ISSN: 2321-7782 (Online)
Volume 4, Issue 3, March 2016
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Lightweight Scheme for Detecting Attacks in Wireless Sensor Network

Abhishek Drakshe[1]
Bachelor of Computer Engineering
D.Y. Patil Institute of Engg, Ambi ,Talegaon Dabhade
Pune, India

Rajat Throve[2]
Bachelor of Computer Engineering
D.Y. Patil Institute of Engg, Ambi ,Talegaon Dabhade
Pune, India

Ashwini Ambekar[3]
Bachelor of Computer Engineering
D.Y. Patil Institute of Engg, Ambi ,Talegaon Dabhade
Pune, India

Prof. Rupali Adhau[4]
Master of Computer Engineering
D.Y. Patil Institute of Engg, Ambi ,Talegaon Dabhade
Pune, India

Abstract: ***Considerable sensor networks will be implemented in numerous software domains, plus the info they gather are becoming used in decision-making intended for crucial infrastructures. Data will be live-streaming from multiple resources through intermediate processing nodes that aggregate information. A destructive adversary may expose extra nodes in the network or compromise existing types. Therefore, assuring excessive facts trustworthiness is vital for appropriate decision-making. Facts provenance presents a vital aspect in analyzing the standing of fuller details Provenance management intended to sensor networks introduces a number of challenging requirements, such because low strength and band width usage, efficient storage and protected transmission. Here, we propose a novel lightweight scheme to securely transmit provenance designed for sensor data. The recommended technique is determined simply by package Bloom filters to encode provenance. We present effective mechanisms for source confirmation and reconstruction in the base station. In addition, all of almost all of us extend the secure resource scheme with functionality to identify box drop difficulties staged simply by malicious facts forwarding nodes. We examine the suggested technique both equally analytically and empirically, as very well as the results show the performance and effectiveness of the lightweight protected provenance system in discovering packet forgery and reduction attacks.***

Keywords: Bloom filters, publish/subscribe, multicast forwarding, Base station (BS), sensor networks (SN), novel lightweight scheme.

I. INTRODUCTION

Fuller networks are being used in several application area, such because cyber physical infrastructure devices, environmental monitoring, power plants, etc. Data are developed at large number of sensor node sources and processed in-network at advanced hops on their method to a Base station (BS) that performs decision-making. The diversity of data sources creates that want to assure the standing up of Information, such that only dependable information is usually considered in the decision process. Data provenance is usually a powerful method to examine data trustworthiness, since that summarizes a brief history of possession and the actions performed on the data. Latest research highlighted the key contributions of provenance in devices where the consumption of untrustworthy info may cause catastrophic outages (e. g., SCADA systems). Although provenance modelling, collection, and querying have recently been studied extensively for work flow and curated databases, source in sensor networks offers not been properly resolved. We investigate the condition of secure and efficient source transmission and processing intended for sensor networks, and all of us use provenance to discover packet loss attacks taking place by malicious sensor nodes.

Provenance management for sensor networks introduces several challenging requirements, such as low energy and bandwidth consumption, efficient storage and secure transmission. In this project, we are proposing a novel lightweight scheme to securely transmit provenance for sensor data. 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. Therefore, assuring high data trustworthiness 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 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.

II. Related Work

Conventional provenance protection solutions use cryptography and digital signatures which need an encryption, checksum, and incremental chained signature mechanisms. Also uses digital signatures for a DAG model of provenance. Like this solutions not applicable on sensor networks because of their particular constrained resources. Propose a near real-time provenance for application specific systems which trace the source of information streams after the process has completed. However, the real time operations in sensor networks needed quick responses before processing the information to stop malicious activities which can cause catastrophic failures. Other approaches capture provenance of network packets in the form of per packet tags that store the history of all nodes. However, such approaches have large memory need especially in large scale sensor networks. Proposes a scheme that embeds the provenance of a data source within a data set .So, this approach is not intended as a security mechanism and never deals with malicious attacks. Our approach is designed to specifically protect from malicious attacks while on similar time assuring the best performance.

