International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Designing of Secure Privacy Preserving Algorithm for Detecting Packet Loss and Provenance Verification in Wireless Ad-Hoc Networks

K. Anil Kumar[1], Sumithra R[2]

[1]Assistant Professor, Computer Science Department, New Horizon College of Engineering

[2]M. Tech, Student, Computer Network Engineering, New Horizon College of Engineering

Abstract: ***Large-scale sensor networks are deployed in numerous application domains, and 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 introduce additional nodes in the network or compromise existing ones. In this paper, we propose a novel lightweight scheme to securely transmit provenance for sensor data. The proposed technique relies on in-packet Bloom filters to encode provenance. We introduce efficient mechanisms for provenance verification and reconstruction at the base station. In addition, we extend the secure provenance scheme with functionality to detect packet drop attacks staged by malicious data forwarding nodes. We evaluate the proposed technique both analytically and empirically, and the results prove the effectiveness and efficiency of the lightweight secure provenance scheme in detecting packet forgery and loss attacks.***

Keywords: Wireless Sensor Network, Secure Transmission, Sensor Network, lightweight, verification and packet forgery.

1. Introduction

SENSOR networks are used 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 (BS) 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 highlighted the key contribution of provenance in systems where the use of untrustworthy data may lead to catastrophic failures (e. g., SCADA systems). Although provenance modelling, collection, and querying have been studied extensively for workflows and curated databases provenance in sensor networks has not been properly addressed. We investigate the problem of secure and efficient provenance transmission and processing for sensor networks, andprovenance to detect packet loss attacks staged by malicious sensor nodes.

In a multi-hop sensor network, data provenance allows the BS to trace the source and forwarding path of an individual data packet. Provenance must be recorded for each packet, but important challenges arise due to the tight storage, energy and bandwidth constraints of sensor nodes. Therefore, it is necessary to devise a light-weight provenance solution with low 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 (BF) that is transmitted along with the data. Upon receiving the packet, the BS extracts and verifies the provenance information. We also devise an extension of the provenance encoding scheme that allows the BS to detect if a packet drop attack was staged by a malicious node.

2. 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.

2.1 Network Model

We consider a multichip wireless sensor network, consisting of a number of sensor nodes and a base station that collects data from the network. The network is modelled as a graph G (N, L) where N ¼ the set of nodes, and L is the set of links, containing an element li;j for each pair of nodes ni and nj 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 node ID and a symmetric cryptographic key Ki. In addition, a set of hash functions H ¼ fh1; h2; . . . ; hkg are broadcast to the nodes for use during provenance embedding.

2.2 Data Model

We assume a multiple-round process of data collection. Each sensor generates data periodically, and individual values are aggregated towards the BS using any existing hierarchical (i.e., tree-based) dissemination scheme.

A data path of D hops is represented as <nl; n1; n2; . . . ; nD >, where nl is a leaf node representing thedata source, and node ni is i hops away from nl. Each non-leaf node in the path aggregates the received data and provenance with its own locally-generated data and provenance.

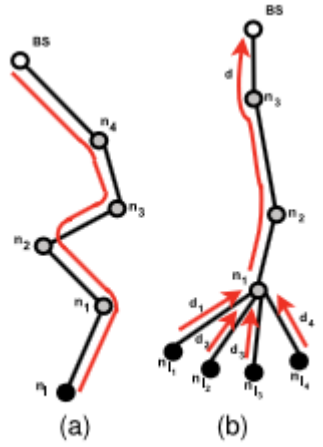

Fig. 1. Provenance graph for a sensor network.

Each data packet contains 1) a unique packet sequence number, 2) a data value, and 3) 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. The sequence number integrity is ensured through MACs, as discussed in Section 3.

