A Fast And Easy Mechanism For Identifying Data Provenance Imitation And Packet Drop Attacks In Wireless Sensor Networks

JYOTI1 , PATIL REKHA2

1PG Student, Department of CSE, PDA College of Engineering ,Kalaburagi,Karnataka,India.
2Assistant Professor, Department of CSE, PDA College of Engineering ,Kalaburagi,Karnataka,India.

---***---

Abstract -*Large scale sensor networks are used in many application domain and those data collected at the destination are used in decision making. Data is transferred from source to base station through many intermediate nodes. When the packets are transmitting from these nodes, a malicious adversary may compromise with the existing ones or introduce additional nodes in the network. Therefore, the data which is obtained at the destination by passing all these nodes cannot be trusted, as the data may get modified or dropped by an attacker. And also there are several challenging requirements like low energy, less bandwidth consumption, less storage and secure transmission. Hence we introduce a fast and easy mechanism for identifying data provenance imitation and packet drop attacks in wireless sensor networks.*

Key Words: **Data Provenance, Security, Sensor Networks, In-Packet Bloom Filter, Base Station.**

1. INTRODUCTION

Sensor networks are used in many application domains like Environmental monitoring, Military applications and many more . Data is transmitted from source to destination (base station) while passing from many intermediate nodes. And the data that is obtained at the base station are used for decision making. Some research [1] highlighted that provenance in systems lead to catastrophic failures and [2] curated database , [3] provenance has not been properly addressed.

In multi-hop networks the data which is transmitted from source to base station via many intermediate nodes cannot be considered as trustworthy . Hence confidential and integrity also the originality of the data should be maintained.
So we introduce a provenance strategy where data on each node securely inserts in the bloom filter and transmitted along the data. The bandwidth is efficiently used, and also provides low error rates by using bloom filter. At the destination it extracts and verifies the information to check weather packet is modified or dropped or original information is received. [4] Uses two channels for transmission of data and provenance. We require only single transmission channel. [5] uses digital signatures and cryptography for provenance security which leads to high costs.

Objectives of this paper are
To form a provenance encoding and decoding system which fulfills such safety and performance needs.
To invent an addition of the provenance encoding system that lets the BS(Base station) to identify if a packet drop attack was showed by a malicious node.
To invent an addition of the provenance encoding system that lets the BS to identify if a packet modified was staged by a malicious node.
We suggest an in-packet Bloom Filter (iBF) provenance-encoding system.

2. PROPOSED WORK

2.1. System Architecture

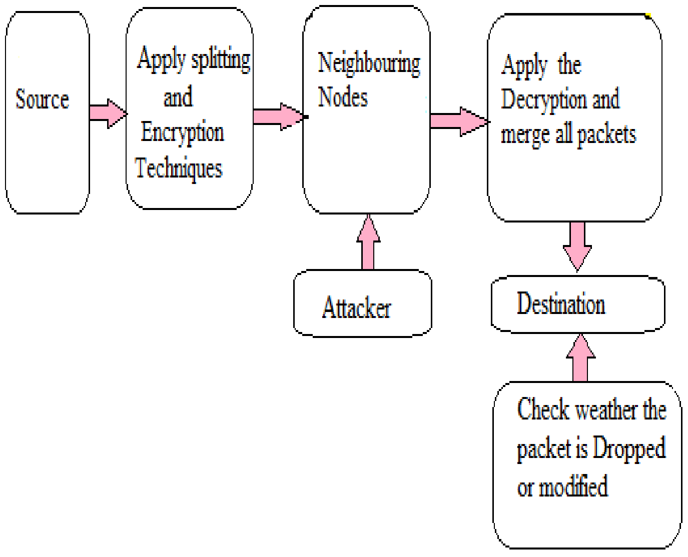

Source
Apply splitting and Encryption Techniques
Neighbouring Nodes
Apply the Decryption and merge all packets
Attacker
Destination
Check weather the packet is Dropped or modified

Fig-1: System Architecture

1 . Node Configuration .

a. Link Configuration : In this module fixed number of nodes are configured. We create the network group by connecting nodes to sink.

