rc is incremented to $rc+1$. Now, if the BS receives the packet p in this round, it detects a replay attack because the round count value in p's sequence number is already expired.

7.2 Security Overhead

Our dictionary based provenance compression technique has some security overheads due to: (1) the generation of the packet sequence number $E_{k_i}(count_i||rc)$ by a *data source node* for each packet generation; (2) the binding of the provenance with its packet through the AM-FM sketch.

Different encryption techniques result in different degrees of security, e.g., AES provides better security than that of DES by incurring higher computational cost. For secure packet sequence number generation, we can use any such encryption technique depending on the choice of having stronger security or lower computational overhead.

Unlike commonly used MACs, e.g., MD5 and SHA-1, the size of the MAC generated by the AM-FM sketch is adjustable. Sensor nodes can set an acceptable verification error rate ε for the generated MACs and a sample size k of the provenance so that the size of such generated MAC never grows beyond a range $[(1-\varepsilon)k, (1+\varepsilon)2k]$ with probability $(1-\delta)$ where $0<\delta<1$, if $k \geq O(\frac{log(2/\delta)}{\varepsilon^2})$ [13]. Here, the verification error refers to false positives and false negatives of MACs' verification. Hence, the larger the MAC size is, the lower the error rates are. Therefore, security overheads incurred in our approach is adjustable as there is a tradeoff between the security and its overheads.

7.3 Performance Analysis

Dictionary based approach is a lossless compression technique. The high compression rate of provenance in our approach attributes to the extra storage space for dictionaries. The space complexity of our provenance scheme is described in the following scenarios:

1) Linear topology. If a packet reuses the previous packets' paths all the way from its source to the BS, the storage complexity of the $pathIndex$ is constant and so is $prIndex$.

Suppose that at some point in the network operations, we have K distinct $pathIndexes$ $\langle n_{b1}, n_{e1}\rangle, \ldots, \langle n_{bK}, n_{eK}\rangle$. Now if a new packet is generated, and traverses a path that can be built entirely from the above $pathIndexes$, the storage complexity of the provenance remains $O(K)$. Note that K does not have direct relationship with the number of nodes on the traversed path. Therefore, the storage complexity increases only with the increase in the number of $pathIndexes$ that are involved in the provenance encoding.

2) Tree topology. Assume that a tree has L branches and each branch contains K path snippets on average. Thus the storage complexity of provenance for such a topology is $O(LK)$. Since the packet size has an upper-bound, the number of packets being aggregated at a time is limited, i.e., the value of L is bounded.

If we use the recursive provenance scheme, the resulting provenance size of the tree topology is only one bit longer than that of the linear topology as it uses a binary variable *flag*. In this approach, the more the path snippets are reused, the better compression rate is achieved.

Compared with existing lightweight provenance schemes [2], [5], our scheme needs extra space for storing dictionaries but can save more energy through high compression rate of provenance. We believe ours is the correct choice because memory chips are today more compact, cheaper, and have low power consumption. On the contrary, the battery is still a bottleneck for sensor nodes.

8 Simulation

We have evaluated the performance of our proposed dictionary based provenance scheme (DP) through simulation for both linear and tree topologies. We have used the TinyOS 2.1.2 TOSSIM simulator [15] and *micaz* as energy model to implement our scheme. In our simulation, we consider a sensor network of 100 stationary nodes with IDs 0 through 99 and vary the network diameter from 2 to 14 hops. The node with ID 0 is assumed to be the BS. Some nodes are randomly selected as *data source nodes* and *aggregator nodes*, and the rest are considered as *forwarder nodes*. The duration of each data collection round is set to 2 s.

We compare the performance of our approach with that of the following three schemes:

Bloom filter based provenance scheme (BFP). This scheme [2], [16] uses a fixed size Bloom filter (BF) to encode the provenance of a packet. It embeds all the nodes on a packet's path in the BF using a set of hash functions.