2.3 Provenance Model

We consider node-level provenance, which encodes the nodes at each step of data processing. This representation has been used in previous research for trust management and for detecting selective forwarding attacks. Given packet d, its provenance is modelled as a directed acyclic graph GðV; EÞ where each vertex v 2 V is attributed to a specific node HOSTðvÞ ¼ n and represents the provenance record

2.4 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

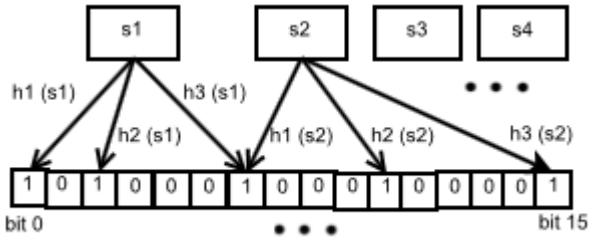

Fig. 2. A Bloom filter with $n = 4$, $m = 16$ and $k = 3$.

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. 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 and hence generate an alarm at the BS.

3. Secure Provenance Encoding

We propose a distributed mechanism to encode provenance at the nodes and a centralized algorithm to decode it at the BS. The technical core of our proposal is the notion of inpacket Bloom filter. Each packet consists of a unique sequence number, data value, and an iBF which holds the provenance. 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.). Our secure provenance technique can be used in conjunction with such work to obtain a complete solution that provides security for data, provenance and data-provenance binding, as shown in Section 3.3.

3.1 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. The VID is generated per-packet based on the packet sequence number (Seq) and the secret key Ki of the host node. We use a block cipher function to produce this VID in a secure manner. Thus for a given data packet, the VID of a vertex representing the node ni is computed as

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

Where E is a secure block cipher such as AES, etc. When a source node generates a packet, it also creates a BF (referred to as ibf0), initialized to 0. The source then generates a vertex according to Eq. (1), inserts the VID into ibf0 and transmits the BF as a part of the packet. Upon receiving the packet, each intermediate node nj performs data as well as provenance aggregation. If nj receives data from a single child nj_1, it aggregates the Partial provenance contained in the packet with its own provenance record. In this case, the iBF ibfj_1 belonging to the received packet represents a partial provenance, i.e., the provenance graph of the sub-path from the source up to nj_1. On the other hand, if nj has more than one child, it generates an aggregated provenance from its own provenance record

At first, nj computes a BF ibfj_1 by bitwise-ORing the iBFs from its children. Ibfj1 represents a partial

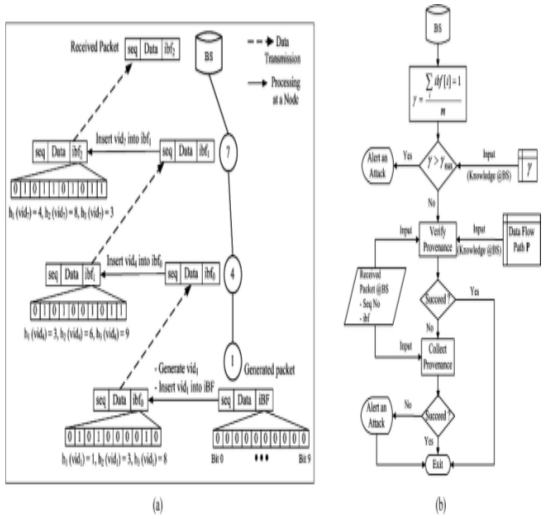

Fig. 3. (a) Mechanism for encoding provenance (node 1 is data source). (b) Provenance processing workflow at the BS upon receiving a packet.

Aggregated provenance from all of the children. In either case, the ultimate aggregated provenance is generated by encoding the provenance record of nj into ibfj_1.

3.2 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 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, a provenance collection process is necessary, 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 encoded in the packet. Below we discuss these processes in detail:

Algorithm 1 ProvenanceVerification

Input: Received packet with sequence *seq* and iBF *ibf*.
Set of hash functions H, Data path $P' = < n'_{i_1}, ..., n'_1, ..., n'_p >$

$BF_c \leftarrow 0$ // Initialize Bloom Filter
for each $n'_i \in P'$ **do**
  vid'_i = generateVID (n'_i, seq)
  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 packet's path is P0. At first, the BS initializes a Bloom filter BFc with all 0's. The BF is then updated by generating the VID for each node in the path P0 and inserting this ID into the BF.