III. Background and System Model

Background:

A. Network Model: We have create a multi hop wireless sensor network, consisting of a number of sensor node and a base station that collects data from the network. The networks 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 link, containing an element l i, j for each pair of nodes n i and n j that are communicating directly with each other. The Base station assigns each node a unique identifier node ID and a symmetric cryptographic key K i .

B. Data Model: We consider a multiple-round process of collecting data. Each sensor generates data periodically, and individual values are aggregated towards the Base station using any existing hierarchical dissemination scheme . Each data packet contains of (i) a unique packet sequence number, (ii) a data value, and (iii) provenance.

C. Threat Model: It is also important to provide Data-Provenance Binding i.e., a coupling between data and provenance so that an attacker cannot successfully drop or alter the legitimate data while retaining the provenance, or swap the provenance of two packets.

D. The Bloom Filter (BF): Several BF variations that provide additional functionality exist. A Counting Bloom Filter (CBF) associates a small counter with every bit, which is incremented/decremented upon item insertion/deletion. To answer approximate set membership queries, the distance sensitive Bloom filter has been proposed. However, aggregation is the only

operation needed in our problem setting. The cumulative nature of the basic BF construction inherently supports the aggregation of BFs of a same kind, so we do not require CBFs or other BF variants.

E. Provenance Model: All of us consider node-level provenance, which usually encodes the nodes that are involved each and every step of info processing. This representation features been used in earlier research for trust administration as well as for detecting selective forwarding disorders. Given a great information packet d, it is provenance is modelled because a directed acyclic chart (V, E) where every vertex v? V is usually attributed to a particular client HOST(v) = n and represents the provenance record (i. e. node ID) for that node. Every vertex in the source graph is uniquely determined by a Vertex Identification (VID) which is made by the host client using cryptographic hash features. The advantage set Electronic contains directed edges that hook up sensor nodes.

System Model:

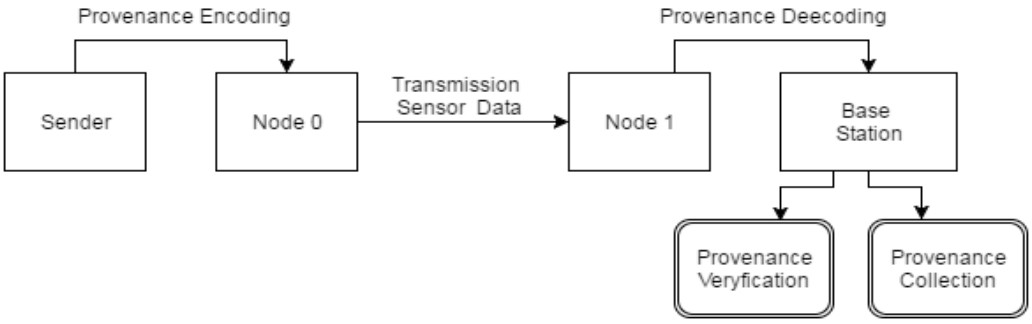

Fig1. System Architecture

IV. MATHEMATICAL MODEL OF SYSTEM

Let W be the whole system which consists:

W= {IP, PRO, OP}

IP is the input of system.

IP= {BS, G, N, L, K, H, d, ID, V, E, S, BF}.

Where,

1. Let BS is the Base Station which collects data from network.
2. Let G is the graph , G(N,L)

   Where, N is the set of nodes.

   N = {ni|, $1 \leq i \leq |N|$} is 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.
3. K is set of symmetric cryptographic key
4. H is a set of hash functions

   H = {h1, h2, ...,hk} .
5. E is edge set consists of directed edges that connect sensor nodes.
6. d is the set of data packets,

Let G is 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. node ID) 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.

Procedure:

Let S is a set of items

S = {s1, s2, ...,sn}

We use an array of m bits with k independent hash functions h1, h2, ...,hk.

The output of each hash function hi maps an items uniformly to the range [0, m-1], i.e., an index in am-bit array.

