

Article

## Bloom Filter-Based Advanced Traceback Scheme in Wireless Sensor Networks

Sungmo Jung <sup>1</sup>, Jong Hyun Kim <sup>2</sup> and Seoksoo Kim <sup>1,\*</sup>

<sup>1</sup> Department of Multimedia, Hannam University, Daejeon 306-791, Korea;  
E-Mail: sungmoj@gmail.com

<sup>2</sup> Electronics and Telecommunications Research Institute, Daejeon 305-700, Korea;  
E-Mail: jhk@etri.re.kr

\* Author to whom correspondence should be addressed; E-Mail: sskim0123@naver.com;  
Tel.: +82-42-629-8336; Fax: +82-42-629-8093.

*Received: 30 August 2012; in revised form: 20 November 2012 / Accepted: 20 November 2012 /*

*Published: 23 November 2012*

---

**Abstract:** Many recent studies have focused on traceback schemes with the aim of finding the source of spoofed malicious packets and tracing the path of denial-of-service attacks. Although such schemes are academically well known, most exhibit some critical points in wireless sensor networks, which could become a significant issue as they are increasingly used for IP networks. This paper suggests an advanced traceback scheme based on existing Bloom filter methods. The proposed traceback scheme extends the basic Bloom filter design, enabling it to identify which entity added a given element, albeit with the incursion of false positives. However, our scheme allows only specific tracebacks, which can reduce the false positive rate of a node near a sink. Performance results show that the scheme can perform efficient tracebacks with very few false positives.

**Keywords:** network security; wireless sensor networks; traceback

---

### 1. Introduction

Recently, there has been growing interest in traceback schemes to find the source and path of denial-of-service (DoS) [1] attacks. Such schemes are well known from a theoretical standpoint, but they often operate in highly limited environments when applied to wireless sensor networks (WSNs) [2], which could become a significant issue as they are increasingly being used for IP networks [3]. Thus,

existing traceback schemes cannot be immediately applied to WSNs, because they require certain resources to be available.

Traceback schemes can be classified into three types: messaging [4], packet marking [5], and logging [6]. Messaging schemes have non-trivial communication overheads, and packet marking methods increase the packet size due to the bits required in the packet header. As communication is the most costly function in a sensor node (which stores information on forwarded packets in a suitable data structure), this paper suggests a logging-based traceback scheme as such methods do not add to the packet headers. In case of an attack, victims can consult upstream nodes to restructure the attack path by broadcasting the malicious packets' information in a traceback request.

Logging-based traceback schemes have been the subject of relatively few studies. In this paper, we base our research on the CoordinAted Packet TRAcback (CAPTRA) scheme, which uses a Bloom filter [7] to store traffic logs. A Bloom filter is represented by bit vectors with randomized data structures [8]. The Bloom filter performs membership queries, determining whether an entity (*i.e.*, a data packet under traceback) is part of a data set, albeit with controllable false positives. The Bloom filter and its variants are of prime importance, and they are heavily used in various distributed systems [9].

In its basic form, the Bloom filter is simply used for membership queries—it does not store any information about the entities in question. Therefore, a traceback request is generally broadcast to neighboring nodes, whereupon the chances of generating a false query increase due to the generation of false positives by the Bloom filter. To overcome this defect, this paper suggests an advanced traceback scheme based on an existing Bloom filter for WSNs. First, we add a support-directed query to reduce the number of messages generated by the traceback procedure. Second, we propose an algorithm that reduces the false positive rate at nodes near a sink by using multiple Advanced Tagged Bloom Filters (ATBFs). The suggested scheme can provide effective hop-by-hop traceback to a single attacker in WSN environments.

## 2. Related Research

### 2.1. Coordinated Packet Traceback

The CAPTRA [10] mechanism for WSNs takes advantage of the broadcast protocol for packet transmission. By remembering, and later retrieving, the packets in multi-dimensional Bloom filters that are distributed via overhearing sensors, CAPTRA identifies the packet transfer path using a series of REQUEST-VERDICT-CONFESS message exchanges between the forwarding and overhearing nodes. CAPTRA imposes only a small memory footprint on the sensors due to its usage of Bloom filters, and it allows the sensors to asynchronously refresh the Bloom filters so that the network traffic is continuously monitored.

### 2.2. Contract-Based Traceback

In contract-based traceback (CTrace) [11], each node maintains information about its neighbors within a radius of  $R$  hops. The node at  $R$  hops distance is called the contact. With certain probability, the nodes mark identification information on arriving packets, allowing each node to construct a partial attack path back to one of its contacts by collecting a few packets. CTrace builds the whole attack path

using a series of REQ and PATHSEG message exchanges between a sensor node and its contacts. Only a small number of packets are required to track down the source of false data.

### 2.3. Resource-Efficient IP Traceback

IP traceback in a mobile *Ad Hoc* network (MANET) is more difficult than in traditional wired networks because of the dynamically changing topologies and limited resources. Research into IP traceback schemes for MANETs largely employs the approaches used for wired networks, and thus, faces difficulties in overcoming the above-mentioned topology and resource limitation problems. Kim *et al.* [12] overcame the topology changing problem with a novel logging technique based on Bloom filters, and they used time-tagging of the Bloom filter entry and dealt with the resource problem. The technique has been tested through extensive simulation, and the results show that an attacker can be tracked with a reasonably sized Bloom filter as long as the collision rate in the filter is kept below 0.07%.