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 testingover all the nodes. For each node ni in the network, the BS creates the corresponding vertex (i.e., vi with VID vidi) using Eq. (1). The BS then performs the membership query of vidi within ibf. If the algorithm returns true, the vertex is very likely present in the provenance, i.e., the host node ni is 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 we show later in Section 6, the false positive probability obtained is very low.

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 $S \leftarrow \emptyset$
  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

Once the BS finalizes the set of potential candidate nodes S ¼ <n0l1; . . . ; n01; n02; . . . ; n>, 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.

First, the algorithm computes the AMs and bucket levels for individual sensors(2) Next, these manifests are unioned up the aggregation topology, only keeping elements at the maximum level max (L1, L2) with every PSR merge. To keep the sketch size under control, the sampling rate drops by a factor of 2

When the sample size grows beyond 2pð1 þ _Þ, where _ < 1 denotes an error parameter

$$\mathcal{M}(\langle L_1, \mathcal{A}_1\rangle, \langle L_2, \mathcal{A}_2\rangle) = \langle L, \mathcal{A}(L, \mathcal{A}_1, \mathcal{A}_2)\rangle,$$

where

$$\langle L, \mathcal{A}(L, \mathcal{A}_1, \mathcal{A}_2)\rangle = \{\langle l, d, n, s_n(d)\rangle \in \mathcal{A}_1 \cup \mathcal{A}_2 : l \geq L\}$$

and

$$L = \begin{cases} max(L_1, L_2), & |\langle L, \mathcal{A}(L, \mathcal{A}_1, \mathcal{A}_2)\rangle| \leq (1+\xi)2p \\ max(L_1, L_2) + 1 & \text{otherwise.} \end{cases}$$

Finally, the evaluation at the BS is performed as

$$\mathcal{E}(\langle L, \mathcal{A}\rangle) = \{d : \langle l, d, n, s_n(d)\rangle \in \mathcal{A}\}.$$

The AM-Sample proof sketches protect against the adversarial inflation of the collected random sample in two ways. First, through the use of authentication manifests. For data tuples, the sketch prevents aggregators from forging new data, since all tuples are signed by a sensor. Second, AM signatures also prevent aggregators from migrating tuples across bucket levels (thereby biasing random sampling choices) since the level is determined. Through hashing by the signed tuple and sensor identifier.

4. Detecting Packet Drop Attacks

We extend the secure provenance encoding scheme to detect packet drop attacks and to identify malicious node (s). We assume the links on the path exhibit natural packet loss and several adversarial nodes may exist on the path. For simplicity, we consider only linear data flow paths (i.e., as illustrated in Fig. 1a). Also,

We do not address the issue of recovery once a malicious node is detected. Existing techniques that are orthogonal to our detection scheme can be used, which may initiate multipath routing or build a dissemination tree around the compromised nodes.

We augment provenance encoding to use a packet acknowledgement that requires the sensors to transmit more meta-data. For a data packet, the provenance record generated by a node will now consist of the node ID and an acknowledgement in the form of a sequence number of the lastly seen (processed/forwarded) packet belonging to that

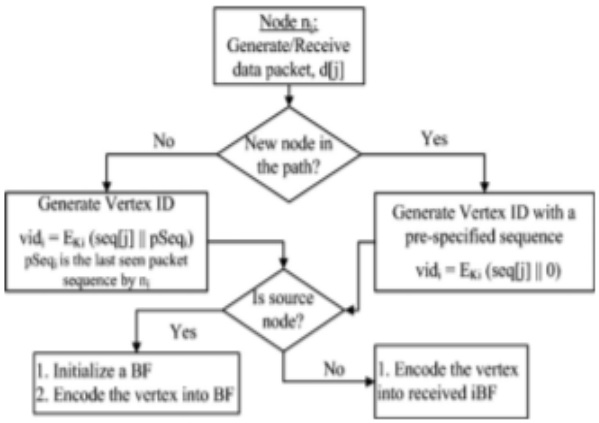

Fig. 4. Extended provenance framework to detect packet drop attacks and identify malicious nodes.

