A Lightweight Secure Provenance Scheme for Wireless Sensor Networks

Salmin Sultana #1, Gabriel Ghinita †2, Elisa Bertino #3, Mohamed Shehab ‡4

# *Purdue University,* † *University of Massachusetts,* ‡ *University of North Carolina*

1,3 {ssultana, bertino}@purdue.edu, 2 gabriel.ghinita@umb.edu, 4 mshehab@uncc.edu

***Abstract*—Large-scale sensor networks are being deployed in numerous application domains, and often the data they collect are used in decision-making for critical infrastructures. Data are streamed from multiple sources through intermediate processing nodes that aggregate information. A malicious adversary may tamper with the data by introducing additional nodes in the network, or by compromising existing ones. Therefore, assuring high data trustworthiness in such a context is crucial for correct decision-making. Data provenance represents a key factor in evaluating the trustworthiness of sensor data. Provenance management for sensor networks introduces several challenging requirements, such as low energy and bandwidth consumption, efficient storage and secure transmission. In this paper, we propose a novel light-weight scheme to securely transmit provenance for sensor data. The proposed technique relies on *in-packet Bloom filters* to encode provenance. In addition, we introduce efficient mechanisms for provenance verification and reconstruction at the base station. We evaluate the proposed technique both analytically and empirically, and the results prove its effectiveness and efficiency for secure provenance encoding and decoding.**

***Keywords*-Provenance, Security, Sensor Networks**

I. Introduction

Sensor networks are becoming increasingly popular in numerous application domains, such as cyberphysical infrastructure systems, environmental monitoring, power grids, etc. Data are produced at a large number of sensor node sources and processed in-network at intermediate hops on their way to a base station that performs decision-making. The diversity of data sources creates the need to assure the trustworthiness of data, such that only trustworthy information is considered in the decision process. Data provenance is an effective method to assess data trustworthiness, since it summarizes the history of ownership and the actions performed on the data. Recent research [1] highlighted the key contribution of provenance in systems where the use of untrustworthy data may lead to catastrophic failures (e.g., SCADA systems for critical infrastructure). Although provenance modeling, collection, and querying have been investigated extensively for workflows and curated databases [2], [3], provenance in sensor networks has not been properly addressed. In this paper, we investigate the problem of secure and efficient provenance transmission and processing for sensor networks.

In a multi-hop sensor network, *data provenance* allows the base station to trace the source and forwarding path of an individual data packet since its generation. Provenance must be recorded for each data packet, but important challenges arise due to the tight storage, energy and bandwidth constraints of the sensor nodes. Therefore, it is necessary to devise a light-weight provenance solution which does not introduce significant overhead. Furthermore, sensors often operate in an untrusted environment, where they may be subject to attacks. Hence, it is necessary to address security requirements such as *confidentiality*, *integrity* and *freshness* of provenance. Our goal is to design a *provenance encoding and decoding mechanism* that satisfies such security and performance needs. We propose a provenance encoding strategy whereby each node on the path of a data packet securely embeds provenance information within a *Bloom filter*, that is transmitted along with the data. Upon receiving the data, the base station extracts and verifies the provenance.

As opposed to existing research that employs separate transmission channels for data and provenance [4], we only require a single channel for both. Such an approach is more practical. Furthermore, traditional provenance security solutions use intensively cryptography and digital signatures [5], and they employ append-based data structures to store provenance, leading to prohibitive costs. In contrast, we use *Bloom filters (BF)*, which are fixed-size data structures that compactly represent provenance. BFs makes efficient usage of bandwidth, and even if they only provide probabilistic decoding guarantees, they yield very low error rates in practice. Our specific contributions are:

- We formulate the problem of secure provenance transmission in sensor networks, and identify the challenges specific to this context;
- We propose an in-packet BF provenance-encoding scheme;
- We design efficient techniques for provenance decoding and verification at the base station;
- We perform a detailed security analysis and performance evaluation of the proposed technique.

The rest of the paper is organized as follows: Section II sets the problem background and describes the system, threat and security models. Section III introduces the provenance encoding scheme. Section IV presents the security analysis. Section V provides a theoretical performance analysis, whereas Section VI presents the experimental evaluation results for the proposed scheme. We survey related work in Section VII and conclude in Section VIII.

II. Background and System Model

In this section, we introduce the network, data and provenance models used. We also present the threat model and security requirements. Finally, we provide a brief primer on Bloom filters, their fundamental properties and operations.

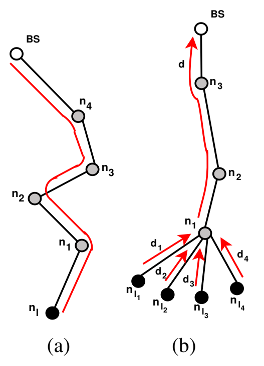


Figure 1. Provenance graph for a sensor network.