## 3. Suggested Scheme

### 3.1. Traceback Queries

This paper suggests a traceback scheme that extends the basic Bloom filter design, enabling it to identify which entity added a given element, albeit with the incursion of false positives, because we have a collision at a location in our filter among two or more flows with different state values. The idea for this design came from existing Tagged Bloom filters, although the definition and usage is different. For this reason, we call the suggested scheme an Advanced Tagged Bloom Filter (ATBF).

In ATBF, all feasible traceback nodes maintain an Advanced Tag Table (ATT). The ATT stores the addresses of neighboring nodes and assigns long Advanced Tags (ATs) of  $i$ -bits to each node. Under IEEE 802.15.4, 16- and 64-bit addresses can easily store the addresses and ATs of neighboring nodes. The value of  $i$  is given by the following formula:

$$i = \lceil \log_2(l_{ATT} + 1) \rceil \quad (1)$$

where  $l_{ATT}$  is the maximum number of entries in the ATT, which is set during the deployment phase. The value of  $l_{ATT}$  depends on the maximum number of neighbors that a node can have. In general,  $l_{ATT} = 7$  is a reasonable upper bound for many networks. In this paper, however, a value of  $l_{ATT} = 15$  will be used for densely deployed networks.

We consider a two-dimensional array of  $m \times i$ , instead of taking a single-bit array of length  $m$ , so that the AT assigned by the ATT can be stored. For each incoming packet, the ATT is checked to ascertain the AT corresponding to the node from which the packet was received. If the address of the forwarding node does not exist in the ATT, a new AT is assigned to that node. The payload is then passed to the hash function  $k$  that results in  $k$  indexes, similar to the normal Bloom filter operation. The AT assigned by the ATT is inserted into  $k$  indexes instead of setting only one bit. The payload of the packet helps determine the location of a given payload in the ATBF, whereas the source of the packet determines what should be written at the given location. Membership queries can be carried out by passing the payload to  $k$  hash functions and confirming the corresponding position in the  $m \times i$

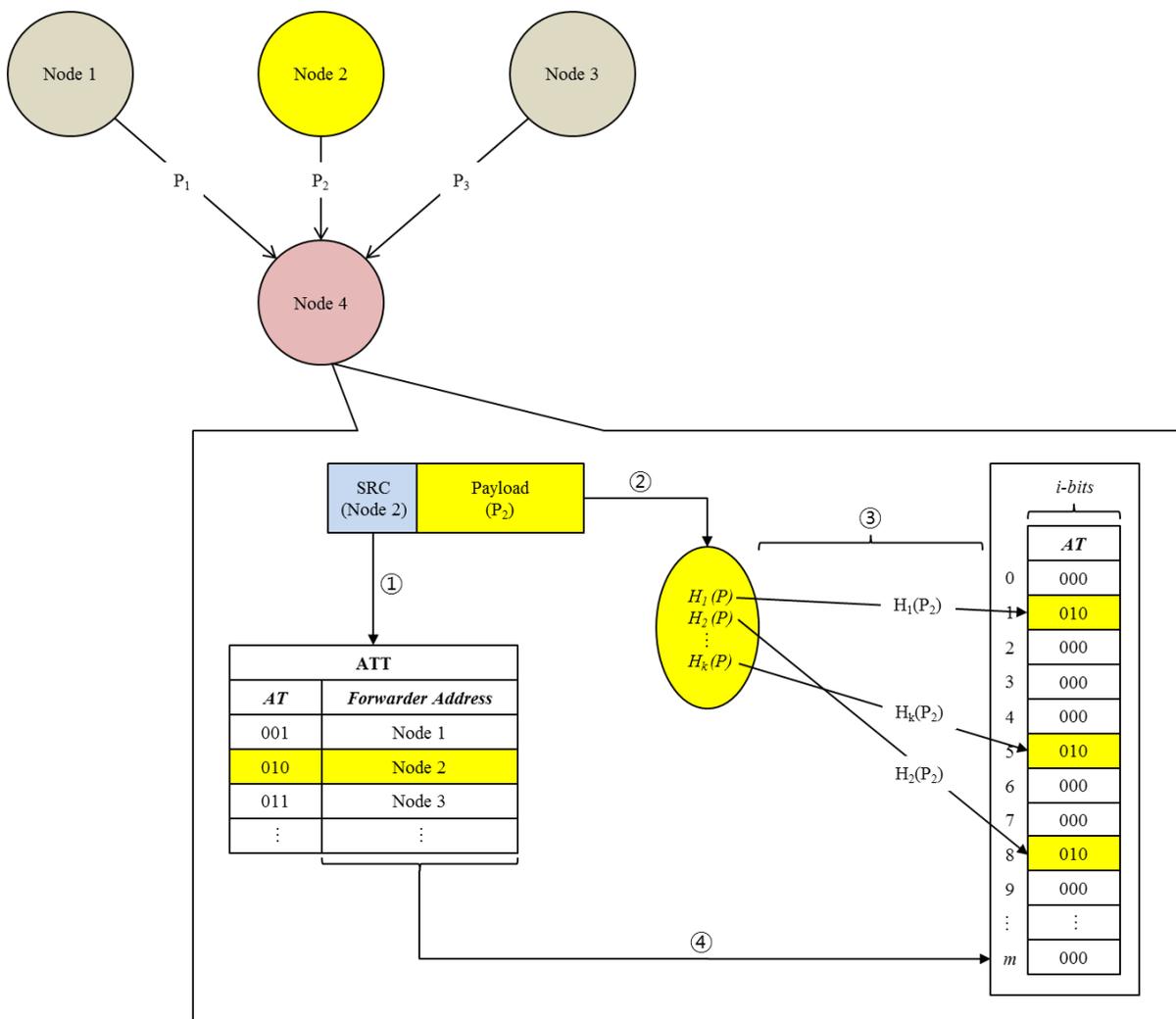
array. If any cell position is found to be 0, then the payload in question has not been stored. If all the values are non-zero, then the node that forwarded the packet in question can be identified by the following formula:

$$\vec{AT} = Mode(at[H_1(P)], at[H_2(P)], \dots, at[H_k(P)]) \tag{2}$$

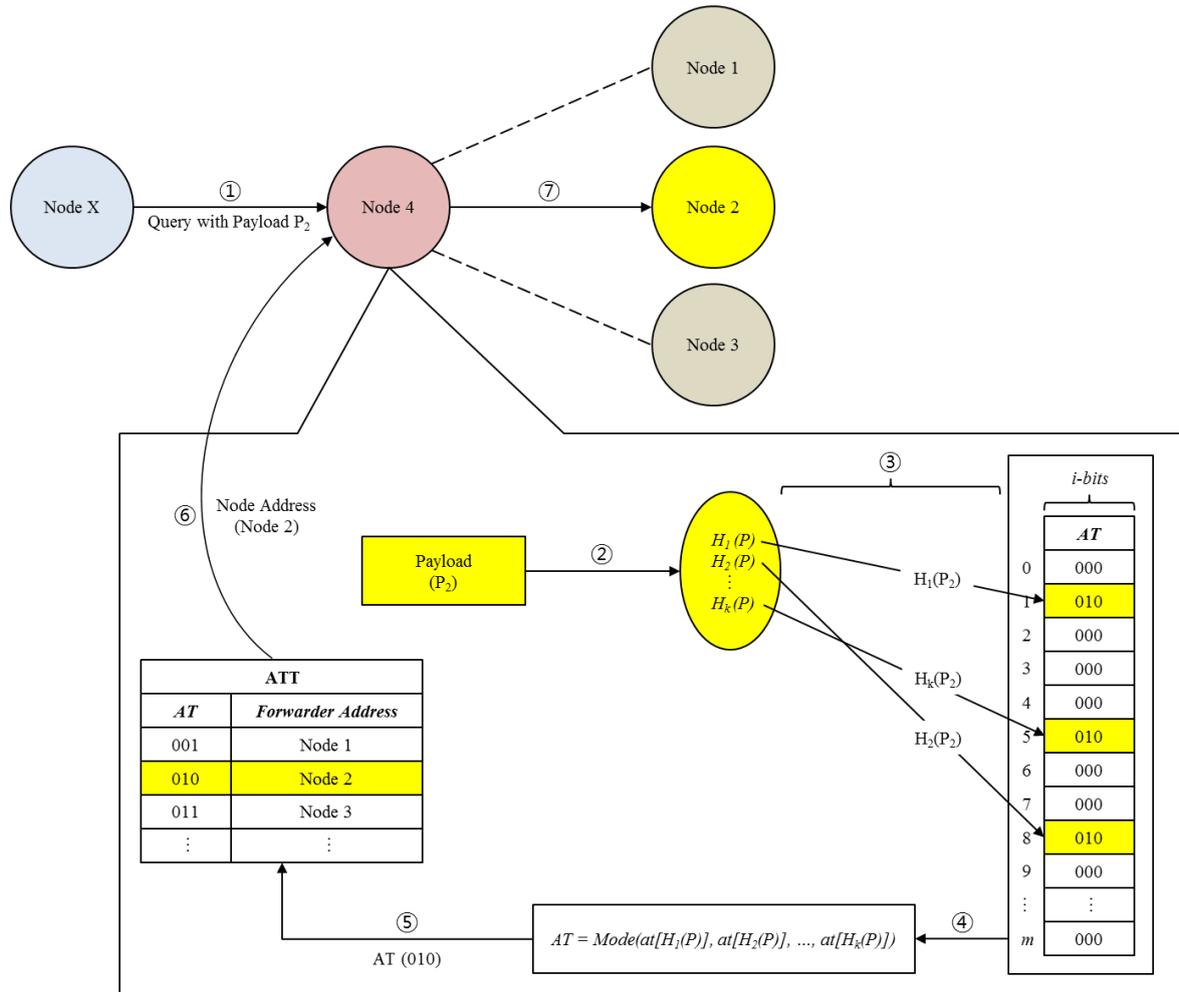
where  $P$  is the payload,  $at[H(P)]$  returns the AT at location  $H(P)$ , and  $\vec{AT}$  is a vector of ATs pointing to the corresponding addresses of neighbors in the ATT when the *Mode* function returns the most frequently occurring value. The traceback query is then forwarded to the nodes returned by the ATT.