Data flow. If there is an intermediate packet drop, some nodes on the path do not receive the packet. Hence, during the next round of packet transmission, there will be a mismatch

4.1 Data Packet Representation

To enable packet loss detection, a packet header must securely propagate the packet sequence number generated by the data source in the previous round. In addition, as in the basic scheme, the packet must be marked with a unique sequence number to facilitate per-packet provenance generation and verification. Thus, in the extended provenance scheme, any jth data packet contains

1) The unique packet sequence number (seq½j_),
2) The previous packet sequence number (pSeq),
3) A data value, and
4) Provenance.

4.2 Provenance Encoding

Fig. 4 depicts the extended provenance encoding process.

The provenance record of a node includes
1) The node ID,
2) An acknowledgement of the lastly observed packet in the flow. The acknowledgement can be generated in various ways to serve this purpose. In our solution, a node ni creates a vertex vi for every jth packet it generates/forwards. The vertex ID vidi is generated as

$$vid_i = generateVID(n_i, seq[j], pSeq_i) = E_{K_i}(seq[j] \| pSeq_i),$$

(3) Where pSeqi is the knowledge of Ni about the sequence number of the previous packet in the flow. Ni updates the provenance of the packet by inserting vidi into the iBF.

For the remainder of the discussion, we assume that a data packet d½j_ has been dropped by an intermediate node ni. Thus, the nodes nl; n1; . . . ; ni received d½j_ and updated their lastly seen packet sequences to seq½j_. On the contrary, nodes niþ1; . . . ; np as well as the BS did not observe d½j_, They have no information to update the preceding packet sequence, and they retain the same old identifier

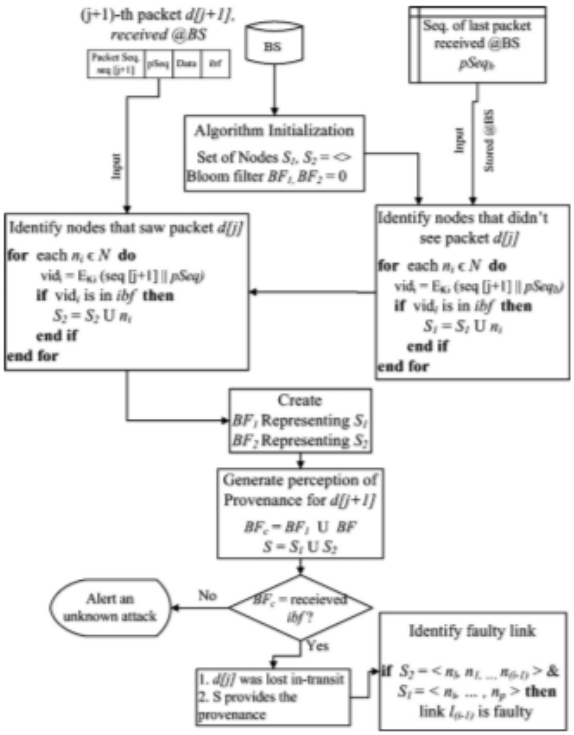

Fig. 5. Provenance collection, packet loss detection, and malicious node

Upon receiving the next packet in the flow, NL; n1; . . . ; ni_1 include seq½j_ in the provenance metadata, whereas niþ1; . . . ; np use seq½j _ 1_ for this purpose when computing

Their VIDs. However, the malicious node Ni may either 1) use seq½j_, or ii) use seq½j _ 1_. Without any loss of generality, we assume that the malicious node encodes seq½j _ 1_ in

5. Security Discussion

In this section, we discuss the security properties of the proposed provenance scheme. Confidentiality. Claim 1. It is computationally infeasible for an attacker to gain information about the sensor nodes included in the provenance by observing data packets. Justification. The confidentiality of the scheme is achieved through two factors: the use of BF and the use of encryption keys. When one-way hash functions are used to insert elements in the BF, the identities of the inserted elements cannot be reconstructed from the BF representation. An attacker may collect a large sample of iBFs to infer some common patterns of the inserted elements.