2. Sender Node.
a. Packet Splitting: In this module Sender selects the file which is to be sent, then split it into the number of packets based on the size of the file.

b. Send Packets to Intermediate: Then sender encrypts all the splitted packets by adding some bits to each packets before sending packets to intermediate nodes. Identification for sender is done by the bit addition for each packet. Once the bits are added to each packet, it sends the packets to the nearest node or intermediate node.

3. Intermediate Node (Router).

a. Send Packets to Sink: In this module the intermediate node receives Packets from the sender. Once the packets are received, it encrypts all packets again for authentication. Before sending to sink, intermediate node add some bits to each packet for node identification. And when bits are added from intermediate node ,it sends all packets to the sink.
b. Modify or Drop: Before sending all packets to sink, packets dropping or packets modification may occur in intermediate nodes.

4. Sink

a .Verify: In this module sink receives all packets from the sender node, and it verifies all packets which are dropped or modified based on the bit information.
b. Merge Packets: After receiving all packets at the sink, it decrypts them. After the decryption, all the packets are merged including modified or dropped. At last original file is reached to the destination.
c. Categorization And Ranking.
Based on the node behavior, categorization and ranking is performed and sink gives ranking as Good, Temporarily Good, Suspiciously Bad, Bad .

3. PROVENANCE MODEL

3.1. Provenance model

Node-level provenance is considered here, which encodes the nodes at each step of data processing from source to sink. This representation has been used in previous research for trust management[1] and for detecting selective forwarding attacks [8]. Let the packet be d_i, its provenance is given as a directed acyclic graph G(V,E) where each vertex v €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 separately identified by a vertex ID (VID) which is generated by the host node . The edge set E consists of directed edges which connects sensor node.

Fig.2 shows examples of provenance.

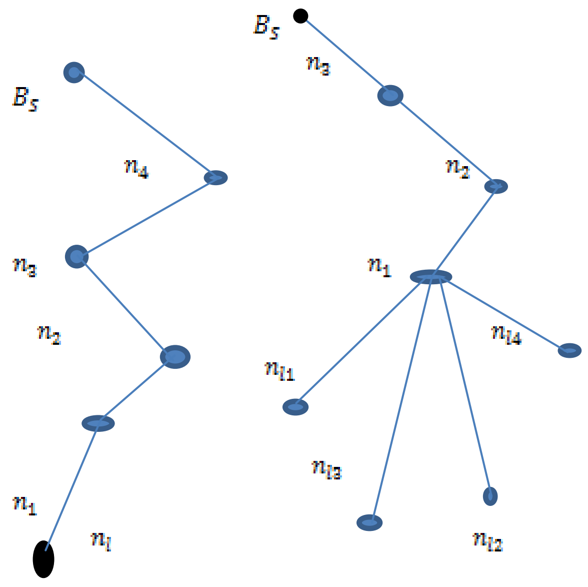


Fig:2a Fig:2b

Fig-2(a,b) :Provenance graph for a sensor network .

In Fig. 2a, the leaf node n_l generates a data packet (d), and each intermediate node aggregates its own sensory data with and 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 Fig. 2b, the internal node n1 generates the data d by aggregating data $d_1 \dots\dots d_4$ from $n_{l1}, \dots\dots n_{l4}$ and then passing it to the destination or base station.

Our objective is to achieve the following security properties.
Confidentiality: A challenger cannot obtain any knowledge about the data provenance by analyzing the whole content of the packets.
Integrity: A challenger cannot add or remove the non-colluding nodes from the provenance of original data.
Freshness: A challenger or an enemy cannot repeat or duplicate the captured data and provenance without being detected by the BS. It is also important to provide Data-Provenance Binding.

A Bloom Filter.