Let BF is the Bloom Filer, can be represented as {b0, . . . , bm−1}.

Initially all m bits are set to 0.

To insert an element s ∈ S into a BF, s is hashed with all the k hash functions producing the values hi(s) ($1 \leq i \leq k$).

The bits corresponding to these values are then set to 1 in the bit array.

To query the membership of an item s` within S, the bits at indices hi(s`) ($1 \leq i \leq k$) are checked. If any of them is 0, then certainly s` not within S.

Otherwise, if all of the bits are set to 1, s`∈ 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 hi(s`) being set to 1 even if s` not within S. This is called a false positive.

V. PROVENANCE ENCODING AND DECODING

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 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 niis computed as vidi= generate VID(ni, seq) = EKi(seq) (1) 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 iBFibfj−1 belonging to the received packet represents a partial provenance, i.e., the provenance graph of the sub-path from the source upto nj−1. On the other hand, if nj 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, nj computes a BF ibfj−1 by bitwise-ORing the iBFs from its children. ibfj−1 represents a partial 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. To this end, nj creates a vertex using Eq. (1) and inserts the VID into ibfj−1 which is then referred to as ibfj. 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.

We 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. 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 in-packet Bloom filter (iBF). 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 Base station. We 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.

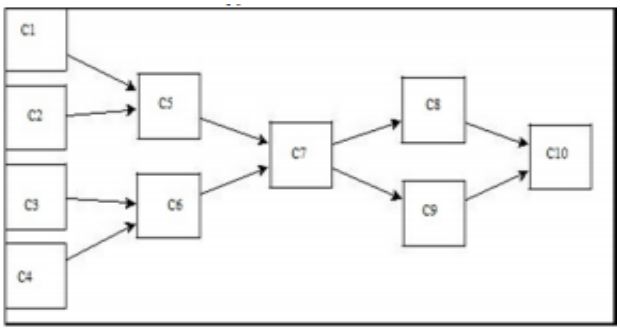

Fig 2. Provenance Encoding Graph

The Figure shows that to produce the final result, the contributor C5 uses the outputs of contributors C1 and C2 while contributor of C6 uses the output of contributors C3 and C4. Contributor C7 uses the output of C5 and C6 which later used by C8 and C9. C10 is the final process is executed by that processes the outputs of C8 and C9. After each process is executed and the provenance of the process we had created/generated, the provenance is stored in the provenance database.