If the attacker has the knowledge of the complete element space (i.e., provenance records of all the nodes) and the hashing schemes, it can try a dictionary attack by testing for the presence of every element and obtain a probabilistic answer to what elements are carried in a given iBF. However, the elements inserted in the iBF, i.e., provenance records of the nodes, depend on a per-packet variable - sequence number, and also there is a secret key that is used in deriving the node VIDs that are inserted in the iBF.

For legitimate nodes, these secrets are unknown to the attacker, as each key Ki is shared only between the node and the BS. To increase the level of security, we can use pseudo-random functions (PRFs) seeded with the secret key and produce a different key instance at each epoch. Therefore, the shared key is not directly exposed, and each instance key is used only once. Thus, even if an adversary obtains plaintexts and corresponding cipher texts for one epoch, the confidentiality at other time epochs is preserved. To conclude, an attacker cannot gain any information through the observation of packets and the encoded provenance.

6. Performance Analysis

We present an analysis of the space and energy overhead of our scheme. We use the following benchmarks:

1) We adapt the generic secure provenance framework SProve to sensor networks. In this lightweight version of the scheme, referred to as SSP, we simplify the provenance record at a node ni as Pi ¼ <n i; hashðDiÞ; C i >, where hashðDiÞ is a cryptographic hash of the updated data, and Ci contains an integrity checksum as
$$Sign(hash(n_i, hash(D_i)|C_{i-1})).$$
2) 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 prove- nance record.

6.1 Space Complexity

To implement SSP, we use SHA-1 (160 bit) for cryptographic hash operations and the TinyECC library 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 Tiny Seclibrary to compute a 4-byte CBC-MAC. Hence, a provenance record has 6 bytes in this case.

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. The probability is
$$P_{fp} = \left(1 - \left(1 - \frac{1}{m}\right)^{kD}\right)^k \approx (1 - e^{-\frac{kD}{m}})^k.$$

For a given m and D, the number of hash functions that mizes the false positives is computed as [22]
$$k_{opt} = \frac{m}{D} ln2.$$
Given D and a desired false positive probability Pfp, the required number of bits m can be computed by substituting the optimal value of k in Eq. (4) 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 to maintain a fixed false positive probability, the length of a BF should grow with the number of elements. For example, if we consider Pfp ¼ 0:02 and a 14-hop path, the BF size m is computed as 114 bits and kopt ¼ 6. Thus, a 120-bit (15 byte) BF is sufficient to encode provenance while maintaining low false positives. In practice, we bound Pfp by a small constant d> 0Þsuch that Pfp < d. To find the appropriate value of m we have
$$ln(P_{fp}) > ln\delta \Rightarrow -\frac{m}{D} * (ln2)^2 > ln\delta \Rightarrow m < \frac{Dln\frac{1}{\delta}}{(ln2)^2}.$$

7. Simulation Results

We implemented and tested the proposed techniques using the TinyOS simulator have used the micas energy model and PowerTOSSIM z plug-in to TOSSIM to measure the energy consumption.

We consider a network of 100 nodes and vary the network diameter from 2 to 14. All results are averaged over 100 runs. First, we look at how effective the secure provenance encoding scheme (introduced in Section 3) is in detecting provenance forgery and path changes. Next, we investigate the accuracy of the proposed method for detecting packet loss (which was presented in Section 4). Finally, we measure the energy consumption overhead of securing provenance.