Network Model. We consider a multihop wireless sensor network, consisting of a number of sensor nodes and a base station (BS) that collects data from the network. The network is modeled as a graph $G(N, L)$, where $N = \{n_i|, 1 \leq i \leq |N|\}$ is the set of nodes, and L is the set of links, containing an element $l_{i,j}$ for each pair of nodes n_i and n_j that are communicating directly with each other. Sensor nodes are stationary after deployment, but routing paths may change over time, e.g., due to node failure. Each node reports its neighboring (i.e. one hop) node information to the BS after deployment. The BS assigns each node a unique identifier $nodeID$ and a symmetric cryptographic key K_i. In addition, a set of hash functions $H = \{h_1, h_2, ..., h_k\}$ are broadcast to the nodes to use during provenance embedding.

Data Model. We assume a multiple-round process of data collection. Each sensor node generates data periodically, and individual values are routed and aggregated towards the BS using any existing hierarchical (i.e., tree-based) dissemination scheme, e.g., [6]. A data path of p hops is represented as $< n_l, n_1, n_2, ..., n_p >$, where n_l is a leaf node representing the data source, and node n_i is i hops away from n_l. Each non-leaf node in the path aggregates the received data and provenance with its own locally-generated data and provenance.

Each data packet contains (i) a unique packet sequence number (ii) a data value, and (iii) provenance. The sequence number is attached to the packet by the data source, and all nodes use the same sequence number for a given round [7]. The sequence number integrity is ensured through message authentication codes (MAC), as discussed in Section III.

Provenance Model. We consider *node-level* provenance, which encodes the nodes that are involved at each step of data processing. This representation has been used in previous research for trust management [1] and for detecting selective forwarding attacks [8].

Given a data packet d, its provenance is modeled as a directed acyclic graph $G(V, E)$ where each vertex $v \in V$ is attributed to a specific node $HOST(v) = n$ and represents the provenance record (i.e. nodeID) for that node. Each vertex in the provenance graph is uniquely identified by a vertex ID (VID) which is generated by the host node using cryptographic hash functions. The edge set E consists of directed edges that connect sensor nodes.

Definition 1 (Provenance): Given a data packet d, the provenance p_d is a directed acyclic graph G(V,E) satisfying the following properties: (1) p_d is a subgraph of the sensor network $G(N, L)$; (2) for $v_i, v_j \in$ V, v_i is a child of v_j if and only if HOST (v_i) = n_i participated in the distributed calculation of d and/or forwarded the data to HOST (v_j) =n_j; (3) for a set U = $\{v_i\} \subset$ V and $v_j \in$ V, U is a set of children of v_j if and only if HOST (v_j) collects processed/forwarded data from each HOST($v_i \in$ U) to generate the aggregated result.

Figure 1 shows two provenance examples in sensor networks. In Figure 1(a), the leaf node n_l generates a data packet d and each intermediate node aggregates its own sensory data with d then forwards it towards the BS. Hence, the provenance corresponding to d is $< v_l, v_1, v_2, v_3 >$, which can be represented as a simple path. In Figure 1(b), the internal node n_1 generates the data d by aggregating data d_1 , ..., d_4 from n_{l_1} , ..., n_{l_4} and then passes d towards the BS. Here, n_1 is an aggregator and the aggregated provenance $< \{v_{l_1}, v_{l_2}, v_{l_3}, v_{l_4}\}, v_1, v_2, v_3 >$ is represented as a tree.

Threat Model and Security Objectives. We assume that the BS is trusted, but any other arbitrary node may be malicious. An adversary can eavesdrop and perform traffic analysis anywhere on the path. In addition, the adversary is able to deploy a few malicious nodes, as well as compromise a few legitimate nodes by capturing them and physically overwriting their memory. These malicious nodes might collude to attack the system. If an adversary compromises a node, it can extract all key materials, data, and codes stored on that node. The adversary may drop, inject or alter packets on the links that are under its control. We do not consider denial of service attacks such as the complete removal of provenance, since a data packet with no provenance records will make the data highly suspicious [5] and hence generate an alarm at the BS. Instead, the primary concern is that an attacker attempts to misrepresent the data provenance. Our objective is to achieve the following security properties:

- *Confidentiality*: An adversary cannot gain any knowledge about data provenance by analyzing the contents of a packet. Only authorized parties (e.g., the BS) can process and check the integrity of provenance.
- *Integrity*: An adversary, acting alone or colluding with others, cannot add or remove non-colluding nodes from the provenance of benign data (i.e. data generated by benign nodes) without being detected.
- *Freshness*: An adversary cannot replay captured data and provenance without being detected by the BS.

Bloom Filters (BF). A Bloom filter is a space-efficient data structure for probabilistic representation of a set of items $S = \{s_1, s_2, ..., s_n\}$ using an array of m bits with k independent hash functions $h_1, h_2, ..., h_k$. The output of each hash function h_i maps an item s uniformly to the

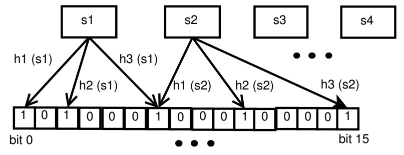


Figure 2. A Bloom filter with $n = 4$, $m = 16$ and $k = 3$.

range [0, m-1] and is interpreted as an index pointing to a bit in a m-bit array. Hence, the BF can be represented as $\{b_0, \ldots, b_{m-1}\}$. Initially each of the m bits is set to 0.