In Figure 1, the source of the packet is checked against Node 4’s ATT, which assigns the AT 010. Thereafter, payload  $P_2$  is passed to the  $k$  hash functions, and the corresponding cell positions are set to 010. In Figure 2, the corresponding cell positions of the payload in the query are checked and passed from  $k$  hash functions. In the figure, all cell positions return 010. Hence, 010 is passed to the ATT, which consequently returns Node 2 as a possible forwarder of the packet.

**Figure 1.** Logging phase of the suggested ATBF scheme. Node 2 sends data as payload  $P_2$  to Node 4.



**Figure 2.** Query phase of the suggested ATBF scheme. Traceback commences when Node X sends a query with payload  $P_2$ .



### 3.2. Multiple ATBFs

Traffic generally flows from sensors to sinks in a sensor network. Therefore, nodes nearer to a sink exhibit higher traffic loads than the others. If nodes nearer to a sink have a larger filter size than the others, the false positive rate can be reduced. For this reason, we suggest further improvements to our scheme by allowing variable size ATBFs at different nodes.

First, the hop distance of certain nodes from the sink must be known in order to dynamically control the ATBF size. Second, suitable values for the length of the ATBF ( $m$ ), number of inserted packets ( $n$ ), and number of hash functions ( $k$ ) must be chosen according to the network size. This study utilizes the hop-count value provided by routing protocols to determine the distance to the nearest sink. Routing protocols that handle hop-count values can support a multiple-ATBF (M-ATBF) scheme. For example, in the *Ad Hoc* On-demand Distance Vector (AODV) routing protocol, a Route Reply (RREP) message is used to return the route, which reflects the distance of a node from a sink. Therefore, this paper uses the hop count available in the RREP to estimate the ATBF size. If the routing protocol cannot provide a hop-count value, the sink sends a message with the hop-count value 1. When a sensor node receives this message, it collects and updates the hop-count value before forwarding the message to upstream nodes.

Depending on the position of a node, parameters of the ATBF can be scaled up or scaled down. Therefore, this study attempts to increase the capacity of the ATBF (*i.e.*,  $n$ ), which is proportional to its length (*i.e.*,  $m$ ). Additionally, the dynamic control capability of  $m$  requires a set of hash functions with a higher range so that the input entities can be mapped to a larger data set. Substituting the set of hash functions is not a feasible solution as this is very complex. Although we can replace the set of hash functions, the values stored in the ATBF cannot be verified in the query phase if the original set of hash functions has been replaced by a new set. For this reason, controlling  $k$  or  $m$  is not feasible. To overcome this problem, we instead utilize M-ATBFs to change the ATBF size. Figure 3 illustrates this process.