7.1 Provenance Decoding Error

Provenance decoding retrieves the provenance from the in-packet BF and consists of verification and collection phases. To quantify the accuracy and efficiency of our provenance

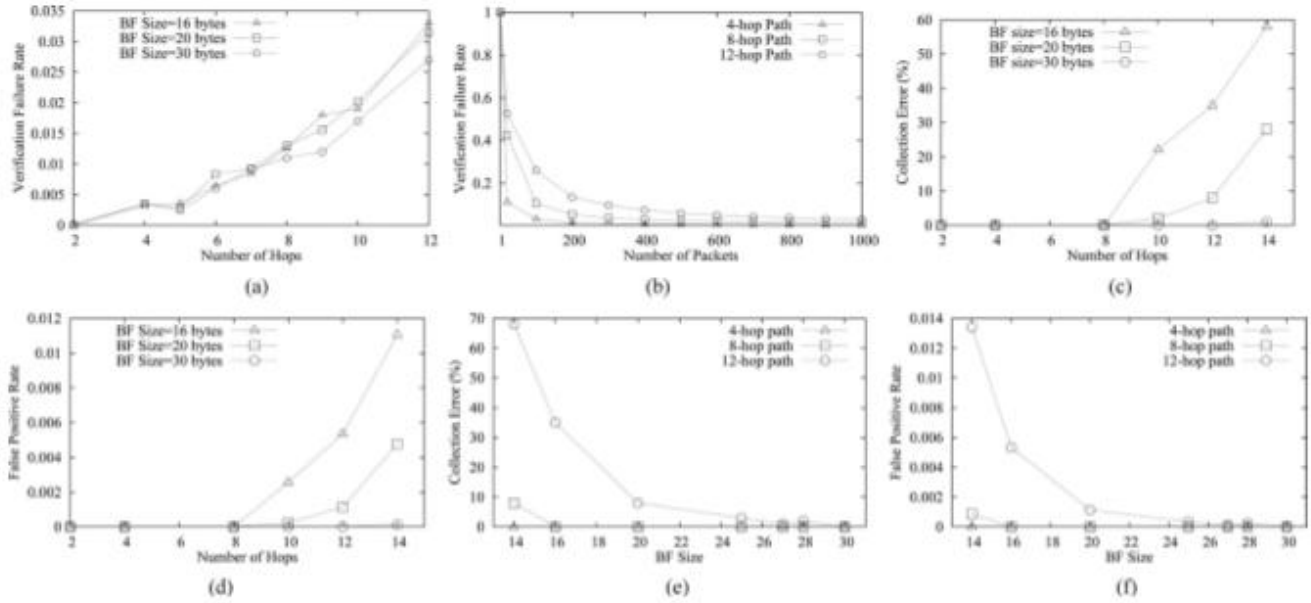

Fig. 6. (a) Provenance VFR vs path length. (b) VFR variation with time as network stabilizes. (c), (d), (e), (f) Collection Error and False Positive Rate for various path lengths and BF sizes.

scheme, we measure the 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 the local knowledge at the BS.

This may happen when there is a data flow path change or upon a BF modification attack. Prove- nance verification failure rate (VFR) measures the ratio of packets for which verification fails. Fig. 6a shows the VFR for paths of 2 to 12 hops with various BF sizes. For each path length, the VFR is averaged over 1,000 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.

8. Related Work

Pedigree 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 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 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.