To insert an element $s \in S$ into a BF, s is hashed with all the k hash functions producing the values $h_i(s)(1 \leq i \leq k)$. The bits corresponding to these values are then set to 1 in the bit array. Figure 2 illustrates an example of BF insertion.To query the membership of an item s' within S, the bits at indices $h_i(s')(1 \leq i \leq k)$ are checked. If any of them is 0, then certainly $s' \notin S$. Otherwise, if all of the bits are set to 1, $s' \in S$ with high probability. There exists a possibility of error which arises due to *hashing collision* that makes the elements in S collectively causing indices $h_i(s')$ being set to 1 even if $s' \notin S$. This is called a *false positive*. Note that, there is no *false negative* in the BF membership verification.

The cumulative nature of BF construction inherently supports the *aggregation* of BFs of a same kind, by performing bitwise-OR between the bitmaps.

III. Secure Provenance Scheme

We propose a distributed mechanism to encode provenance in a data packet and a centralized algorithm at the BS to decode the provenance. The technical core of our proposal is an *in-packet Bloom filter* (iBF) [9]. Each packet consists of a unique sequence number, data value, and an iBF which holds the provenance. We focus on transmitting provenance graph vertices over an iBF.

We emphasize that our focus is on securely transmitting provenance to the BS. In an aggregation infrastructure, securing the data values is also an important aspect, but that has been already addressed in previous work (e.g., [10]). Our secure provenance technique can be used in conjunction with existing work to obtain a complete solution that provides security for data, provenance and data-provenance binding.

A. Provenance Encoding

For a data packet, *provenance encoding* refers to generating the vertices in the provenance graph and inserting them into the iBF. Each vertex originates at a node in the data path and represents the provenance record of the host node. A vertex is uniquely identified by the vertex ID (VID). The VID is generated per-packet based on the packet sequence number (seq) and the secret key K_i of the host node. We use a *block cipher function* to produce this ID in a secure manner. Thus for a given data packet, the VID of a vertex representing the node n_i is computed as

$$vid_i = generateVID(n_i, seq) = E_{K_i}(seq) \qquad (1)$$

where E is a secure block cipher such as AES, etc.

Whenever a source node generates a data packet, it also creates a BF (referred to as ibf_0), initialized to all 0's. The source then generates a vertex according to Eq. 1, inserts the VID into ibf_0 and transmits the BF as a part of the packet.

Upon receiving the packet, each intermediate node n_j performs data as well as provenance aggregation. If n_j receives data from a single child n_{j-1}, it aggregates the partial provenance contained in the packet with its own provenance record. In this case, the iBF ibf_{j-1} belonging to the received packet represents a partial provenance i.e. the provenance graph of the sub-path from the source upto n_{j-1}. On the other hand, if n_j has more than one child, it generates an aggregated provenance from its own provenance record and the partial provenance received from its child nodes. At first, n_j computes a BF $ibf_{(j-1)}$ by bitwise-ORing the iBFs received from the children. ibf_{j-1} represents a partial but aggregated provenance from all of the child nodes. In either case, the ultimate aggregated provenance is generated by encoding the provenance record of n_j into $ibf_{(j-1)}$. To this end, n_j creates a vertex using Eq. 1, inserts the VID into $ibf_{(j-1)}$ which is then referred to as ibf_j.

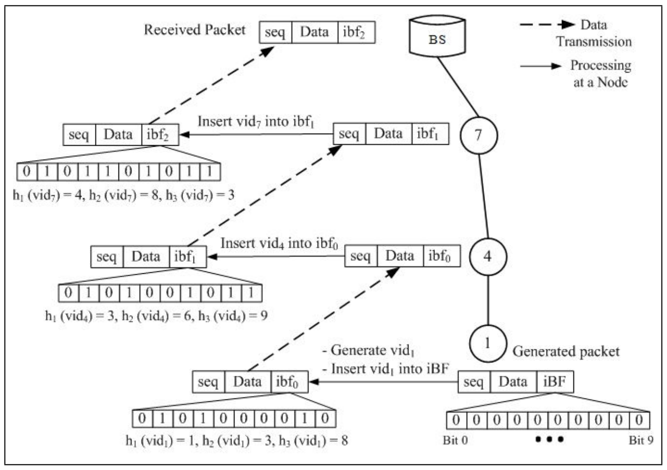
When the packet reaches the BS, the iBF contains the provenance records of all the nodes in the path i.e. the full provenance. We denote this final record by ibf.

Example: We illustrate the encoding mechanism by using the example network in Fig. 3(a). The data path considered is $< 1, 4, 7 >$, where node 1 is the data source. We use a 10-bit BF and a set of 3 hash functions $H = \{h_1, h_2, h_3\}$ for BF operations. When node 1 generates a data packet with sequence number seq, it creates the BF ibf_0 which is set to all 0's. The node then creates a vertex corresponding to its provenance record and computes the VID as $vid_1 = E_{K_1}(seq)$. To insert vid_1 into ibf_0, node 1 generates three indices as $h_1(vid_1) = 1$, $h_2(vid_1) = 3$, $h_3(vid_1) = 8$. The VID is then inserted by setting ibf_0[1], ibf_0[3], and ibf_0[8] to 1. The updated ibf_0 along with the packet is then sent towards the BS.