**Figure 3.** Final architecture of a traceback scheme with M-ATBFs. M-ATBF is used to change the ATBF size.

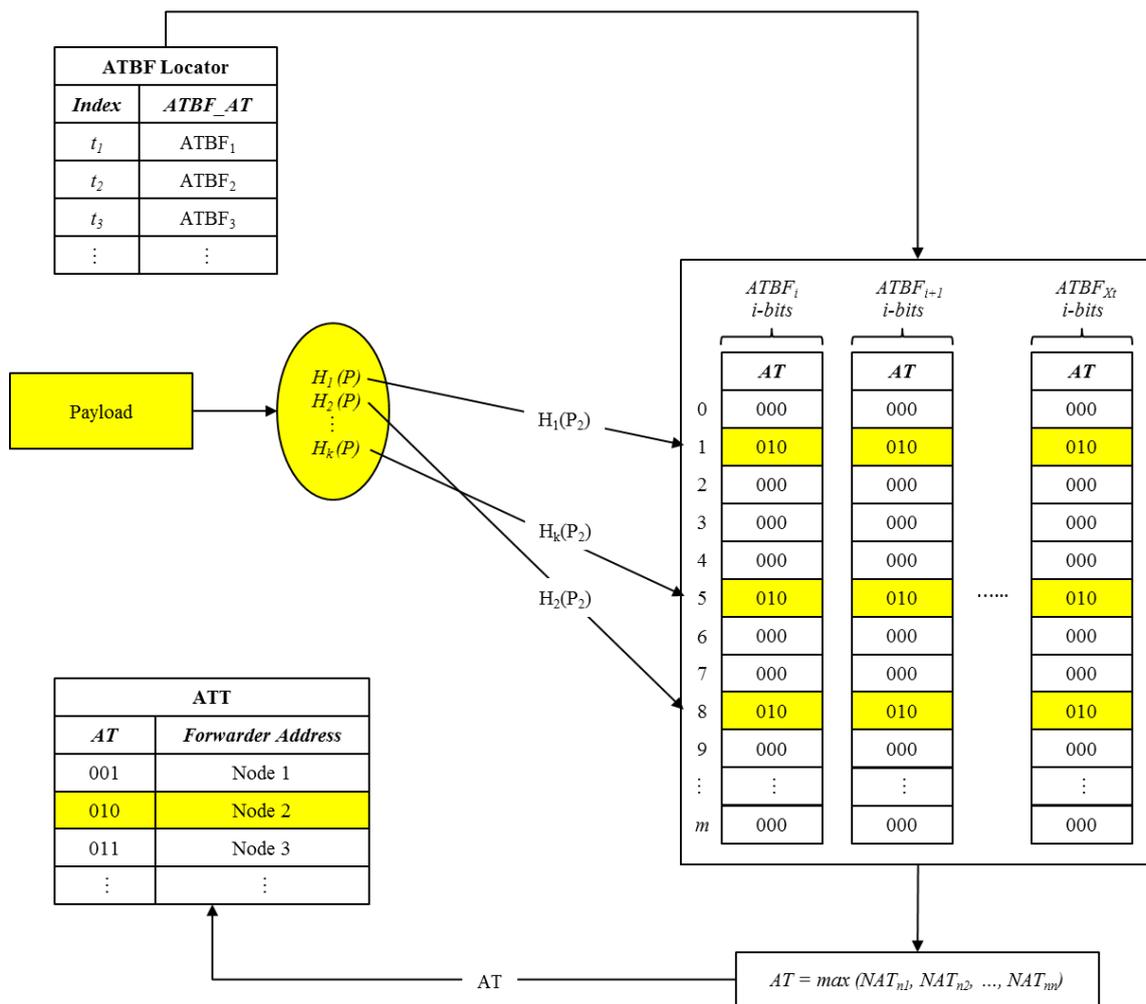


Figure 4 shows how the hop-count value can be used to obtain the value of  $X_t$ , which represents the number of ATBFs. The algorithm for this procedure is as follows:

- (1) Node A sends a Route Request (RREQ) message to get routing information for the sink.
- (2) The sink replies to this RREQ with an RREP message.
- (3) The RREP message is captured and sent to an intermediate node.
- (4) The intermediate node calculates the value of  $X_t$  using the following formula:

$$X_t = \max (MAX - hop\_count, X_{t-1})$$

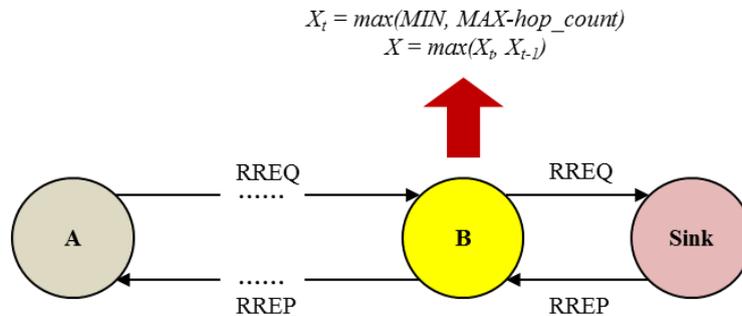
where:

$$X_0 = MIN$$

(3)

(5) Consequently, the value of  $X_t$  is used to create an M-ATBF in Algorithm 1.

**Figure 4.** Calculation of  $X_t$  from the hop-count value. The hop-count value can be used to obtain the value of  $X_t$ , which represents the length of the ATBF.



**Algorithm 1.** Pseudo code to create an M-ATBF.

```

Data :  $X_t$  is the number of ATBFs
Function : The intermediate node calculates the value of  $X_t$ 
#i ← intermediate node
#rep.msg.RREP ← msg.RREP with msg.RREQ
DO (hop.count = 0)
    // Node A sends msg.RREQ to get routing information for the sink
    IF (sink != null)
        send msg.RREQ
        IF (get msg.RREQ)
            reply rep.msg.RREP ← msg.RREQ
            // rep.msg.RREP is captured and send to a intermediate node
        ENDIF
    ELSE
        BREAK
    ENDIF
    // M-ATBF creation
    IF (i get rep.msg.RREP) THEN
         $X_t = \max (MAX - hop\_count, X_{t-1})$  where  $X_0 = MIN$ 
        create M-ATBF ←  $X_t$ 
    ENDIF
WHILE (own.hop.count = ∞)

```

Our algorithm ensures that nodes nearer to the sink would have M-ATBF ranges from  $ATBF_1$  to  $ATBF_{x_t}$ . Once an ATBF, say  $ATBF_i$ , reaches its optimal capacity, it is archived and replaced by  $ATBF_{i+1}$  takes place of by optimal capacity. This means that the false positive rate of the ATBF is less than 0.05. The MIN and MAX values are used to prevent the hop-count value from being spoofed in the RREP messages. This is important because (for example) if a malicious node between the sender and receiver modifies the hop-count value, it may result in too many ATBFs, or even none at all. Therefore, the upper and lower bounds can be set for the hop count according to network requirements. The suggested scheme only stores unique packets to ensure that a simple search of archived ATBFs is feasible. When all ATBFs reach their optimum capacity, the first ATBF is reset to 0.