The Bloom Filter(BF): The BF is a space-efficient data structure for probabilistic representation of a set of items S = $\{s_1, s_2, s_3, s_n\}$ using an array of m bits with k independent hash functions $h_1, h_2, \dots, h_k$. The output of each hash function hi maps an item s uniformly to the range $[0, m-1]$, i.e., an index in a m-bit array. The BF can be represented as $\{b_0, \dots, b_{m-1}\}$. Initially all m bits are set to 0.

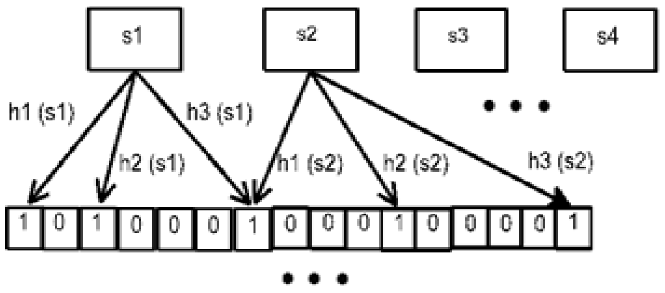


Fig-3: A Bloom filter with n=4, m = 16 and k =3.

4 .SECURE PROVENANCE

4.1. Provenance Encoding

For a data packet, provenance encoding mentions to generating the vertices in the provenance graph and inserting them into the iBF. Each vertex starts at a node in the data path and represents the provenance record of the host node. A vertex is individually identified by the vertex ID. 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 VID in a secure manner. Thus for a given data packet, the VID of a vertex representing the node n_i is computed as

$vid_i = generateVID(n_i, seq) = E_{ki}$ (1).

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

When a source node produces a packet, it also creates a BF (referred to as ibf_0), initialized to 0. The source then produces 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 bits in the packet bloom filter, ibf_{j-1} belonging to the received packet constitutes a partial provenance, i.e., the provenance graph of the sub-path from the source up to 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 from its children. ibf_{j-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 n_j into ibf_{j-1}. To this end, n_j creates a vertex using Eq. (1) and 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.

4.2. Provenance Decoding

As soon as the Base station receives a data packet, it performs the provenance verification process, which believes that the Base station knows what the data path should be, and checks the in-packet bloom filter to see whether the correct path has been followed or not. However, once the network scatters, 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. Provenance collection process can be considered here ,which gets provenance from the obtained in-packet bloom filter. Thus the BS obtains data path from a source node. It is sufficient for the BS to verify its knowledge of provenance with that encoded in the packet, once the packet are received.

Now we will discuss the process in detail.

```
[1]   Algorithm 1 provenance verification
Input: Received packet with sequence $seq$ and in packet BF $ibf$.
Set of hash functions $H$,data path
$P' =< n'_{l1}, \ldots, n'_l \ldots, n'_p >$
$BF_c \leftarrow 0$   // initialize bloom filter
for each $n'_i \in p'$ do
$vid'_i = generate\ VID(n'_i, seq)$
Insert $vid'_i$ into $BF_c$ using hash functions in $H$
end for
If ($BF_c = ibf$) then
return true   // provenance is verified
end if
return false
```

Provenance verification: The Base Station carryout the verification process to conform its knowledge of provenance and also to check the unity of the transmitted provenance .Algorithm 1 describes 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 makes the Bloom filter BF_c 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. BF_c now reflects the perception of BS about the encoded provenance. To check 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 the data path is changed or a BF modification attack is occurred. The verification failure stimulates the provenance collection process which tries to obtain the nodes from the encoded provenance and also to distinguish between the events of a path change and an attack.

4.3 Scheme For Data Provenance Binding

One of the major security challenges for a provenance scheme is to combine data and provenance. In an aggregation infrastructure, the data value is revised at each intermediate node which makes it a critical problem to maintain the relationship between provenance and the intermediate data. Our concern is to implement a secure provenance scheme, we utilize secure in-network aggregation mechanisms to connect provenance with the intermediate aggregation results. Our goal is to absorb provenance scheme with a safer aggregation mechanism so that the aggregation verification process can also be used to verify the data-provenance binding .To perform this purpose, we can utilize an existing secure aggregation scheme such as [6], [7], [8].

