

# CAPTRA: Coordinated Packet Traceback

Denh Sy and Lichun Bao  
Bren School of Information and Computer Sciences  
University of California, Irvine, CA 92697

{dsy,lbao}@ics.uci.edu

## ABSTRACT

Network-based attacks can be either persistent or sporadic. Persistent attack flows can be relatively easy to trace by mechanisms such as probabilistic packet marking, traffic logging, data mining etc. Sporadic attacks are sometimes easily detected by the Intrusion Detection Systems (IDSs) at the victims, but are hard to trace back to the attack origins. We propose CAPTRA, a CoordinAted Packet TRAcEback mechanism, for wireless sensor networks (WSNs) that takes advantage of the broadcasting nature of the packet transmissions. By remembering packets in multi-dimensional Bloom filters distributed in overhearing sensors and later retrieving the information, CAPTRA identifies the path of the packet transfers using a series of REQUEST-VERDICT-CONFESS message exchanges between the forwarding and overhearing nodes. CAPTRA requires only small memory footprint on the sensors due to the usage of Bloom filters, and allows sensors to asynchronously refresh the Bloom filters so that the network traffic is continuously monitored. CAPTRA is simulated using J-Sim, and a few key parameters are tuned for the best tracing performance.

**Categories and Subject Descriptors:** H.3.4 [Information Storage and Retrieval]: Systems and Software – *Information networks*; C.2.2 [Computer-Communication Networks]: Network Protocols – *Applications*.

**General Terms:** Algorithms.

**Keywords:** Wireless sensor networks, packet traceback, Bloom filter.

## 1. INTRODUCTION

Wireless sensor networks (WSNs) are networks of a large number of small wireless devices, which collaborate to monitor environments and report sensing data via wireless channels. Wireless sensors are very limited in their computing, communication and storage capabilities due to extremely constrained battery power supply and costs. For instance,

the typical Crossbow MICA mote MPR300CB [7] has a low-speed 4MHz processor equipped with only 128KB flash, 4KB SRAM and 4KB EEPROM. It has a maximal data rate of 40kbps and a transmission range of about 100 feet, powered by two AA batteries. In addition, a wireless sensor network is usually deployed with high density in order to improve a WSN's reliability and lifetime.

Security is important for many sensor network applications. Besides robust networking protocol design requirements to combat security exploitations, application-level intrusion detection and traceback mechanisms are necessary in many situations. A particularly harmful attack against sensor and ad hoc networks is known as the Sybil attack [10], where a node illegitimately claims multiple identities. Without bulletproof hardwares, packet traceback seems to be the only mechanism left to track down Sybil attackers.

Unfortunately, a lot of the existing research in the packet traceback techniques and architectures were designed for Internet packet traceback, and are not viable in WSNs because of the characteristics of WSNs. First of all, human intervention or in-network intrusion detection are not possible to prevent attacks due to the wide deployment nature for distributed processing, and the limited power and silicon supply. Secondly, the wireless transmission medium is openly accessible to anyone in the network adjacency, thus making the network vulnerable to malicious packet injections. Third, sensor networks are capable of in-network data processing, thus eliminating the need for accurate topology maintenance, which makes address-based packet filtering and location inference impossible.

Fortunately, what makes WSNs vulnerable is also the strength of WSNs for security functions. We propose CAPTRA, a CoordinAted Packet TRAcEback protocol, for WSNs that takes advantage of the open transmission and in-network processing power of the WSNs, and uses multi-dimensional Bloom filters for packet traceback purposes. In CAPTRA, each sensor allocates a small memory block for the Bloom filter. When a packet of interested categories traverses the network, each forwarding sensor records the packet in its Bloom filter, so do the overhearing sensors. Different from previous approaches that build Bloom filter hash functions based on the packet information alone, the forwarding node ID is taken as well an input to the hash function in the Bloom filter in CAPTRA. During traceback operations, the predecessor on the packet source route is determined by a collaborative mechanism by summarizing all possible packet witnesses that are collected from the overhearing sensors. If the witness quorum reaches certain threshold, traceback op-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.  
Copyright 2006 ACM 1-59593-334-4/06/0004 ...\$5.00.

erations are recursively continued toward the true origin of the packet hop-by-hop. Three messages, REQUEST, VERDICT and CONFESS, are used for the traceback coordination, witness collection, and source route construction processes, respectively.