Generic secure provenance scheme (SPS). For comparison purpose, we adapt the generic secure provenance scheme

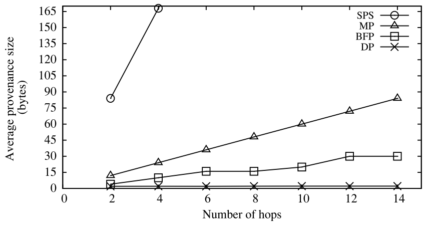


Fig. 4. Average provenance size for different hop counts.

[17] for use in WSN. Here, we simplify the provenance record of a node n_i as $P_i = \langle\ n_i,\ hash(D_i),\ C_i\ \rangle$, where $hash(D_i)$ is a cryptographic hash of the updated data D_i, and C_i contains an integrity checksum computed as $Sign(hash(n_i,\ hash(D_i)\ ||\ C_{i-1}))$. As each node n_i on the packet's path appends its P_i to the received provenance, the size of the provenance increases linearly. To implement *SPS*, we use SHA-1 (160 bit) for cryptographic hash operations and the TinyECC library [18] to generate 160-bit digital signatures. The node ID is 2 bytes in size and thus the increase in the size of the provenance at each hop is 42 bytes.

MAC based provenance scheme (MP). We also consider a MAC-based provenance scheme [2], [16] where a node transmits its node ID and a MAC computed on it as the provenance record. We use TinySec library [19] to compute a 4-byte CBC-MAC. Hence, the provenance size increases by 6 bytes at each hop.

8.1 Performance Metrics

We analyze the performance of our proposed provenance scheme using the following performance metrics:

(a) *Average provenance size*. When the BS receives a packet, it computes the traversed path using the $pathIndex$ and agr fields in the provenance. Since the sizes of n_i and seq are fixed, for comparison purpose, we only consider the sizes of $pathIndex$ ($S_{pathIndex}$) and agr (S_{agr}) to determine the provenance size. Therefore, the provenance size of a packet in our scheme is $PS_{DP} = S_{pathIndex} + S_{agr}$. On the other hand, the size of the provenance for the *BFP* scheme (PS_{BFP}) is determined by the size of the Bloom filter (S_{bf}).

We compute the average provenance size (*APS*) for m packets $p_1, p_2, p_3, \ldots, p_m$ as follows:

$$APS\ =\ \frac{\sum_{i=1}^{m} PS_i}{m},$$

where PS_i denotes the provenance size for packet p_i.

(b) *Verification failure rate*. For each packet, the AM-FM sketch evaluates the consistency between its data and provenance. When the BS receives a packet from a particular *data source node* for the first time, it stores the packet's traversed path into its *PPD*. Later on, when the BS receives more packets from that same *data source node*, it verifies their provenance records using the AM-FM sketch and then compares them with the stored path information. If the provenance information is tampered or the packets' paths are changed, the verification fails at the BS. Now, if the BS receives packets $p_1, p_2, p_3, \ldots, p_m$, the verification failure rate (*VFR*) is defined as the ratio of the number of packets for which verification fails over the total number of packets received,

$$VFR\ =\ \frac{\sum_{i=1}^{m} vf_{p_i}}{m} \times 100\%,$$

where

$$vf_{p_i} = \begin{cases} 1 & \text{if verification fails for } p_i \\ 0 & \text{if verification passes for } p_i \end{cases}$$

and m is the total number of received packets.

(c) *Total energy consumption*. We use PowerTOSSIM Z to measure the amount of energy (in joule) consumed by each sensor node. If there are $n_1, n_2, n_3, \ldots, \ldots n_m$ nodes in the network, the total energy consumption (*TEC*) is computed as follows:

$$TEC\ =\ \sum_{i=1}^{m} EC_{n_i},$$

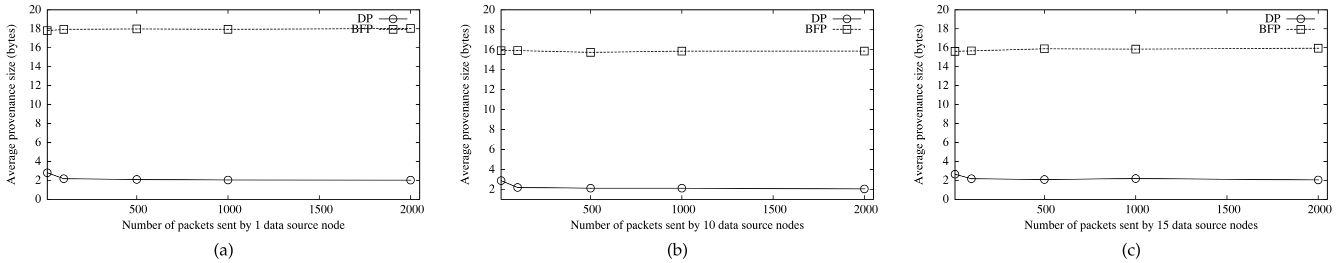
where EC_{n_i} represents the energy consumed by a node n_i and m is the total number of nodes in the network.

(d) *Packet loss detection rate*. It is defined as the ratio of the number of packets detected as lost to the number of packets that are actually lost. If l packet losses are detected out of m actual packet losses, loss detection rate (*LDR*) is computed as $\frac{l}{m}$.

8.2 Simulation Results

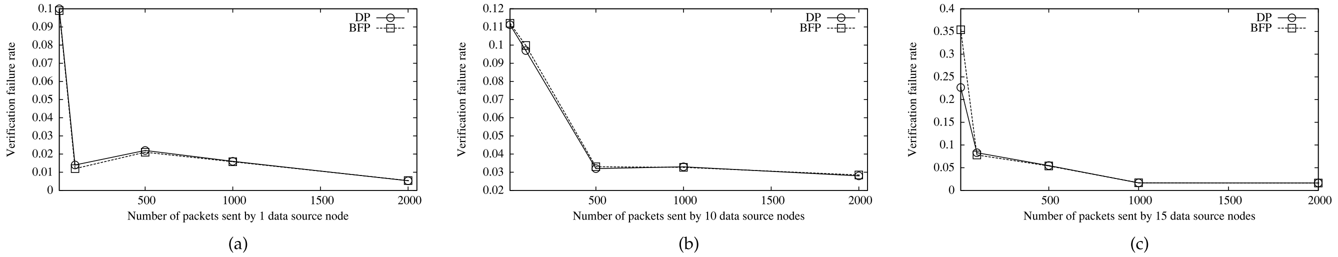
Fig. 4 presents the average provenance size (in bytes) for *DP*, *BFP*, *MP* and *SPS* schemes with respect to the number of hops the packets traverse. In both *SPS* and *MP*, each node appends its encoded provenance to the received one. As a result, the provenance size increases linearly with the path length. However, the *SPS* approach appends 42 bytes at each hop and thus results the provenance size to increase at a much higher rate than that of *MP*. Also in *BFP*, the size of the provenance increases with the number of nodes traversed, but not as fast as in *MP*. As shown in Fig. 4, to correctly decode the provenance of a packet that traverses 12 hops, a 30-byte size Bloom filter is required. The provenance size in our scheme does not increase as the number of hops increases and remains constant at around 2 bytes. Hence, our scheme demonstrates much better performance than *BFP* in reducing the provenance size as the number of hops increases.

We also compare the average provenance size of *DP* and *BFP* ([2], [16]) schemes for three different scenarios in Figs. 5a, 5b, and 5c where there are 1, 10, and 15 *data source nodes* in the network, respectively. In this simulation, we assume that no node aggregates provenance information while the packets move towards the BS. For each scenario, the *data source nodes* send 10, 100, 500, 1,000, and 2,000 packets to the BS. Upon receiving the packets, the BS calculates the average provenance size. Fig. 5 shows that the average provenance size in our scheme decreases with the increase of packet transmissions, and eventually reaches a point where it remains stable. Note that, the higher the number of transmitted packets is, the higher the probability that a transmission path is reused. Therefore, the provenance size decreases correspondingly. However, for the *BFP* scheme, every time a packet is sent, all the nodes along its path are embedded into the Bloom filter [2], [16]. As a result, when a