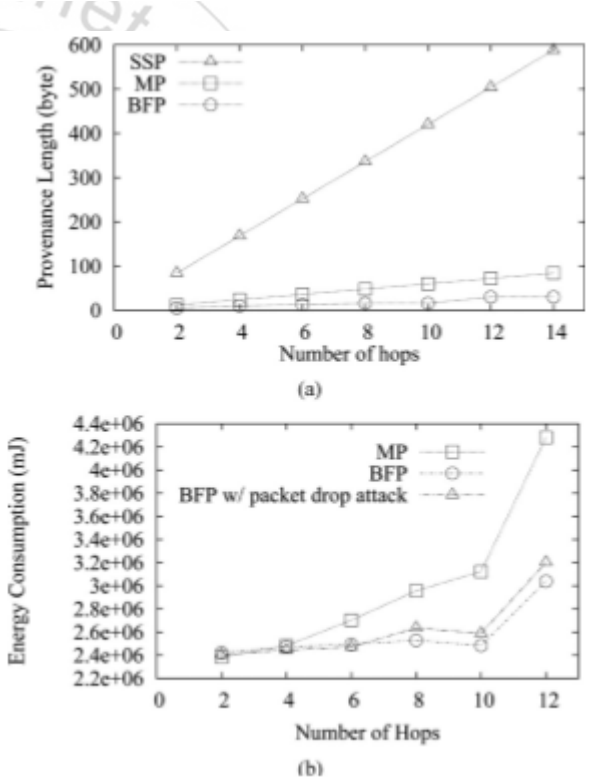

Fig. 8. (a) Provenance length. (b) Energy consumption.

To grow very fast, transmission of a large amount of prove- nance information along with data will incur significant bandwidth overhead, hence low efficiency and scalability. Vijayakumar and Plale 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. Closer to our work, Chong et al. embed the provenance of data source within the data set.

9. Conclusion

We addressed the problem of securely transmitting provenance for sensor networks, and proposed a light-weight provenance encoding and decoding scheme based on Bloom filters. The scheme ensures confidentiality, integrity and freshness of provenance. We extended the scheme to incorporate data-provenance binding, and to include packet sequence information that supports detection of packet loss attacks.

Experimental and analytical evaluation results show that the proposed scheme is effective, light-weight and scalable. In future work, we plan to implement a real system prototype of our secure provenance scheme, and to improve the accuracy of packet loss detection, especially in the case of multiple consecutive malicious sensor nodes.

References

[1] H. Lim, Y. Moon, and E. Bettino, "Provenance-Based Trustworthi- ness Assessment in Sensor Networks," Proc. Seventh Int'l Workshop Data Management for Sensor Networks, pp. 2-7, 2010.
[2] I. Foster, J. Vockler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," Proc. Conf. Scientific and Statistical Database Management, pp. 37-46, 2002.
[3] K. Muniswamy-Reddy, D. Holland, U. Braun, and M. Seltzer, "Provenance-Aware Storage systems," Proc. USENIX Ann. Technical Conf., pp. 4-4, 2006.
[4] Y. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Prove- nanceinE-Science,"ACMSIGMODRecord,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," Proc. Seventh Conf. File and Storage Technologies (FAST), pp. 1-14, 2009.
[6] S. Madden, J. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," ACM SIGOPS Operating Systems Rev., vol. 36, no. SI, pp. 131-146, Dec. 2002.
[7] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An Efficient Clustering Based Heuristic for Data Gathering and Aggregation in Sensor Networks," Proc. Wireless Comm. and Networking Conf., pp. 1948- 1953, 2003.
[8] S. Sultana, E. Bertino, and M. Shehab, "A Provenance Based Mechanism to Identify Malicious Packet Dropping Adversaries in Sensor Networks," Proc. Int'l Conf. Distributed Computing Systems (ICDCS) Workshops, pp. 332-338, 2011.
[9] L. Fan, P. Cao, J. Almeida, and A.Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," IEEE/ACM Trans. Networking, vol. 8, no. 3, pp. 281-293, June 2000.
[10] A.KirschandM.Mitzenmacher,"Distance-SensitiveBloomFilters," Proc.WorkshopAlgorithmEng.andExperiments,pp.41-50,2006.
[11] C. Rothenberg, C. Macapuna, M. Magalhaes, F. Verdi, and A. Wies- maier, "In-Packet Bloom Filters: Design NetworkingApplications,"ComputerNetworks,vol.55,no .6,pp.1364
[12] M. Garofalakis, J. Hellerstein, and P. Maniatis, "Proof Sketches: Verifiable In-Netwok Aggregation," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 84-89, 2007.
[13] T. Wolf, "Data Path Credentials for High-Performance Capabilities-Based Networks," Proc. ACM/IEEE Symp. Architectures for Net- working and Comm. Systems, pp. 129-130,