### 3.3. Hash Functions

In the proposed scheme, the hash functions included in the ATBF must be lightweight because our algorithm cannot afford strong cryptography. In addition, the set  $S$  of hash functions at one relay node must not be the same as set  $S'$  at a neighboring relay node (*i.e.*,  $Pr [S = S']$  should be very low). This ensures that the probability of a false positive at one relay node is independent of false positives at the other relay nodes, and thus, any traceback request generated by a false positive will slowly disappear as it progresses.

## 4. Performance Analysis

This paper considers static WSNs that may be subjected to a DoS attack by an individual or group of attackers. In the case of spoofing, it is difficult to detect the complete attack path, but an attacker's one-hop location can be found. In addition, attackers may attempt to fill the Bloom filters with false data so that any information about malicious packets is overwritten. However, the suggested scheme can find the attacker because benign nodes do not flood the network with unnecessary messages, which would affect the computation of *Mode* in the membership query phase. Similar to Bloom filters, the performance of the ATBF depends mainly on the values of  $m$ ,  $n$ , and  $k$  mostly. We now compare the performance of the proposed scheme with the results presented in [13]. For this reason, we first state the false positive rate given by Bloom filters:

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \quad (4)$$

The above equation can be used to ascertain when  $ATBF_t$  should be replaced by  $ATBF_{t+1}$ . For example, the suggested scheme can store about 164 unique packets in an ATBF for a false positive rate of 0.5 and with  $m = 1,024$  and  $k = 4$ . Therefore,  $ATBF_t$  is replaced by  $ATBF_{t+1}$  after inserting 164 unique packets.

The experimental results for hundreds of nodes would be interesting to see the scalability. In this paper, however, we illustrate the conditions with node numbers 2 to 5, because we only need to prove that the M-ATBF scheme is superior to the existing traceback scheme.

The use of the M-ATBF requires more memory in the sensor nodes, as shown in Figure 5. In this figure, with a binary tree, the scheme uses more memory than that used with a higher number of child

nodes as this would increase the number of hops from a leaf node to a sink. Therefore, there is a high false positive rate compared to the M-ATBF.

**Figure 5.** Memory requirement of the existing scheme and the M-ATBF traceback scheme. Each solid curve shows the memory requirement for a specific form of tree. The dotted line shows the memory requirement of the existing scheme. In this case, the memory is static and set to 128 bytes.

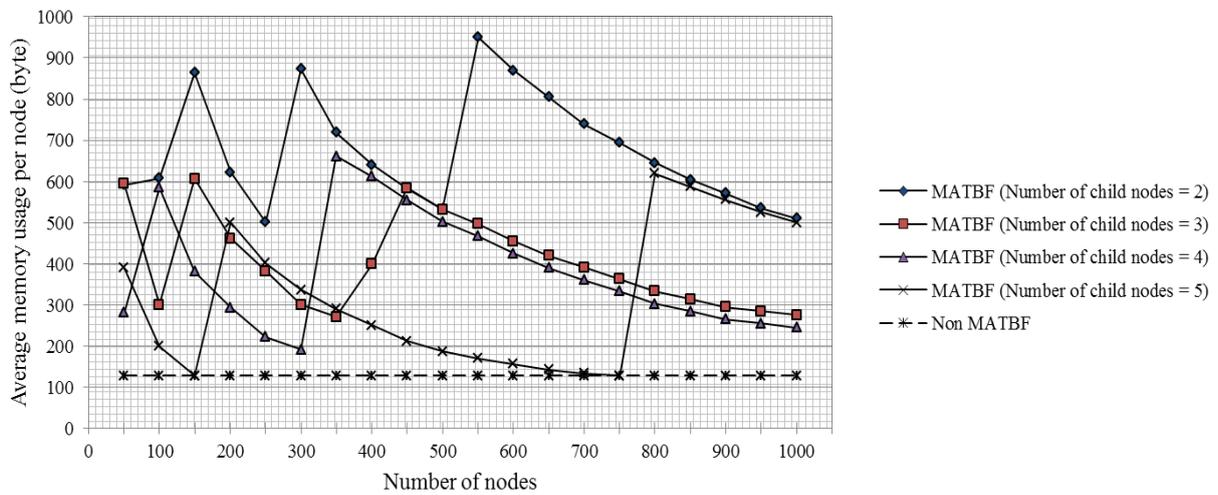
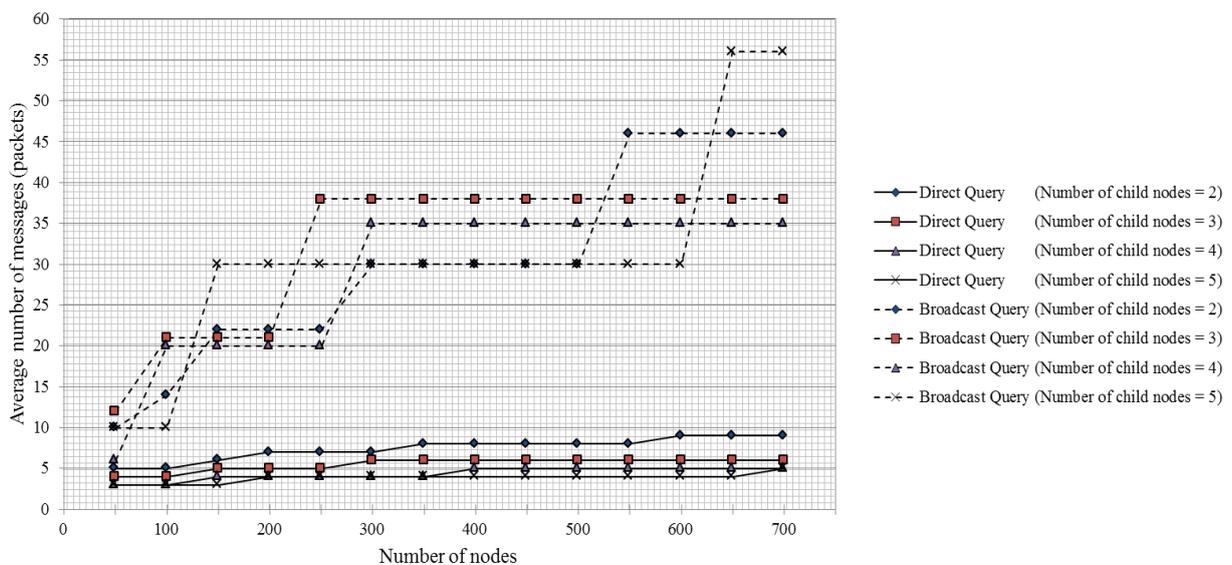