(a) (b) (c)

Fig. 5. Average provenance size for (a) 1, (b) 10, and (c) 15 *data source nodes.*



(a) (b) (c)

Fig. 6. Verification failure rate for (a) 1, (b) 10, and (c) 15 *data source nodes.*

packet traverses large number of nodes, the size of the Bloom filter, i.e., the size of the provenance increases to keep the error rate minimum. Hence, the provenance size in *BFP* scheme increases with the number of nodes on the packet's path. In Fig. 5, we show that this scheme exhibits larger provenance size than that of our *DP* scheme.

Fig. 6 shows the verification failure rate of *DP* and *BFP* schemes. The causes for verification failure can be attributed to provenance information tampering by malicious nodes, node failure on system crash, or link quality degradation. However in general, the *BFP* scheme has high verification failure rate because of the usage of Bloom filter. During the simulation we consider 200 bytes as the size of the Bloom filter which is large enough and thus causes high communication overhead. To reduce such overhead, we could decrease the size, but doing so increases the error rate as well as the verification failure rate. Fig. 6 manifests that *VFR* for *DP* scheme is initially high and gradually decreases over time as the number of packet transmissions increases. Since we consider a large Bloom filter for *BFP* scheme, the simulation shows (in Fig. 6) similar *VFR* for both provenance schemes.

Fig. 7 illustrates the total energy consumption in *DP*, *BFP*, and *MP* schemes for different hop counts over 100 packet transmissions. We see that the energy consumption in *MP* and *BFP* increases at a much higher rate than that of *DP* scheme. Hence, Fig. 7 shows that our scheme is more energy efficient than other provenance schemes.

Fig. 8 presents the average provenance size of *DP* scheme for three different scenarios of data aggregation-(1) five *data source nodes* and five *aggregator nodes*, (2) 10 *data source nodes* and five *aggregator nodes*, and (3) 10 *data source nodes* and 10 *aggregator nodes*. For all these scenarios, the provenance size decreases as the number of packet transmissions increases. In the first scenario, the probability is high that two or more packets do not meet on the *aggregator nodes*. Therefore, even if we do not use the *recursive provenance scheme*, the average provenance size is close to the linear one's as shown in Fig. 5. However, in the second scenario the probability of aggregation is higher. As a result, the average provenance size increases correspondingly. For such scenario, we can use the *recursive provenance scheme* which causes the provenance size to increase only one bit from that of the dictionary based scheme presented in Section 4.

Fig. 9a shows the detection rate of packet losses in the *DP* scheme. In this simulation, we set the natural link loss rate to 1 percent, and the malicious link loss rate to 3, 6, and 9 percent. We also consider one malicious node in every data path. In Fig. 9a, we see that the higher the link loss rate is, the more the detection rate decreases. With the increase of link loss rate, the probability of consecutive packet drops also increases which our approach cannot detect.

Fig. 9b shows the packet loss rate over time which is used to detect packet drop attacks. In this simulation, we consider a 12-hop network where the second node of each path is set as malicious. The results show that the packet loss rate for a benign network converges to the natural link loss rate. In contrast, with the presence of malicious nodes, the loss

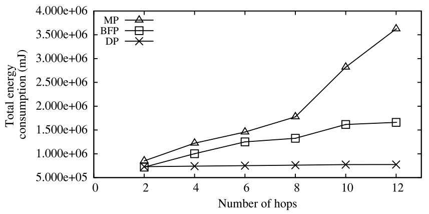


Fig. 7. Total energy consumption for different hop counts.