5. DETECTING PACKET DROP ATTACKS

We expand the secure provenance encoding scheme to detect packet drop attacks and to identify malicious node(s). We presume the links on the path exhibit natural packet loss and several adversarial nodes may exist on the path. We now increase the provenance encoding to utilize a packet acknowledgement which requires 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 previously seen (processed/forwarded) packet belonging to that 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 an inconsistent between the acknowledgements generated from different nodes on the path. We use this fact to detect the packet drop attack and to localize the malicious node.

5.1 Data Packet Representation

To find the packet loss detection, a packet header must securely propagate the packet sequence number generated by the data source in the previous round. In the basic scheme, the packet must have a unique sequence number to facilitate per-packet provenance generation and verification. Thus, in the extended provenance scheme, any j^{th} data packet contains 1) the unique packet sequence number $(seq(j))$,2) the previous packet sequence number$((pSeq)$., 3)a data value, and 4) provenance.

5.2 Provenance Encoding.

The provenance record for each node has the node ID and an acknowledgement of the last seen packet in the flow .Here in this solution, a node n_i creates a vertex v_i for every j^{th} packet it generates/forwards.

The vertex ID vid_i is generated as

$$vid_i = generateVID(n_i, seq[j], pseqi) \quad (2)$$

where $pSeq_i$ is the knowledge of n_i about the sequence number of the previous packet in the flow. n_i updates the provenance of the packet by inserting vid_i into the iBF. After a node n_i forwards any j^{th} packet, it updates the $pSeq_i$ record for the corresponding data flow with the recently processed packet sequence, $seq[j]$. If a packet that is received at a node from a data flow for which it has no previous packet information, then it may use a pre-specified special purpose identifier, such as 0, as the previous packet sequence $pSeq_i$. This labels the case of change in a routing path ,where a new node in the path can use this special identifier for encoding provenance. And, if a node does not obtain packets from a data flow for a prolong time, it can clear the previous packet information for that flow to reduce space overhead. The node can get updated and maintain this flow-specific record when it receives packets from that flow more frequently.

5.3 Provenance Decoding at the BS

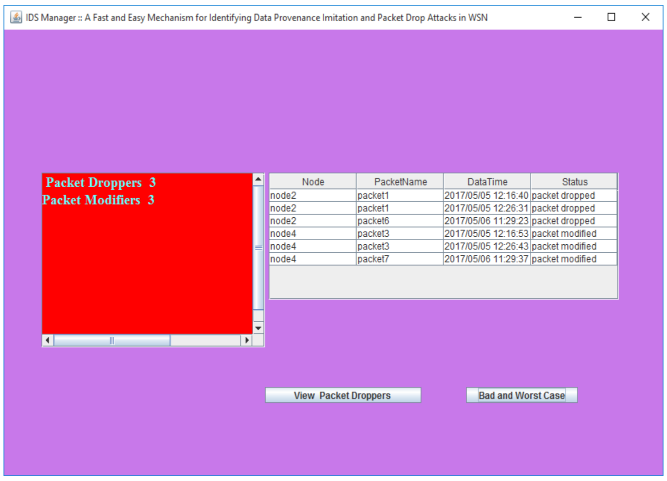
Not only the intermediate nodes, but also the BS stores and upgrades the latest packet sequence number for each data flow. Once the packets are received, the BS gets the previous packet sequence $(pSeq)$ transmitted by the source node from the packet header, gets the last packet sequence for the flow from its local storage$(pSeq_b)$, and utilizes these two sequences in the process of provenance verification and collection. Similar to the basic scheme in Section4, the BS first executes the provenance verification process upon receiving a packet. The BS knows 1) the present path of the data for the packet (decoded from the provenance of the previous packet in the flow) and 2) the following packet sequence number forwarded by each node in the path.