The rest of the paper is organized as follows. Section 2 reviews the current techniques and architectures for intrusion detection and packet traceback in the Internet. Section 3 briefly describes the widely used Bloom filter, and introduces a new perspective regarding the dimensions of Bloom filter, which we call space-time Bloom filter (STBF), a special case of multi-dimensional hash table (MDHT), with the Bloom filter as a special case under this perspective. Section 4 discusses the methods of applying STBF for packet traceback. Section 5 describes our coordinated packet traceback protocol in details, and Section 6 presents the evaluation results of the protocol in a WSN using simulations. Section 7 concludes the paper.

## 2. RELATED WORK

Network attacks can be either persistent or sporadic [28]. In persistent attacks, the offenders must frequently launch attack packets to bombard the victims. Whereas in sporadic attacks, a single packet can render havoc at the potential victim, such as the WinNuke, Ping of Death and Teardrop attacks.

Intrusion Detection Systems (IDSs) use *attack signature* or *pattern* to help distinguish malicious packets from normal traffic. At the very least, an attack signature is defined by the IP address or address range of the entity that is being attacked. A variety of methods for IDSs were discussed in [26] that can help detect if received packets have spoofed source addresses.

After malicious attacks are detected, the subsequent traceback to the attack origins requires the network participate in the pre-attack tracking and post-attack tracing operations, in which *packet tracking* refers to the recording of packets or packet flows when the packets are forwarded from their sources to their destinations, and *packet tracing* refers to the operations to find out the *source route* of a packet, which is the sequence of nodes that have forwarded the packet or the flow. This term is slightly different from what is commonly known by the source routing in IP forwarding.

Source route identification problem is commonly referred to as *network traceback*, or *IP packet traceback* in the Internet arena. Two network tracing problems are currently being studied: “single-domain IP traceback” and “traceback across stepping-stones” (or a “connection chain”). Traceback across stepping-stones is to identify the origin of an anonymous attacker through a chain of connections before the attacker interacts with a victim host. The Decentralized Source Identification System (DECIDUOUS) uses IPsec security associations (SAs) and authentication headers to dynamically deploy secure authentication tunnels, and traces back to the attacks origins [5, 6]. The premise of DECIDUOUS is that if an attack packet has been correctly authenticated by a certain router, the attack packet must have been transmitted through that router. It utilizes IPsec security associations to dynamically deploy secure authentication tunnels in order to further trace down the possible attackers’ locations. Other approaches such as correlating the inter-packet delay were also proposed [30].

Many approaches have been proposed to trace single-domain

IP traceback. We briefly go over some of these approaches.

The IP marking approaches enable routers to probabilistically mark packets with partial path information, and try to reconstruct the complete path from the packets with the markings [12, 17, 22, 24, 29]. A more explicit approach using IP marking is to add the IP address of the router to the IP header by turning on the *Record Route* option of the IP header [9]. An algebraic approach is proposed to transform the IP traceback problem into a polynomial reconstruction problem, and uses techniques from algebraic coding theory to recover the true origin of spoofed IP packets by having routers to embed information randomly into packets [8]. This is similar to the technique used in [22], which uses algebraic techniques to encode the path information as points on polynomials, and then reconstruct these polynomials at the victim. Li *et al.* proposed a hybrid approach to selective mark packets for tracking in the intermediate routers using Bloom filters [14].

ICMP traceback (iTrace) proposes to introduce a new ICMP message (or an iTrace Traceback message) so that routers can, with a low probability, generate iTrace messages to help the victim or its upstream ISP to identify the source of spoofed IP packets [2]. With enough ICMP Traceback messages from enough routers along the path, the traffic source and path can be determined. An intention-driven iTrace is also introduced to reduce unnecessary iTrace messages to improve the performance of iTrace systems [15].

An IP overlay network-based traceback system, named CenterTrack, selectively reroutes interesting IP packets directly from edge routers to special tracing routers, and the hop-by-hop input-debugging is then used [25]. *Input debugging* refers to the diagnostic features required to recursively determine from which adjacency a packet arrived that matches an attack signature on an individual router, until the edge of the network is reached and the edge ingress adjacency is identified.

Snoeren *et al.* proposed an architecture, Source Path Isolation Engine (SPIE), that integrates the IDS and single-packet traceback engines [23] to reconstruct the identify the attack graph. According to SPIE, once the IDS detects an abnormal attack event, the attack packet is fed into the traceback manager to generate a traceback request, which is sent to multiple network agents for constructing the regional attack graph based on the attack packet. Afterward, the regional attack graphs are assembled into a complete attack graph at the traceback manager, and fed back the IDS. SPIE is based on Bloom filter [3] for packet logging purposes [21].