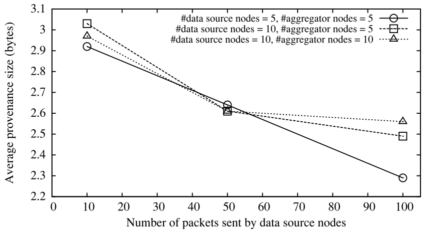
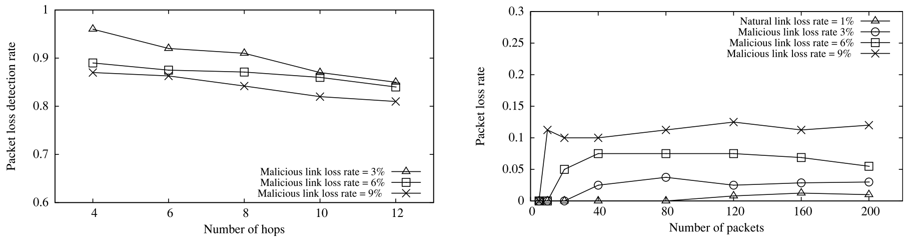


Fig. 8. Average provenance size in aggregation scenario.



(a) (b)

Fig. 9. (a) Packet loss detection rate for various malicious link loss rates, (b) Packet loss rate over time.

rate gradually increases and eventually reaches a point higher than the natural loss rate.

9 Experiments

To further asses our scheme we deployed it in a test-bed containing 10 real sensor motes. In this section, we present its performance evaluation with respect to the metrics mentioned in Section 8.1.

9.1 Experimental Setup

We used the TelosB sensor mote to port the implementation of our approach. TelosB has a 8 MHz TI MSP430 micro-controller, 2.4 GHz radio, 10 kB RAM, and 1 MB external flash for data logging. We placed TelosB motes in an indoor environment (network area of size 10x10 ft^2) and controlled the transmission power of the motes (i.e., set to the lowest power level) to ensure multi-hop communication in the network. All motes were battery powered and a special mote was used as the base station to collect statistical information. The *data source nodes* generated packets in every one second. For the purpose of performance analysis, we collected provenance information by running the experiments for 10, 100, 500, 1,000, and 1,500 packet transmissions. We connected the BS to a laptop through USB port in order to collect the statistical data from the network.

9.2 Experimental Results

Fig. 10a presents the average provenance size resulting in the testbed experiments of our scheme. In this experiment we consider only one *data source node* that sends packets to the BS at every second. This result reflects the trends observed in the simulation results as shown in Fig. 5a. It also confirms the claim that the average provenance size decreases with the number of packet transmissions and eventually reaches an optimal value where it remains stable.

Concerning the verification failure rate, the experimental result in Fig. 10b shows similar trend to the result in Fig. 6a. At the initial stage of testbed experiments, the verification failure rate is higher than at any other time. Due to initial network instability, data-flow paths change frequently at that time. However, as the network becomes stable over time, more stable routes are established. As a result, *VFR* decreases exponentially to almost 0.

Fig. 10c shows the average provenance size for an aggregation scenario where a particular node aggregates packets from two other nodes and forwards this aggregated packet to the next node. In this experiment, the provenance size also decreases with the number of packet transmissions and remains stable at size 2 bytes.

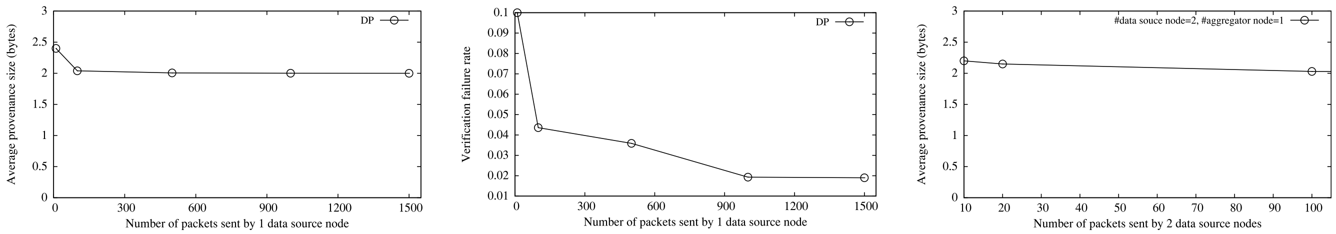
Hence, the experimental results show trends similar to the simulation results and generate almost the same values for different performance metrics.