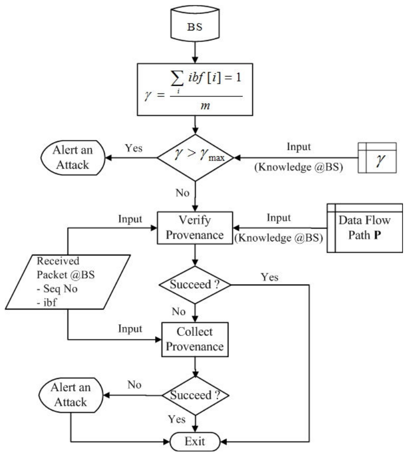
Upon receiving the packet, node 4 performs data and provenance aggregation. Since the node has one child, it only aggregates its own provenance record with ibf_0. For this purpose, the node generates a VID vid_4; computes 3 indices as $h_1(vid_4) = 3$, $h_2(vid_4) = 6$, $h_3(vid_4) = 9$; and inserts vid_4 into ibf_0 by setting bits 3, 6, 9 of the iBF to 1. This updated iBF is referred to as ibf_1. The data packet with ibf_1 is then forwarded to node 7 which repeats the provenance aggregation steps. At the end, the BS receives the packet with the final iBF (ibf_2 from node 7) and stores this iBF for further processing.

B. Provenance Decoding

When the BS receives a data packet, it executes the *provenance verification* process, which assumes that the BS knows what the data path should be, and checks the iBF to



(a)



(b)

Figure 3. (a) Mechanism for encoding provenance (node 1 is data source). (b) Provenance processing workflow at the BS upon receiving a packet.

see whether the correct path has been followed. However, right after network deployment, as well as when the topology changes (e.g., due to node failure), the path of a packet sent by a source may not be known to the BS. In this case, the provenance collection process is executed, which retrieves provenance from the received iBF and thus the BS learns the data path from a source node. Afterwards, upon receiving a packet, it is sufficient for the BS to verify its knowledge of provenance with that in the packet. Below we discuss these two processes in more details:

Algorithm 1 ProvenanceVerification

```
Input: Received packet with sequence seq and iBF ibf.
Set of hash functions $H$, Data path $P' = < n'_{l_1}, ..., n'_1, ..., n'_p >$

$BF_c \leftarrow 0$   // Initialize Bloom Filter
for each $n'_i \in P$ do
    $vid'_i$ = generateVID ($n'_i, SeqNo$)
    insert $vid'_i$ into $BF_c$ using hash functions in $H$
endfor
if ($BF_c = ibf$) then
    return true   // Provenance is verified
endif
return  false
```

Provenance Verification: The BS conducts the verification process not only to verify its knowledge of provenance but also to check the integrity of the transmitted provenance. Algorithm 1 shows the steps to verify provenance for a given packet. We assume that the knowledge of the BS about this packets path is P. At first, the BS initializes a Bloom filter BF_c with all 0s. The BF is then updated by generating the VID for each node in the path P and inserting this ID into the BF. BF_c now reflects the perception of BS about the encoded provenance. To validate its perception, the BS then compares BF_c to the received iBF ibf. The provenance verification succeeds only if BF_c is equal to ibf . Otherwise, if BF_c differs from the received iBF, it indicates either a change in the data flow path or a BF modification attack. The verification failure triggers the provenance collection process which attempts to retrieve the nodes from the encoded provenance and also to distinguish between the events of a path change and an attack.

Provenance Collection: As illustrated in Algorithm 2, the *provenance collection* scheme makes a list of potential vertices in the provenance graph through the ibf membership testing over all the nodes. For each node n_i in the network, the BS creates the corresponding vertex (i.e. v_i with VID vid_i) using Eq. 1. The BS then performs the membership query of vid_i within ibf. If the algorithm returns true, the vertex is very likely present in provenance, i.e., the host node n_i in the data path. Such an inference might introduce errors because of false positives (a node not on the route is inferred to be on the route). However, as shown in Section V, the false positive probability obtained is very low.

Once the BS finalizes the set of potential candidate nodes S = $< n'_{l_1}, ..., n'_1, n'_2, ..., n'_p >$, it executes the provenance verification algorithm on this set. This step is required to distinguish between the cases of a legitimate route change and that of malicious activity. If the verification succeeds, we decide that there was a natural change in the data path and we have been able to determine the path correctly. Otherwise, an attack has occurred.

A possible attack is the *all-one* attack where all bits in the provenance are set to 1, which implies the presence of all nodes in the provenance. To address the issue, we use a *density metric* γ introduced in [11]. γ reflects the number of 1's in the provenance (i.e. the iBF) as a fraction of the total size. To consider the provenance valid, we require that the density is equal or below a certain threshold: $\gamma \leq \gamma_{max}$. Such a requirement is reasonable since in a BF with n elements and k hash functions, there may be at most kn

bits marked as '1'. Hence, we can always find an upper bound for the number of 1's in a BF. Thus, the maximum number of allowable 1's is $m\gamma_{max}$.