To our knowledge, packet traceback in WSNs is among the few with little research so far.

## 3. MULTI-DIMENSIONAL HASH TABLE

### 3.1 Bloom Filter and Its Variants

A Bloom filter is a space-efficient probabilistic data structure that is used to test whether or not an element is a member of a set [3]. Bloom filters are used in myriad of applications. Wherever a list or set is used, and space is a consideration, a Bloom filter is commonly considered [4].

Fig. 1(a) illustrates how the Bloom filter is updated when a new element is inserted into the set, where  $k$  hash functions based on the same input string produce  $k$  index keys to update the single Bloom filter. Traditional hash table is a

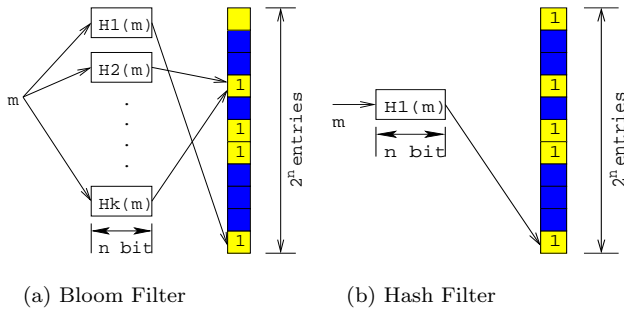


Figure 1: The Bloom Filter

special case of Bloom filter where  $k = 1$ , shown in Fig. 1(a). Usually, elements are only added to the set, but not removed. Given a finite set, *false positive* judgment of the membership are possible, but *false negative* judgment are not in the traditional Bloom filters without refreshing. When the more elements are added to the set, the probability of false positive becomes greater.

Several variants of Bloom filters have been proposed. Attenuated Bloom filters [18] use arrays of Bloom filters to store shortest path distance information. Spectral Bloom filters [20] extend the data structure to support estimates of frequencies. In Counting Bloom Filters [11] each entry in the filter need not be a single bit but rather a small counter. Insertions and deletions to the filter increment or decrement the counters respectively. When the filter is intended to be passed as a message, compressed Bloom filters [16] may be used, where parameters can be adjusted to the desired tradeoff between size and false-positive rate. Space-code Bloom filter provides frequency estimation of an element by probabilistically filling up multiple normal Bloom filters, from which to statistically infer the frequency of the element [13].

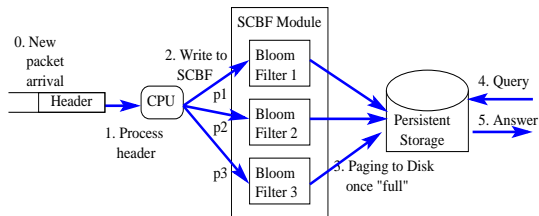


Figure 2: Space-Code Bloom Filter for Traffic Measurement

Fig. 2 illustrates the space-code Bloom filter (SCBF) for accounting flow volume, which is represented by the number of packet in the flow. A randomly chosen Bloom filter of the SCBF module takes the flow ID (source and destination IP addresses and port numbers) as the inputs, and record the membership in the table. When the Bloom filters in the SCBF module saturate, the contents of the Bloom filters are paged out into a log file. Later, according to the number of positive answers to a flow query into the log files offline, the flow volume can be probabilistically inferred.

### 3.2 Multi-Dimensional Hash Table

It is easy to see that a Bloom filter can be reduced to a hash table based on a single hash function, where  $k = 1$ ,

as shown in Fig. 1 (b). Alternatively, a Bloom filter is augmentation of a single hash table by the multiplicity of hashing functions. We generalize the construction of Bloom filters by introducing the concept of *dimensions* in hash algorithms, in which a dimension can expand by the number of either hash functions, hash tables, or both. We count the instances of expansions, and denote the number of dimensions accordingly.