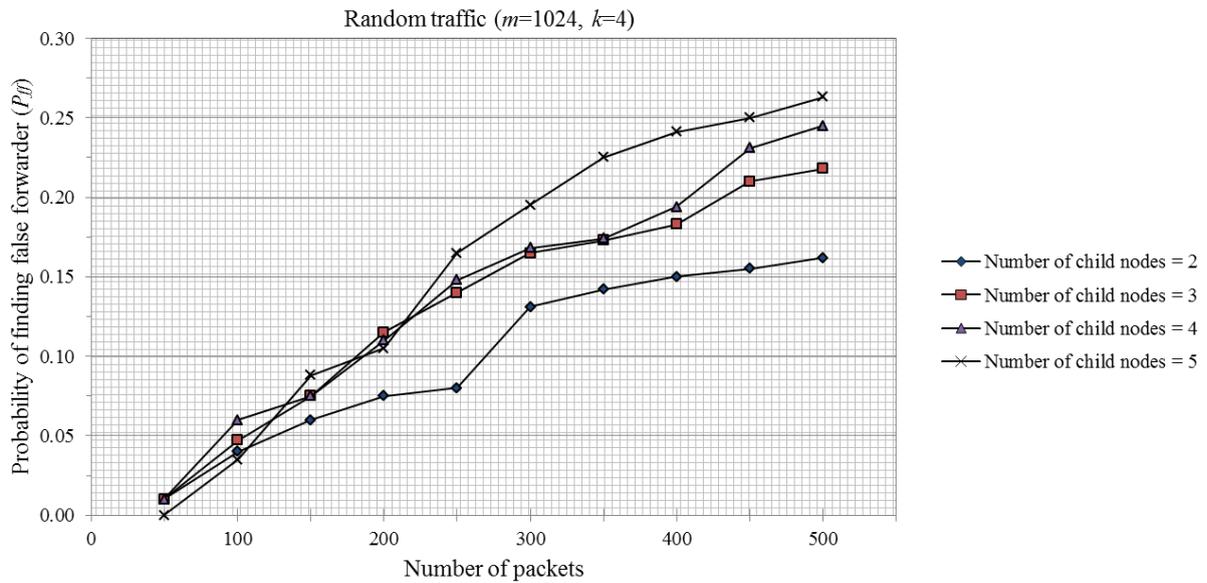
Figure 6 shows the number of messages required to find the correct path in the query phase.

**Figure 6.** Number of messages required in the broadcast and directed query schemes, with respect to different network topologies.



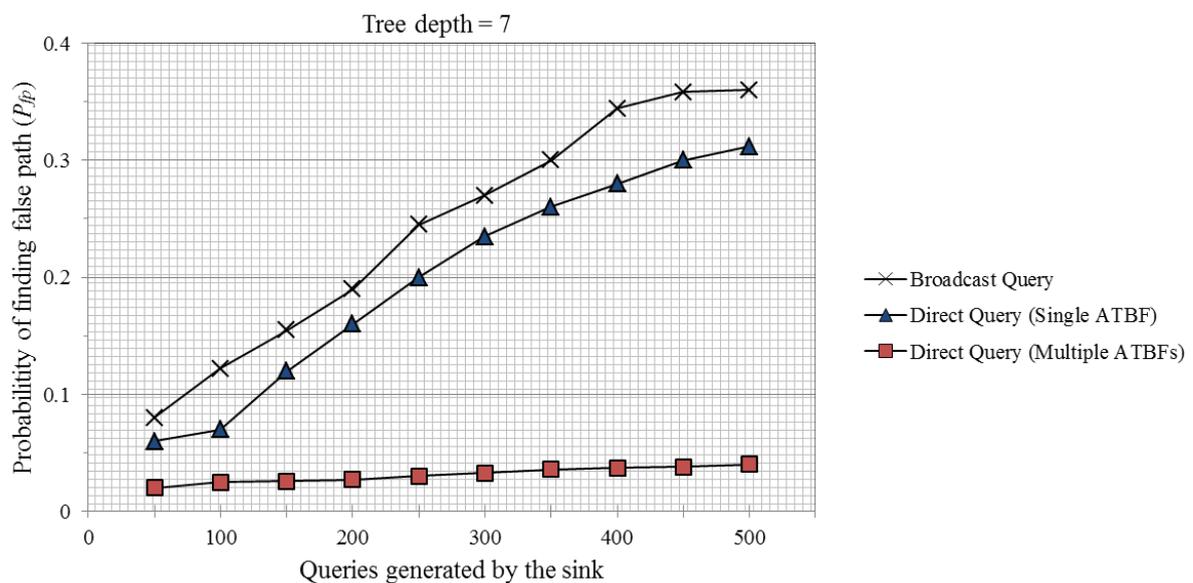
The number of messages in the broadcast query scheme is higher than that in the directed query scheme because more than one possible path can be found in the broadcast scheme. In Figure 7, the number of child nodes affects the probability slightly because  $P_{ff}$  depends on the number of packets inserted by a specific child node.