Algorithm 2 ProvenanceCollection

```
Input: Received packet with sequence seq and iBF ibf.
Set of nodes ($N$) in the network, Set of hash functions $H$
1. Initialize
    Set of Possible Nodes $S \leftarrow <>$
    Bloom Filter $BF_c \leftarrow 0$   // To represent S
2. Determine possible nodes in the path and build the representative BF
    for each node $n_i \in N$ do
      $vid_i$ = generateVID ($n_i, seq$)
      if ($vid_i$ is in $ibf$) then
        $S \leftarrow S \cup n_i$
        insert $vid_i$ into $BF_c$ using hash functions in $H$
      endif
    endfor
3. Verify $BF_c$ with the received iBF
    if ($BF_c = ibf$) then
      return $S$   // Provenance has been determined correctly
    else
      return NULL   // Indicates an in-transit attack
    endif
```

Within this bound, an attacker may also *randomly flip some bits* to add or delete a legitimate node. The chance of being successful in this attack is very small since the attacker has to identify k bit positions corresponding to the node, which again change for each packet. If each bit is guessed randomly, the probability that the attacker guesses all of them correctly is given by $\frac{1}{2^m}$. Moreover, an attempt of blindly altering some bits is detected since the verification process at the end of the *provenance collection* phase does not succeed. A successful attack occurs when the bits set by the attacker (limited by γ_{max}) make all the k bits corresponding to a legitimate node turn out to be '1'. If the data provenance includes n nodes, the kn hash results may map to at least one and at most $m\gamma_{max}$ bits. Thus a smart attacker marks upto $(m\gamma_{max} - 1)$ bits. The total number of bit patterns by $(m\gamma_{max} - 1)$ hash computations is

$$B = \sum_{i=1}^{(m\gamma_{max}-1)} \binom{m}{i}$$

Randomly guessing one of them has $\frac{1}{B}$ chance of success. Hence, the success in manipulation attack has a very small probability. The workflow shown in Fig. 3(b) summarizes the provenance decoding process.

IV. Security Discussion

Confidentiality. Provenance is encoded using BF hashing functions, and the hashed value takes into account the secret key K_i of each node as part of the vertex VIDs (Eq. 1), as well as a unique sequence number. Hence, even if an attacker collects a large sample of iBFs, it cannot perform a dictionary attack without knowing the node secret key.

Integrity. First, an attacker cannot add legitimate nodes to the provenance of data generated by the compromised nodes. Assume the attacker attempts to frame some uncompromised nodes $< n_l, n_1, n_2, \ldots, n_p >$ to make them responsible for false data. Provenance embedding requires the node secret key K_i to compute the VID_i, which the attacker does not have. Hence, the attack is not successful.

Second, an attacker cannot selectively add or remove nodes from the provenance of data generated by uncompromised nodes. Assume that nodes n_e and n_m collude to execute an attack. A benign packet with provenance $< n_l, ..., n_1, n_2, ..., n_p >$ is routed through n_e, and n_e attempts to remove n_2 from the provenance and to replace it with another legitimate node n_2. When the packet reaches n_e , it contains the partial provenance $< n_l, ..., n_1, ..., n_e >$ encoded in the iBF ibf_{pp}. To remove n_2 from provenance, at first n_e has to construct the Bloom filter BF_2 containing the provenance record of n_2. The bitwise-AND of the negated value of BF_2 with ibf_{pp} removes the information of n_2 from the provenance. Assume the modified iBF is ibf_{pp}. To add n_2' to the provenance after the removal of n_2, the BF corresponding to n_2 should be built and then OR-ed with ibf_{pp}. In both cases, the attackers are unable to construct a BF representing uncompromised nodes, due to the absence of the secret keys of legitimate nodes.

Freshness. Provenance replay attacks are detected by our proposed scheme, since provenance is derived using a unique packet sequence number and the secret key of the node. An attempt to change the sequence number of a packet without having the key will be detected at the BS, according to the integrity property discussed above.

V. Performance Analysis

We present an analysis of the space and energy overhead of the proposed scheme. To the best of our knowledge, no secure provenance scheme has been proposed for sensor networks. Hence, we use the following two benchmarks:

(i) We adapt the generic secure provenance framework $SProv$ [5] to sensor networks. In this lightweight version of the scheme, referred to as SSP, we simplify the provenance record at a node n_i as $P_i =< n_i, hash(D_i), C_i >$, where $hash(D_i)$ is a cryptographic hash of the updated data, and C_i contains an integrity checksum as $Sign(hash(n_i, hash(D_i)|C_{i-1}))$.

(ii) We also consider a MAC-based provenance scheme, referred to as *MP*, where a node transmits the nodeID and a MAC computed on it as the provenance record.

Space Complexity. To implement SSP, we use SHA-1 (160 bit) for cryptographic hash operations and the TinyECC library [12] to generate 160-bit digital signatures (ECDSA). The nodeID has length 2 bytes, thus the length of each provenance record is 42 bytes. For MP, we use TinySec library [13] to compute a 4-byte CBC-MAC. Hence, a provenance record has 6 bytes in this case. As each node in the path encodes its own provenance record, the provenance size increases linearly with the number of hops. For a D-hop path, the provenance is $42D$ bytes in SSP and $6D$ bytes in MP.