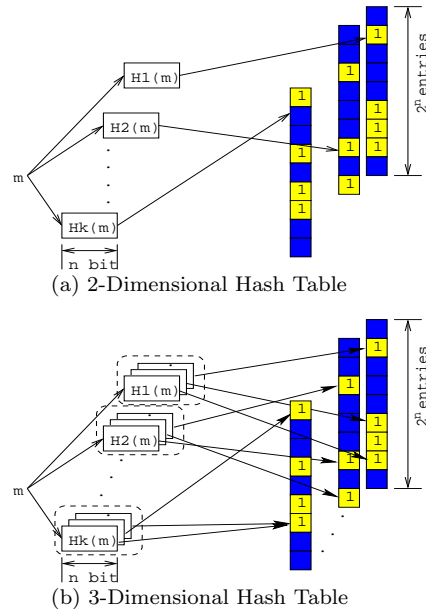


Figure 3: Dimensions of Hash Functions in Filters

For instances, a hash table based on a single hashing operation is one-dimensional, as shown in Fig. 1(b). The Bloom filter is a 2-dimensional hash table because a Bloom filter is implemented using  $k$  hashing functions in an array, as shown in Fig. 1(a). In addition, the Bloom filter table can be split into  $k$  separate hash tables, respectively, as shown in Fig. 3(a), which is another 2-dimensional hash table. In addition, the individual hash functions in Fig. 3(a) can again be augmented by  $k$  different hash functions, thus making it a 3-dimensional hash table, as shown in Fig. 3(b).

In particular, we refer to the 3-dimensional hash table in Fig. 3(b) as *Space-Time Bloom Filter (STBF)* in CAPTRA for WSN packet traceback because the 2-dimensional Bloom filter data structure is replicated and scattered among multiple sensor nodes, and updated asynchronously for packet tracking in different parts of the WSN. It is straightforward that the aforementioned space-code Bloom filter (SCBF) is another 3-dimensional Bloom filter in Fig. 3(b) for a different purpose.

By defining the dimensions in hash tables, we have generalize the concept of Bloom filter to a simpler but more comprehensive form, and captures all the hash algorithms in this paper. Naturally, the dimension of hashing operation is unlimited. Depending on how each Bloom filter in the dimensions is utilized, we can apply the appropriate dimensions of hashing operations for membership query, such as packet tracing, routing and accounting purposes [13].

## 4. GENERIC PACKET TRACEBACK USING SPACE-TIME BLOOM FILTER

We take similar assumptions about network operations for network-assisted packet traceback as in [23] that traceback operations are infrequent so that traceback overhead contributes only a minor addition to the overall network traffic, and that all traceback messages can be authenticated [24]. In addition, we assume that attackers may be aware of the packet traceback operations, therefore avoid being traced through certain other techniques, such as probabilistic packet marking scheme [12], and that router could be subverted, in which case the packet traceback works until it reaches the faulty router.

It is permissible to have small probability of giving false negatives, that is, the traceback protocol may not be able to traceback to the origin of some packets. Under such relaxation, we achieve other benefits as we see later.

The notation in Table 1 is used for our protocol specifications.

Table 1: Notation

$i, j$ etc.	Node identifiers.
<code>pkt</code>	Packet.
<code>pkt.src</code>	The source of a packet.
<code>pkt.dst</code>	The destination of a packet.
<code>pkt.tx</code>	The forward or transmitter of a packet.
<code>pkt.rx</code>	The intended receiver of a packet.
<code>pkt.id</code>	The identification information of a packet, also known as the packet digest.
$\text{Hash}_k$	Hash function with sequence number $k$ .

### 4.1 Packet Digest

Because the main purpose of packet traceback is to capture the path traversed by a specific packet through the network, without loss of generality, we extract the certain identification information to derived the packet digest. For instance, the packet digest of an IP packet may include IP version, header length, source and destination addresses, fragment information and portions of the payload [23], which is then concatenated and packed into a bit string as the input to hash functions.

Note that not all packets are necessarily traced, but only those with potential vulnerabilities to certain network attacks. It is a configurable behavior as to which of the traffic flows are to be tracked and traced in the tracing protocols.

### 4.2 Hash Functions

In order to use the same hash algorithm for different hash functions in the Space-Time Bloom filter (STBF), we feed additional information to the hash functions for packet tracing purpose:

1. An identifier for each hash function, which includes two pieces of information: (a) the host ID, and (b) a unique ID for the hash function.
2. The predecessor from which the packet is received.

By including the hash function ID, we can use the same hashing algorithm, such as MD5 [19] to compose different hash functions in the Space-Time Bloom Filter. By including the predecessor ID, a Bloom filter hit for a packet also

reveals the previous hop of the packet. In the actual computations, all neighbor IDs have to be checked in the Bloom filter in order to find out the predecessor.