Provenance Decoding

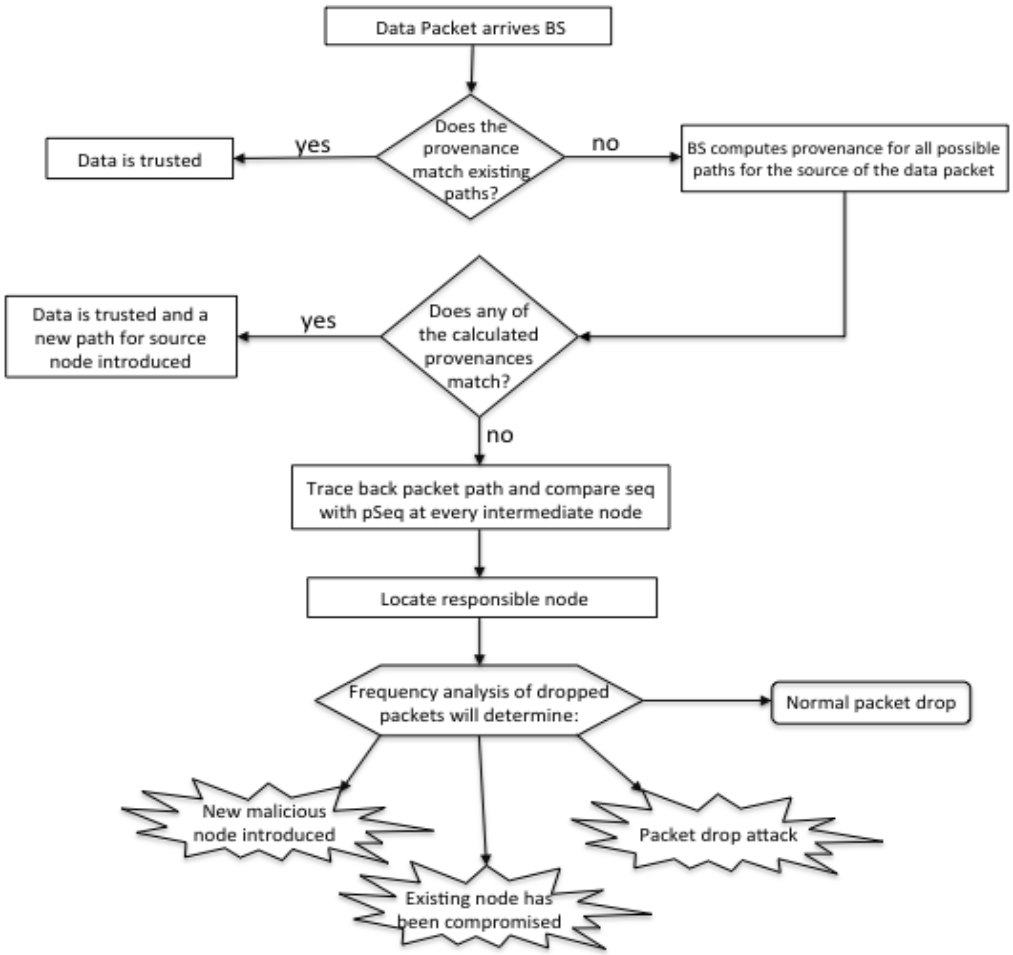

Fig 3: Provenance decoding at the Base Station (BS)

The above figure shows the provenance decoding mechanism steps. When the base station receives a data packet. It checks for the all possible safe paths of packets from the same source node of this packet. It then compute the provenance of these paths by using the information in the packet and compares it with the provenance encoded by the bloom filter enclosed with the data packet. If there is a match , data is considered for further processing and analysis.

If there is mismatch then, the arrived data packet has either taken new route that could be safe but had not been previously saved at base station node, or a previous packets of the same source node has been dropped while on its way to the base station.

In former case, the base station trace the path and computes the provenance at each node until it reaches the source node If the provenance matches, then the base station add this new route to its set of safe paths. If there is still mismatch then the data packet has been tampered by inserting malicious node into the network. In this case we check for compromised nodes and change the route dynamically to avoid such a nodes. In the latter case where the mismatch is caused by a dropped packet, this dropping may be due to non malicious network errors, or to malicious attacks. In the latter case the base station traceback the node responsible for dropping packets and analyses the frequency dropped packets by this node to determine whether it is normal behaviour or an attack. This procedure is executed by every tracking back every intermediate node starting from the base station and comparing the components used for the encoding the provenance and finally comparing the nodes last sequence number of the data packet(pSeq) with the current sequence no. of the passing data packet (Seq) from the corresponding source node.

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.

VI. Algorithm Provenance Verification and Provenance Collection

Algorithm 1: Provenance Verification

Input: Received packet with sequence seq and iBF ibf.

Set of hash functions H, Data path P’ =< n’l1 , ..., n’1,…. n’p >

BFc □ 0 // Initialize Bloom Filter

for each n'i ∈ P’ do

vid’i = generate VID generate VID (n’i, seq)

insert vid’i into BFc using hash functions in H

end for

if (BFc = ibf) then

return true // Provenance is verified

end if

return false

Explanation: 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 P’. 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 P’ and inserting this ID into the BF. BFc now reflects the perception of BS about the encoded provenance. To validate its perception, the BS then compares BFc to the received iBF ibf . The provenance verification succeeds only if BFc is equal to ibf . Otherwise, if BFc 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.