Since our approach is based on BF, the provenance length depends on parameter selections for the BF. The false positive probability for a BF is defined as [14]

$$P_{fp} = \frac{n_a - n}{n_t - n}$$

where n_t is the total number of distinct elements in the element space, n is the number of elements actually encoded in the BF and n_a is the number of elements retrieved by querying the BF. Let m be the BF size, k the number of hash functions and D the maximum number of nodes in any path. The false positive probability is equal to that of getting 1 in all the k array positions computed by the hash functions while querying the membership of an element that was not inserted in the BF. This probability is

$$P_{fp} = (1 - (1 - \frac{1}{m})^{kD})^k \approx (1 - e^{-\frac{kD}{m}})^k \qquad (2)$$

For a given m and D, the number of hash functions that minimizes the false positives can be computed as

$$k_{opt} = \frac{m}{D} ln2 \qquad (3)$$

Given D and a desired false positive probability P_{fp}, the required number of bits m can be computed by substituting the optimal value of k in Eq. (2) and then simplifying it to

$$ln(P_{fp}) = -\frac{m}{D} * (ln2)^2 \ \Rightarrow m = \frac{-D * ln(P_{fp})}{(ln2)^2}$$

This means that in order to maintain a fixed false positive probability, the length of a BF should grow with the number of elements to be inserted. If we consider $P_{fp} = 0.02$ and a 14-hop path, the BF size m is computed as 114 bits and k_{opt} = 6. Thus, a 120-bit (15 byte) BF is sufficient to encode provenance while maintaining low false positives. In practice, we bound P_{fp} by a small constant $\delta\,(> 0)$ such that $P_{fp} < \delta$. To find the appropriate value of m we have

$$ln(P_{fp}) > ln\delta \ \Rightarrow -\frac{m}{D} * (ln2)^2 > ln\delta \ \Rightarrow m < \frac{Dln\frac{1}{\delta}}{(ln2)^2}$$

Energy Consumption. For a D-hop path, SSP has to transmit $42 * D$ bytes (= $336 * D$ bits), MP transmits $6 * D$ bytes (= $48 * D$ bits) whereas our scheme requires m bits transmitted. SSP, MP and our scheme consume a radio energy proportional to $(336 * D)$, $(48 * D)$ and $\frac{ln\frac{1}{\delta}}{(ln2)^2} * D$, respectively. Although all of the terms are proportional to D, the constant coefficient in the first two terms are much larger than the last one. For example, if we set $\delta = 10^{-4}$ then the coefficient in our scheme is 19.17 which is much smaller than the coefficients in SSP and MP. Another part of overhead comes from the signature, MAC and hash computations. However, in sensor networks, usually computation overhead is much smaller than that of communication and adds only marginal energy consumption [15].

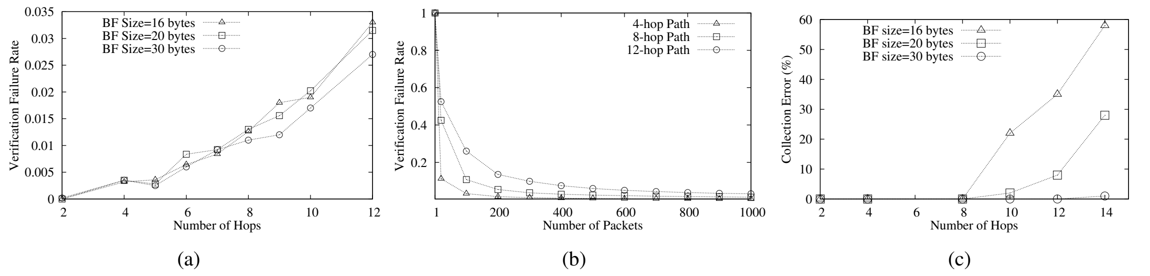
VI. Simulation Results

We implemented and tested the proposed technique using the TinyOS simulator (TOSSIM) [16], and we have used the *micaz* energy model. We consider a network of 100 nodes and vary the network diameter from 2 to 14. All results are averaged over 100 runs with different random seeds.

a) Provenance Decoding Error: The provenance decoding process retrieves the provenance from the in-packet Bloom filter, and consists of the *verification* and *collection* phases. To quantify the accuracy and efficiency of our provenance scheme, we measure decoding error in both the above phases, i.e., *verification* and *collection error*.

Algorithm 1 shows that the verification fails when the provenance graph in the packet does not match with the local knowledge at the BS. This may happen when there is a data flow path change or upon a BF modification attack. Provenance *verification failure rate (VFR)* measures the ratio of packets for which verification fails. Fig. 4(a) shows the VFR for paths of 2 to 12 hops with various BF sizes. For each path length, the VFR is averaged over 1000 distinct paths. The results show that the provenance verification process fails only for a very small fraction of packets. Thus, for most packets the *lightweight verification* process is sufficient to retrieve the provenance. The more costly provenance collection process is executed only for a very few packets when verification fails. As expected, VFR increases linearly with the increase of the path length. On the other hand, VFR is not significantly influenced by BF size, proving that even small BF sizes provide good protection. Fig 4(b) shows the variation of VFR over time, as the number of packet transmissions increases. As the network gets stable with time, the data paths do not change often, and hence the VFR approaches 0.