### 4.3 Reset and Query to the Space-Time Bloom Filter

As an STBF continuously monitors network traffic, the hash tables can gradually saturate, and the probability of false positive increases. To guarantee the best performance tradeoff between the memory utilization and false positive rate, a “50% Golden Rule” is derived and applied so that the Bloom filter can hold the maximum number of elements [3, 13] with lowest false positive rate.

Previously applications of Bloom filters for accounting or tracing purposes reset the whole Bloom filter memory when the filter saturation crosses the threshold. Therefore, the false positive rate is a variable, while the false negative rate is either 0 before flushing or 1 after flushing for certain elements.

In the Space-Time Bloom Filter (STBF), individual Bloom filter elements are dispersed among multiple nodes. When the saturation ratio of an individual hash table crosses the 50% threshold, we only wipe out that single hash table in the STBF. Therefore, we define a majority-vote principle for STBF that allows STBF to yield a “hit” response when the number of “hit” responses from different Bloom filter elements satisfies a quorum. This is drastically different from the membership query operation in the traditional Bloom filter applications, where a hit requires unanimous hits on all the hash functions.

Due to the reset function on the STBF, the majority-vote quorum is a trade-off between the false negative and false positive rates in STBFs. If the quorum is low, the false positive rate increases, and if high, the false negative rate increases.

Intuitively, the majority-vote mechanism in STBF maintains gradually fading memory of the packet events in the past, and give more acute memory to recent traffic flows. We see increasing probability of false negatives because the individual hash tables are asynchronously being reset. In contrast, previous Bloom filter resets abruptly change the false positive and false negative rates of the Bloom filter to 0 or 1 in previous applications.

In order to avoid possible synchronized hash table resets, we add in certain randomness to the reset timing when the saturation rate reaches the threshold.

### 4.4 Traceback Operations

In support of packet traceback operations, three ICMP messages with new codes are proposed as shown in Table 2. Note that the use of ICMP messages are strictly for convenience, which is extended into our WSN simulations. Without loss of generality, similar message types can be implemented in real deployments for WSNs. These three messages jointly provide the majority-vote mechanism to corroborate the conviction of a node as one on the packet source route.

Message `TRACREQ` is to initiate a traceback query to discover the predecessor of a packet, implemented using the normal ICMP message of type 8 “Echo Reply”, used for ping-ing a network host. It is started by the access point in WSNs, and recursively carried out by the nodes on the packet source route. In the payload field of the ICMP message, the packet identification is carried.

**Table 2: New ICMP Codes for Packet Traceback**

Type	Code	Symbol	Meaning
8	1	TRAC <sub>REQ</sub>	ICMP “Echo Request” to <i>request</i> traceback of a packet.
8	2	TRAC <sub>VERD</sub>	ICMP “Echo Request” for <i>verdict</i> of a packet transmission based on the witness of a packet.
0	3	TRAC <sub>CONF</sub>	ICMP “Echo Reply” to <i>confess</i> the transmission of the packet.

Upon each message TRAC<sub>REQ</sub> arrival, the packet traceback protocol extracts the packet identification, and runs the hash functions on all the current one-hop neighbors to determine the potential predecessor of the packet. If such predecessor exists, the traceback operation can continue by sending the traceback request to the predecessor. Message TRAC<sub>VERD</sub> is for a node to indicate that it has witnessed the packet, and to issue a verdict to the sending node of the packet.

Message TRAC<sub>CONF</sub> is sent to the WSN access point by a node  $c$  to tell that node  $c$  has forwarded the packet before if it has collected enough verdicts from its neighbors. In addition, node  $c$  propagate the traceback request farther to its candidate neighbors for farther traceback.

## 5. COORDINATED PACKET TRACEBACK (CAPTRA)

### 5.1 Assumptions

We assume that a wireless sensor network consists of low-power computers that are interconnected via a shared wireless channel using the same channel access control protocol, such as the simplified IEEE 802.11 [1] without collision avoidance mechanism using RTS/CTS. In many wireless sensor network deployments, the use of access points (APs) as an aggregation point and the exit to the Internet is a common practice. APs possess sufficient computing power to implement the intrusion detection mechanisms and initiate traceback operations. In addition, we assume that the sensors’ memories are limited so that the Bloom filters cannot be arbitrarily large, nor can there be permanent storage for logging purposes.