Algorithm 2 Provenance Collection:

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 □ ∅

Bloom Filter BFc □ 0 // To represent S

2. Determine possible nodes in the path and build the representative BF

for each node ni ∈ N do

vidi = generateVID (ni, seq)

if (vidi is in ibf) then

S □ S ∪ ni

insert vidi into BFc using hash functions in H

endif

endfor

3. Verify BFc with the received iBF

if (BFc = ibf) then

return S // Provenance has been determined correctly

else

return NULL // Indicates an in-transit attack

endif

Explanation: 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 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 3: MD5

1. Pad message so its length is 448 mod 512
2. Append a 64-bit original length value to message
3. Initialize 4-word (128-bit) MD buffer (A,B,C,D)
4. Process message in 16-word (512-bit) blocks:

1. Using 4 rounds of 16 bit operations on message block & buffer

2. Add output to buffer input to form new buffer value

5. Output hash value is the final buffer value

This program uses getBytes() method to encode the String into a sequence of bytes using the platform's default char set, storing the result into a new byte array. These byte array data is converted into hash code by invoking the following methods update (byte[]), md5final (byte[]), dumpBytes (byte[]).

Description: MD5 is an algorithm that is used to verify data integrity through the creation of a 128-bit message digest from data input (which may be a message of any length) that is claimed to be as unique to that specific data as a fingerprint is to the specific individual. MD5, which was developed by Professor Ronald L. Rivest of MIT, is intended for use with digital signature applications, which require that large files must be compressed by a secure method before being encrypted with a secret key, under a public key cryptosystem. MD5 is currently a standard, Internet Engineering Task Force (IETF) Request for Comments (RFC) 1321. According to the standard, it is "computationally infeasible" that any two messages that have been input to the MD5 algorithm could have as the output the same message digest, or that a false message could be created through apprehension of the message digest. MD5 is the third message digest algorithm created by Rivest. All three (the others are MD2 and MD4) have similar structures, but MD2 was optimized for 8-bit machines, in comparison with the two later formulas, which are optimized for 32-bit machines. The MD5 algorithm is an extension of MD4, which the critical review found to be fast, but possibly not absolutely secure. In comparison, MD5 is not quite as fast as the MD4 algorithm, but offers much more assurance of data security.

VII. Detecting Packet Drop Attack

All of us extend the secure source encoding scheme to discover packet drop attacks also to identify malicious node(s). We all assume the links upon the path exhibit organic packet loss and a number of adversarial nodes may be able to be found on the path. Intended for simplicity, we consider just linear data flow pathways. Also, we do not really address a deficiency of recovery when a malicious node is usually detected. Existing techniques that are orthogonal to the detection scheme can become used, which may start multi path routing or perhaps build a dissemination shrub throughout the compromised nodes.

We augment provenance encoding to use a packet acceptance that requires the detectors to transmit more meta-data. For a data box, the provenance record made with a node will certainly now consist of the node ID and a great acknowledgement in the kind of a chapter number of the last but not least seen (processed/forwarded) box belonging to that info flow. If there is usually an intermediate packet drop, some nodes on the path usually do not receive the packet. Hence, throughout the following round of packet transmitting, there will be a mismatch between the acknowledgements made from different nodes on the path. All of us make use of this fact to discover the packet drop attack also to localize the malicious node. We consider a data flow route P where nl may be the only data source. We all denote the link among nodes ni and n(i+1) as li. We illustrate next packet representation, source encoding and decoding intended for detecting packet loss.

VIII. Security Discussion

Privacy Claim 1: It is usually computationally infeasible for a great attacker to gain information regarding the sensor nodes included in the provenance simply by observing data packets.