10 Related Works

Extensive research has been carried out on the topic of network provenance. However, due to limited computational ability, energy constraints, and low bandwidth, these conventional provenance schemes [20], [21] cannot be applied in WSNs directly.

Alam and Fahmy propose a probabilistic approach [5] to encode the nodes' IDs into the provenance. Shebaro et al. use Bloom filter [6] to encode the IDs of the nodes that are on a packet's path. These approaches minimize the size of provenance information by keeping only the nodes' IDs. However, the edges which refer the packet transmissions are discarded. Hence, those approaches are lossy provenance compression techniques. Salmin et al. [2], [16] propose a lightweight secure provenance scheme based on in-packet Bloom filter. This approach binds data and its provenance together and also chains the packet sequence numbers adjacently to detect provenance forgery and packet dropping attacks.

Lossy provenance encoding schemes [2], [5], [6], [16] share the following characteristics: (i) only nodes' IDs are recorded in the data provenance, (ii) unavoidable false positive in provenance decoding, and (iii) increase in the



(a) (b) (c)

Fig. 10. Experimental (a) average provenance size, (b) verification failure rate, and (c) average provenance size in aggregation scenario.

provenance size with the number of nodes traversed in order to keep the false positive rate under a given threshold.

The compression technique used in this paper is based on the compression algorithms proposed by Ziv and Lempel [11], [12]. To build the dictionary they use the substrings that appear multiple times in a message. As cyclic paths are not allowed in sensor networks, a node shows up at most once on a packet's path i.e., there is no repeated nodes on a path. As a result, these traditional dictionary based algorithms cannot be applied directly to provenance compression.

Chen and Reif [22] reduce the subtrees recursively into their roots. So, the resulting graph represents a coarse granular view of the provenance. By recursively shrinking children nodes into their *aggregator nodes*, the provenance tree can be represented as a set of linear path snippets. Then each of these path snippets is compressed using *PPD*s. This work is close to our approach. However, a significant difference between these two approaches is that our approach can recover the complete provenance tree from the coarse granular view by looking up the $pathIndexes$ in their *PPD*s whereas the approach by Chen and Reif cannot.

Hasan et al. [17] use the provenance chain model and MAC to ensure integrity of the provenance. Compared with our security solutions, this approach requires to transmit a large amount of provenance information as it is not specifically designed for WSNs.

11 Conclusions

In this paper, we propose a dictionary based secure provenance scheme for wireless sensor networks. Using packet path dictionaries, we enclose path indexes instead of the path itself in the provenance. Hence, the size of the compressed provenance in our lossless approach is smaller than that of the existing lossy provenance schemes. By using the AM-FM sketch scheme and a secure packet sequence number generation technique, we ensure the security objectives of our scheme. Simulation and experimental results show that our scheme can save more energy and bandwidth than other existing provenance schemes.

Acknowledgments

The work reported in this paper has been partially supported by the Purdue Cyber Center, by the National Science Foundation under grant CNS-1111512, and by the Bajian Project for Selected Researchers in Jiangsu University under contract number 1213000013. At the time this work was done, C. Wang was with the Department of Computer Science, Purdue University. C. Wang and S. R. Hussain contributed equally to this work. S. R. Hussain is the corresponding author.

References

[1] H.-S. Lim, Y.-S. Moon, and E. Bertino, "Provenance-based trustworthiness assessment in sensor networks," in *Proc. 7th Int. Workshop Data Manage. Sensor Netw.*, 2010, pp. 2–7.

[2] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab, "A lightweight secure provenance scheme for wireless sensor networks," in *Proc. IEEE 18th Int. Conf. Parallel Distrib. Syst.*, 2012, pp. 101–108.

[3] I. Foster, J. Vockler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," in *Proc. 14th Int. Conf. Sci. Statist. Database Manage.*, 2002, pp. 37–46.