6. IMPLEMENTATION

The AES(Advanced Encryption Standard) algorithm at source is implemented using Java which is run on Eclipse Luna. Once the code is run, the algorithm "provenance verification " which is implemented in Java, starts comparing the bit information of the two packets to find weather any packet is dropped or modified at any nodes while passing through all the nodes. After verifying all the bits, the output is displayed in the database using MySQL and JDBC connectivity .

7. RESULTS AND DISCUSSION

We obtain the secure data transmission from source to destination .If packets are dropped or modified while in transit by the attacker ,then those packets are notified at the destination. Hence data is securely transmitted from source to destination. The bandwidth consumed is less because of in-packet bloom filters and errors occurred are also minimized by splitting the packets at the source node. Hence acting as a fast and easy mechanism for identifying packets dropped or modified in wireless sensor networks. The performance can be analyzed by the total number of packets dropped or modified. The less the number of packets dropped or modified, the more the accurate data obtained at the base station. The data obtained at the base station plays an important role for analyzing the performance. The performance can be analyzed by the following figures. The information is stored in the database. In figure 4, the IDS(intrusion detection system) displays at which node ,which packet is dropped or modified and at what time and date this malicious activity is done. And figure 5 shows the graph for the figure 4 ,representing total packets dropped and modified in terms of graph. In Figure 6, we can observe that an empty packet is obtained at the destination as this packet is dropped by an attacker . And in figure 7, the modified packet is obtained as the packet is modified by an attacker with the modified content as "qqqqqqqqqqqqq" at the first line of the packet.


IDS Manager :: A Fast and Easy Mechanism for Identifying Data Provenance Imitation and Packet Drop Attacks in WSN
View Packet Droppers
Bad and Worst Case

Packet Droppers 3
Packet Modifiers 3

Node	PacketName	DataTime	Status
node2	packet1	2017/05/05 12:16:40	packet dropped
node2	packet1	2017/05/05 12:26:31	packet dropped
node2	packet6	2017/05/06 11:29:23	packet dropped
node4	packet3	2017/05/05 12:16:53	packet modified
node4	packet3	2017/05/05 12:26:43	packet modified
node4	packet7	2017/05/06 11:29:37	packet modified

Fig-4: Displaying the total number of packets dropped and modified.

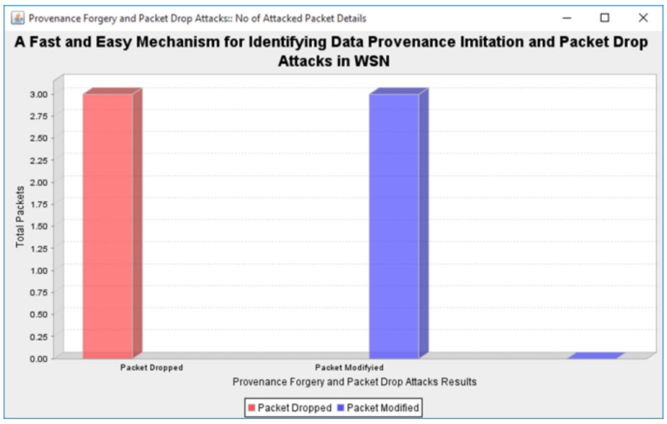
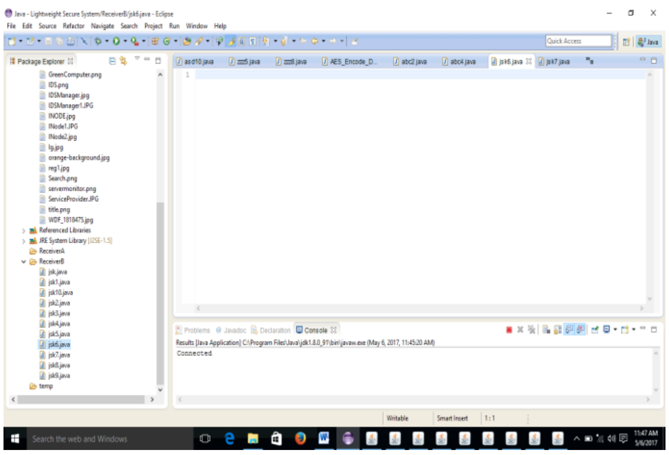
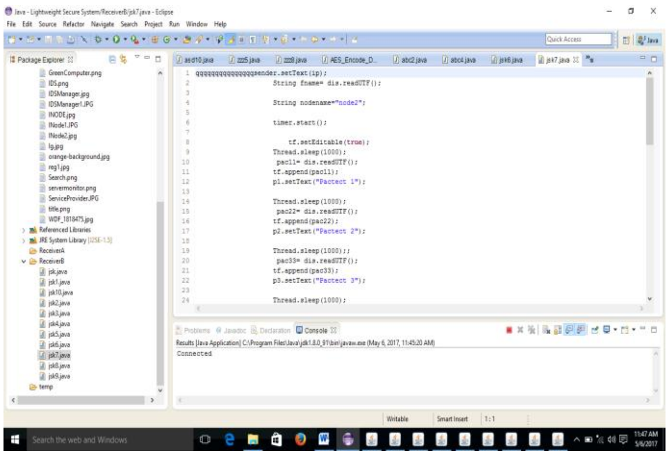