Reason: The confidentiality from the structure is achieved through two factors: consumption of BF and the consumption of encryption keys. Once one-way hash features are being used to insert elements in the BF, the identities of the inserted elements cannot be reconstructed coming from the BF representation. A great attacker might accumulate a huge sample of iBFs to infer a few common habits of the inserted components. If the attacker has got the knowledge of the total aspect space (i. electronic. provenance records of almost all the nodes) and the hashing schemes, it may try a dictionary assault by testing for the presence of every aspect and obtain a probabilistic solution to what {factors are} carried within a provided iBF. However, the

components inserted in the iBF, i. e., provenance information of the nodes, {rely upon} a per-packet variable -- sequence number, and likewise there is a key that can be used found in deriving the node Videos that are inserted in the iBF. For genuine nodes, these secrets will be unknown towards the attacker, while each key Ki is usually shared only between the node at BF. To increase the level of security, we could make use of pseudo-random functions (PRFs) seeded with the secret major and produce a diverse key instance at every epoch. Consequently, the shared key is usually not directly exposed, and each instance key can be used only once. Thus, regardless if an adversary obtains plaintexts and corresponding cipher texts for just one epoch, the confidentiality in other time epochs is usually preserved. To conclude, a great opponent cannot gain any kind of information throughout the statement of packets as well as the protected provenance.

Ethics

Claim 2: An opponent, acting alone or colluding with others, cannot effectively add or legitimate nodes to the provenance of data made by the compromised nodes.

Justification: Attacker(s) may attempt to create fake data and create the provenance including a few innocent nodes < n'l, n'1, n'2, ..., n'p > to make all of them accountable for false data and consequently to mark all of them as untrustworthy. However, the provenance embedding process needs the node specific key Ki for cryptographic calculation of the related video, and the attackers have no idea the key for the legitimate nodes. Hence, this kind of attack will fail.

Quality

Claim 3: Provenance play back attacks are detected simply by our proposed scheme.

Reason: Since provenance encoding will depend on a packet specific info, the value of the constructed iBF varies coming from packet to packet. Therefore, malicious attempts to connect a previously captured iBF with a more latest info packet (benign/fake) is usually detected at the BF.

IX. Conclusion

This paper addressed the condition of securely transmitting provenance intended to get sensor networks, and suggested a light-weight provenance development and decoding scheme depending on Bloom filters. The system ensures confidentiality, integrity and freshness of provenance. All of us extended the scheme to include data-provenance binding, also to consist of packet sequence information that supports detection of box loss attacks. Experimental and analytical results display that the proposed system works well, light-weight and international. At a later day work, we plan to implement a real program prototype of the secure source scheme, also to increase the accuracy and reliability of packet loss recognition, especially in the circumstance of multiple consecutive harmful sensor nodes.

Acknowledgement

We would want to thank the analysts and also distributers to make their assets available. We additionally appreciative to commentator for his or her significant advice furthermore thank the college powers for giving the obliged base and support.

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. 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.
3. L. Fan, P. Cao, J. Almeida, and A. Z. Broder, “Summary cache: a scalable wide-area web cache sharing protocol,” IEEE/ACM Trans. Netw., vol. 8, no. 3, pp. 281–293, Jun. 2000.
4. Kirsch and M. Mitzenmacher, “Distance-sensitive bloom filters,” in Proc. of the Workshop on Algorithm Engineering and Experiments, 2006, pp. 41–50.
5. 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.

6. S. Roy, M. Conti, S. Setia, and S. Jajodia, "Secure data aggregation in wireless sensor networks," IEEE Transactions on Information Forensics and Security, vol. 7, no. 3, pp. 1040–1052, 2012.
7. 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.
8. 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.
9. Y. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," SIGMOD Record, vol. 34, pp. 31–36, 2005.
10. 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.
11. 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.
12. 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.

AUTHOR(S) PROFILE

Abhishek Drakshe, pursuing B.E in computer engineering at D.Y. Patil Institute of Engg, Ambi, Talegaon Dabhade, Pune, India.

Rajat throve, pursuing B.E in computer engineering at D.Y. Patil Institute of Engg, Ambi, Talegaon Dabhade, Pune, India.

Ashwini Ambekar, pursuing B.E in computer engineering at D.Y. Patil Institute of Engg, Ambi, Talegaon Dabhade, Pune, India.

Prof. Rupali Adhau, Professors at D.Y. Patil Institute of Engg, Ambi, Talegaon Dabhade, Pune, India