Fig. 4(c) and 4(d) plot the percentage of *provenance collection error* for different number of hops and the corresponding false positive rates, respectively. Recall that, the collection phase is executed when provenance verification fails. Fig. 4(e) and 4(f) show the collection error corresponding to various BF sizes and the related false positives, respectively. The number of hash functions used are determined using Eq. (3). The resulting false positive rates vary from $0 \sim 0.013$ and it is observed that the collection error becomes negligible when the false positive rate drops at or below 10^{-4}. It is also seen that a BF size of 16 bytes is enough to ensure no decoding error for up to 8-hop paths. The empirical BF size required is much less than the theoretical one($\sim$ 20 bytes for a 8-hop path).



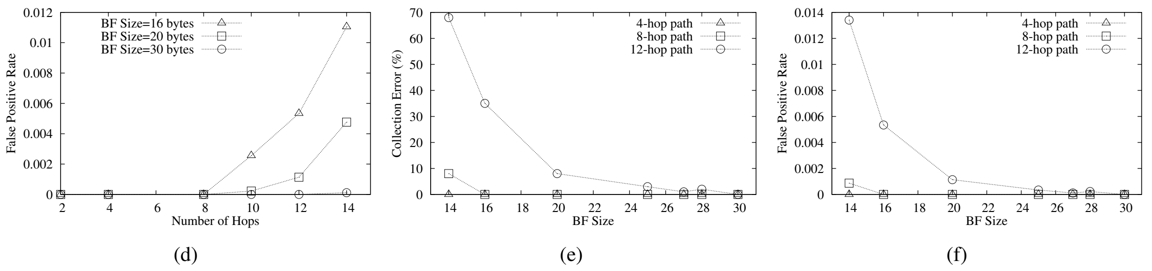


Figure 4. (a) Provenance VFR vs path length. (b) VFR variation with time as network stabilizes. (c)(d)(e)(f) Collection Error and False Positive Rate for various path lengths and BF sizes.

b) Space Complexity and Energy Consumption: Fig. 5(a) shows a comparison among SSP, MP and our provenance mechanism in terms of bytes required to transmit provenance. The provenance length in SSP and MP increases linearly with the path length. For our scheme, we empirically determine the BF size which ensures no decoding error. Although the BF size increases with the expected number of elements to be inserted, the increasing rate is not linear. We see that even for a 14-hop path, a 30 byte BF is sufficient for provenance decoding without any error.

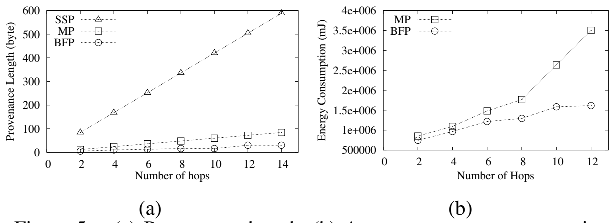


Figure 5. (a) Provenance length. (b) Aggregate energy consumption.

We also measure the energy consumption due to provenance construction and transmission for various hop counts. Note that, modern sensors use *ZigBee* specification for high level communication protocols which allows upto 104 bytes as data payload. Hence, **SSP and MP can be used to embed provenance (in data packet) for maximum 2 and 14 nodes, respectively**. Figure 5(b) compares the aggregate energy consumption of MP with that of our scheme over 100 packet transmissions. The results confirm the energy efficiency of our solution.

VII. Related Work

Pedigree [17] captures provenance for network packets in the form of per packet tags that store a history of all nodes and processes that manipulated the packet. However, the scheme assumes a trusted environment which is not realistic in sensor networks. ExSPAN [18] describes the history and derivations of network state that result from the execution of a distributed protocol. This system also does not address security concerns and is specific to some network use cases. SNP [19] extends network provenance to adversarial environments. Since all of these systems are general purpose network provenance systems, they are not optimized for the resource constrained sensor networks.

Hasan et al. [5] propose a chain model of provenance and ensure integrity and confidentiality through encryption, checksum and incremental chained signature mechanism. Syalim et al. [20] extend this method by applying digital signatures to a DAG model of provenance. However, these generic solutions are not aware of the sensor network specific assumptions, constraints etc. Since provenance tends to grow very fast, transmission of a large amount of provenance information along with data will incur significant bandwidth overhead, hence low efficiency and scalability. Vijaykumar et al. [21] propose an application specific system for near-real time provenance collection in data streams. Nevertheless, this system traces the source of a stream long after the process has completed. Close to our work, Chong et al. [22] propose a scheme for embedding the provenance of data source within the dataset. While it reflects the importance of issues we addressed, it is not intended as a security mechanism, hence, does not deal with malicious attacks. Besides, practical issues like scalability, data degradation, etc. have not been well addressed. In our earlier work [23], secure transmission of the provenance requires several distinct packet transmissions. The underlying assumption is that the provenance remains the same for at least a flow of packets. In this work, we relinquish that assumption.