Fig-5: Representing the graph for total packets dropped and modified .


Java - Lightweight Secure System/ReceiverB/jsk6.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Problems Javadoc Declaration Console

```
Results [Java Application] C:\Program Files\Java\jdk1.8.0_91\bin\javaw.exe (May 6, 2017, 11:45:20 AM)
Connected
```

Fig-6: Displaying an empty packet which has been dropped by the third party.


Java - Lightweight Secure System/ReceiverB/jsk7.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Problems Javadoc Declaration Console

```
qqqqqqqqqqqqqqqsender.setText(ip);
                    String fname= dis.readUTF();

                    String nodename="node2";

                    timer.start();

                        tf.setEditable(true);
                    Thread.sleep(1000);
                     pac11= dis.readUTF();
                    tf.append(pac11);
                    p1.setText("Pactect 1");

                    Thread.sleep(1000);
                     pac22= dis.readUTF();
                    tf.append(pac22);
                    p2.setText("Pactect 2");

                    Thread.sleep(1000);;
                     pac33= dis.readUTF();
                    tf.append(pac33);
                    p3.setText("Pactect 3");

                    Thread.sleep(1000);
```

```
Results [Java Application] C:\Program Files\Java\jdk1.8.0_91\bin\javaw.exe (May 6, 2017, 11:45:20 AM)
Connected
```

Fig-7: Displaying the modified packet which has been modified by the third party.

8.CONCLUSION

Choosing a fast and easy mechanism for identifying packets dropped or modified, reduces the bandwidth consumption and error rates. This scheme ensures confidentiality and originality of the data. We proposed this mechanism of provenance encoding and decoding scheme based on in packet bloom filters. And also we extended the mechanism for data provenance binding, and to include packet sequence information that supports packet loss attacks. The future work can be done on images instead of only text, or both can be included at a time for verifying the provenance.

REFERENCES

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

[2] I. Foster, J. Vockler, Zhao ,M. Wilde, 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
113-127, 2003. forgery with secure provenance," in Proc. Of FAST, 2009, pp. 1–14.

[6] M. Garofalakis, J. Hellerstein, and P. Maniatis, "Proof Sketches: Verifiable In-Network Aggregation," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 84-89, 2007.

[7] H. Chan, D.Song,A. Perrig, "Secure Hierarchical In-Network Aggregation in Sensor Networks," Proc. Conf. Computer and Comm. Security (CCS), pp. 278-287, 2006.

[8] S. Roy, M. Conti, S. Setia, and S. Jajodia, "Secure Data Aggregation in Wireless Sensor Networks," IEEE Trans. Information Forensics and Security, vol. 7, no. 3, pp. 1040-1052, June 2012.