[4] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-aware storage systems," in *Proc. USENIX Annu. Tech. Conf., General Track*, 2006, pp. 43–56.

[5] S. M. I. Alam and S. Fahmy, "Energy-efficient provenance transmission in large-scale wireless sensor networks," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2011, pp. 1–6.

[6] B. Shebaro, S. Sultana, S. R. Gopavaram, and E. Bertino, "Demonstrating a lightweight data provenance for sensor networks," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 1022–1024.

[7] A.-H. Jallad and T. Vladimirova, "Data-centricity in wireless sensor networks," in *Guide to Wireless Sensor Networks*(ser. Computer Communications and Networks). London, U.K.: Springer, 2009, pp. 183–204.

[8] D. Ma, "Secure feedback service in wireless sensor networks," in *Information Security Practice and Experience*. ser. Lecture Notes in Computer Science, vol. 4464, Springer, 2007, pp. 116–128.

[9] S. Sultana, E. Bertino, and M. Shehab, "A provenance based mechanism to identify malicious packet dropping adversaries in sensor networks," in *Proc. 31st Int. Conf. Distrib. Comput. Syst. Workshops*, 2011, pp. 332–338.

[10] S. Ihara, *Information Theory for Continuous Systems*. Singapore: World Scientific, 1993.

[11] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. 23, no. 3, pp. 337–343, May 1977.

[12] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978.

[13] M. Garofalakis, J. M. Hellerstein, P. Maniatis, and IEEE, "Proof sketches: Verifiable in-network aggregation," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, 2007, pp. 971–980.

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, Dec. 2002.

[15] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proc. 1st Int. Conf. Embedded Netw. Sens. Syst.*, 2003, pp. 126–137.

[16] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab, "A lightweight secure scheme for detecting provenance forgery and packet drop attacks in wireless sensor networks," *IEEE Trans. Dependable Secure Comput.*, vol. PP, no. 99, p. 1, 2013.

[17] R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: Preventing history forgery with secure provenance," in *Proc. 7th Conf. File Storage Technol.*, 2009, pp. 1–14.

[18] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. Int. Conf. Inform. Process. Sens. Netw*, Apr. 2008, pp. 245–256.

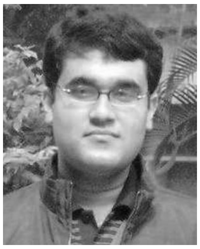
[19] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sens. Syst.*, 2004, pp. 162–175.

[20] W. Zhou, M. Sherr, T. Tao, X. Li, B. T. Loo, and Y. Mao, "Efficient querying and maintenance of network provenance at internet-scale," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2010, pp. 615–626.

[21] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr, "Secure network provenance," in *Proc. 23rd ACM Symp. Oper. Syst. Principles*, 2011, pp. 295–310.

[22] S. Chen and J. H. Reif, "Efficient lossless compression of trees and graphs," in *Proc. IEEE Data Compression Conf. (DCC'96)*, Mar. 1996, p. 428.

Changda Wang is a professor in the School of Computer Science and Communication Engineering, Jiangsu University. His main research interests include security, network communication, and cloud computing. He is the recipient of Qinglan and Liuda Gaofeng awards of Jiangsu Province. He is a member of CCF and serves in the Network & Data Communication Committee.



Syed Rafiul Hussain is working towards the PhD degree in computer science at Purdue University. His research interests include data provenance, network security, and operating systems security. He is a member of the Center for Education and Research in Information Assurance and Security (CERIAS). He is a member of the IEEE.

Elisa Bertino is a professor of computer science at Purdue University, and serves as a research director of the Center for Education and Research in Information Assurance and Security (CERIAS) and a director of Cyber Center (Discovery Park). Previously, she was a faculty member and department head at the Department of Computer Science and Communication of the University of Milan. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She is currently serving as EiC of the *IEEE Transactions on Dependable and Secure Computing*. She also served as an editor in chief of the *VLDB Journal*. She co-authored the book *Identity Management-Concepts, Technologies, and 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.