While BFs are commonly used in networking applications, *iBF*s have only recently gained more attention being utilized in applications such as credential based data path security [11], IP traceback [24], source routing and multicast [25], [26] etc. The basic idea in these works is to encode the link identifiers constituent to the packet routing path into an iBF. However, the encoding of the whole path is performed by the data source, whereas the intermediate routers check their membership in the iBF and forward the packet further based on this decision. This approach is infeasible for sensor networks where the paths may change due to several reasons. Moreover, an intermediate router only checks it own membership which may leave several integrity attacks such as *all-one attack*, *random bit flips* etc., undetected. Our approach resolves these issues by encoding the provenance in a distributed fashion.

VIII. Conclusion

We address the problem of securely transmitting provenance for sensor networks. We propose a light-weight provenance encoding and decoding scheme based on Bloom filters. The security features of the scheme include confidentiality, integrity and freshness. Experimental results evaluating the scheme show that it is efficient, light-weight and scalable. In future work, we plan to protect against attackers that drop data packets entirely, rather than just modifying them. Also, we will implement a real system prototype of our technique.

IX. Acknowledgments

The work reported in this paper has been partially supported by the Northrop Grumman Cybersecurity Research Consortium, and NSF under grants NSF-CNS 0964294 and NSF-CNS 1111512.

References

[1] H. Lim, Y. Moon, and E. Bertino, "Provenance-based trustworthiness assessment in sensor networks," in *Proc. of Data Management for Sensor Networks*, 2010, pp. 2–7.

[2] I. Foster, J. Vockler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," in *Proc. of the Conf. on Scientific and Statistical Database Management*, 2002, pp. 37–46.

[3] K. Muniswamy-Reddy, D. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proc. of the USENIX Annual Technical Conf.*, 2006, pp. 4–4.

[4] Y. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *SIGMOD Record*, vol. 34, pp. 31–36, 2005.

[5] R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: Preventing history forgery with secure provenance," in *Proc. of FAST*, 2009, pp. 1–14.

[6] S. Madden, J. Franklin, J. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Operating Systems Review*, no. SI, Dec. 2002.

[7] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering based heuristic for data gathering and aggregation in sensor networks," in *Proc. of Wireless Communications and Networking Conference*, 2003, pp. 1948–1953.

[8] S. Sultana, E. Bertino, and M. Shehab, "A provenance based mechanism to identify malicious packet dropping adversaries in sensor networks," in *Proc. of ICDCS Workshops*, 2011, pp. 332–338.

[9] C. Rothenberg, C. Macapuna, M. Magalhaes, F. Verdi, and A. Wiesmaier, "In-packet bloom filters: Design and networking applications," *Computer Networks*, vol. 55, no. 6, pp. 1364 – 1378, 2011.

[10] M. Garofalakis, J. Hellerstein, and P. Maniatis, "Proof sketches: Verifiable in-netwok aggregation," in *Proc. of ICDE*, 2007, pp. 84–89.

[11] T. Wolf, "Data path credentials for high-performance capabilities-based networks." in *Proc. of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems.*, 2008, pp. 129–130.

[12] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. of IPSN*, 2008, pp. 245–256.

[13] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proc. of the Intl. Conf. on Embedded networked sensor systems*, 2004, pp. 162–175.

[14] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422–426, 1970.

[15] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *Proc. of INFOCOM*, 2004, pp. 839–850.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire tinyos applications," in *Proc. of the Intl. Conf. on Embedded networked sensor systems*, 2003, pp. 126–137.

[17] A. Ramachandran, K. Bhandankar, M. Tariq, and N. Feamster, "Packets with provenance," Georgia Tech, Tech. Rep. GT-CS-08-02, 2008.

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

[19] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. Loo, and M. Sherr, "Secure network provenance," in *Proc. of ACM SOSP*, 2011, pp. 295–310.

[20] A. Syalim, T. Nishide, and K. Sakurai, "Preserving integrity and confidentiality of a directed acyclic graph model of provenance," in *Proc. of the Working Conf. on Data and Applications Security and Privacy*, 2010, pp. 311–318.

[21] N. Vijayakumar and B. Plale, "Towards low overhead provenance tracking in near real-time stream filtering," in *Proc. of the Intl. Conf. on Provenance and Annotation of Data (IPAW)*, 2006, pp. 46–54.

[22] S. Chong, C. Skalka, and J. A. Vaughan, "Self-identifying sensor data," in *Proc. of IPSN*, 2010, pp. 82–93.

[23] S. Sultana, M. Shehab, and E. Bertino, "Secure provenance transmission for streaming data," *IEEE TKDE*, 2012.

[24] R. Laufer, P. Velloso, D. Cunha, I. Moraes, M. Bicudo, M. Moreira, and O. Duarte, "Towards stateless single-packet ip traceback," in *Proc. of IEEE LCN*, 2007, pp. 548–555.

[25] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, and P. Nikander, "Lipsin: line speed publish/subscribe inter-networking," in *Proc. of ACM SIGCOMM*, 2009, pp. 195–206.

[26] A. Ghani and P. Nikander, "Secure in-packet bloom filter forwarding on the netfpga," in *Proc. of the European NetFPGA Developers Workshop*, 2010.