**Figure 7.** Probability of finding a false forwarder as the number of packets inserted in the Bloom filters increases. It can be seen that the probability of finding a false forwarder ( $P_{ff}$ ) of an inserted packet in the ATBF increases with the number of child nodes ( $N$ ) for random packet generation.



The results shown in Figure 8 were obtained by constructing a tree of sensing nodes on a simulator. In this simulation, the total memory size (786,432 bits) of the M-ATBF is equal to that of the S-ATBF and the broadcast query scheme. The average number of child nodes in each tree was varied, and only leaf nodes were allowed to randomly generate packets. The results reported in this section indicate that the M-ATBF traceback scheme is superior to the existing traceback scheme.

**Figure 8.** Probability of finding a false path as the number of queries generated by the victims increase. The probability of finding a false path ( $P_{fp}$ ) with the directed and broadcast query schemes by using single and multiple ATBFs is shown.



In the evaluation of the memory consumption, however, the broadcast query scheme expends 516,096 bits, the S-ATBF expends 524,288 bits, and the M-ATBF expends 753,122 bits.

## 5. Conclusions

In this paper, we have proposed an advanced traceback scheme based on a Bloom filter technique. To perform a specific traceback, we have added support-directed queries and introduced a method for reducing the false positive rate. Our experimental results show that the suggested scheme results in a lower number of traceback messages and a lower probability of finding a false path compared to the existing traceback schemes. However, the proposed scheme incurs higher memory usage. In future work, therefore, we will attempt to modify this traceback method so that it requires smaller memory.

## Acknowledgments

This paper has been supported by the Software R&D Program of KCC (10914-06002, Development of Global Collaborative Integrated Security Control Systems).

## References

1. Wood, A.D.; Stankovic, J.A. Denial of service in sensor networks. *IEEE Comput.* **2002**, *35*, 54–62.
2. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330.
3. Pathan, A.S.K.; Lee, H.W.; Hong, C.S. Security in wireless sensor networks: Issues and challenges. *Int. Conf. Adv. Commun. Technol.* **2006**, *2*, 1043–1048.
4. Baba, T.; Matsuda, S. Tracing network attacks to their sources. *IEEE Internet Comput.* **2002**, *6*, 20–26.
5. Park, K.; Lee, H. On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. *IEEE Comput. Commun. Soc.* **2001**, *1*, 338–347.
6. Al-Duwairi, B.; Govindarasu, M. Novel hybrid schemes employing packet marking and logging for IP traceback. *IEEE Trans. Parallel Distr. Syst.* **2006**, *17*, 403–418.
7. Mullin, J.K. A second look at Bloom filters. *Commun. ACM* **1983**, *26*, 570–571.
8. Sen, J. A survey on wireless sensor network security. *Int. J. Comm. Network. Inform. Secur.* **2009**, *1*, 55–78.
9. Flavio, B.; Michael, M.; Rina, P.; Sushil, S.; George, V. Beyond bloom filters: From approximate membership checks to approximate state machines. *ACM SIGCOMM Comput. Commun. Rev.* **2006**, *36*, 315–326.
10. Sy, D.; Bao, L. CAPTRA: Coordinated Packet Traceback. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, Nashville, TN, USA, 19–21 April 2006; pp. 152–159.
11. Zhang, Q.; Zhou, X.; Yang, F.; Li, X. Contact-Based Traceback in Wireless Sensor Networks. In *Proceedings of the Wireless Communications, Networking and Mobile Computing*, Shanghai, China, 21–25 September 2007; pp. 2487–2490.

12. Kim, I.Y.; Kim, K.C. A resource-efficient IP traceback technique for mobile *Ad Hoc* networks based on time-tagged Bloom filter. *Int. Conf. Convergence Hybrid Inform. Technol.* **2008**, *2*, 549–554.
13. Broder, A.; Mitzenmacher, M. Network applications of bloom filters: A survey. *Internet Math.* **2004**, *1*, 485–509.

© 2012 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).