The choice of the Bloom filter algorithm is a local choice dependent on the memory resources and computational tasks of the sensor node, and its parameters on each sensor node are configurable, such as the table sizes. The reporting of a hit in the Bloom filter at a wireless node is also dependent on the capability of the node. Nodes in relatively static and dense wireless environments can report the hits less frequently, because the voting quorum can easily be satisfied, than nodes in highly mobile and sparse networks for the same level of traceback accuracy.

For simplicity, we assume that nodes are homogeneous in wireless sensor networks in terms of memory allocation, hashing algorithm and hit reporting frequency.

The essential difference of packet traceback in wireless networks from wired networks operations is that the transmission medium is open and can be overheard by any node in the nearby vicinity of the packet transmitter. Therefore, we can take advantage of such fact in the Space-Time Bloom Filter constructions. That is, the Bloom filters of a single Space-Time Bloom Filter are virtually dispersed among adjacent wireless sensors, and the Bloom filter lookup is now a majority-vote by all the Bloom filters of the potential nodes.

The distribution of the virtual STBF is especially useful in wireless sensor networks because:

1. Although the local memory and computing power of

each sensor is limited, the aggregated Space-Time Bloom Filter has larger memory capacity and parallel computing power.

2. The traffic distribution is uneven in sensor networks, meaning that the traffic may be intense at some locations of the network, and sparse at others. Distributing the Bloom filters off-loads the memory requirements of the nodes with high traffic in its neighborhood.
3. Mobile environments often invalidate the source routes constructed from the packet traceback because of possible node movements and changing network topology. By distributing the Space-Time Bloom Filters to multiple nodes in a vicinity, the traceback operations can tolerate certain node losses along the source route because the predecessor of the missing node can still be detected if there is enough number of witnesses to the packet being sent from the predecessor.
4. The traceback based on the Space-Time Bloom Filters can find out relatively static nodes on the packet source route, which is also an important requirement for reconstructing the attack source path.

### 5.2 Packet Tracking

In CAPTRA, each node maintains a Bloom filter using  $k$  hashing functions. As mentioned before, upon overhearing a packet from a sender, a node hashes a concatenated bit stream containing the packet identification information, the sender’s ID and the node’s ID to a table index, and sets the corresponding bits to 1 in the Bloom filter. For instance, if node  $i$  hears a packet  $\text{pkt}$  with identification information  $\text{pkt.id}$ , the packet is tracked by a set of  $k$  bits in node  $i$ ’s Bloom filter, indexed by using Eq. (1).

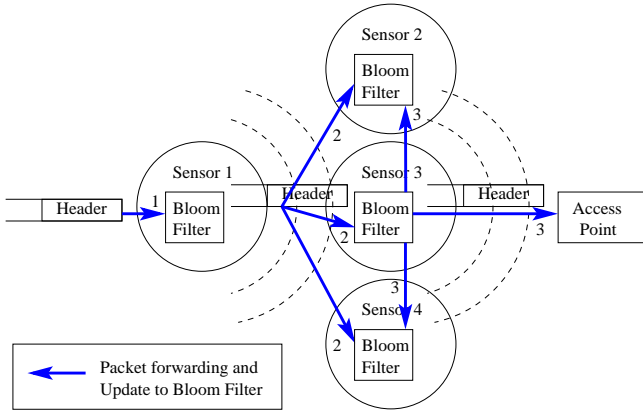
$$\bigcup_{j=1}^k \text{Hash}_j(\text{pkt.id} \parallel \text{pkt.tx} \parallel i \parallel j). \quad (1)$$

in which the symbol “ $\parallel$ ” represents the concatenation of the operands,  $j$  is the hash function ID, and  $k$  is the number of hash functions of the Bloom filter.

A packet is tracked when a sensor forwards the packet, or a sensor overhears and successfully receives the packet. Because the overhearing node’s ID is involved in the hashing algorithm, each node overhearing the packet maps the packet to a different Bloom filter entry. Therefore, the Bloom filter fill-up rates are different at different nodes.

Fig. 4 illustrates a partial sensor network with three sensors and an access point for data collection purposes. The blue arrows indicate the packet transmissions and receptions. The senders and the receivers of the packet record the packet in their Bloom filters.

According to the “50% Golden Rule”, the Bloom filters are refreshed once the saturation ratio reaches 50% of the Bloom filter capacity, and every bit of the Bloom filter is reset to 0.

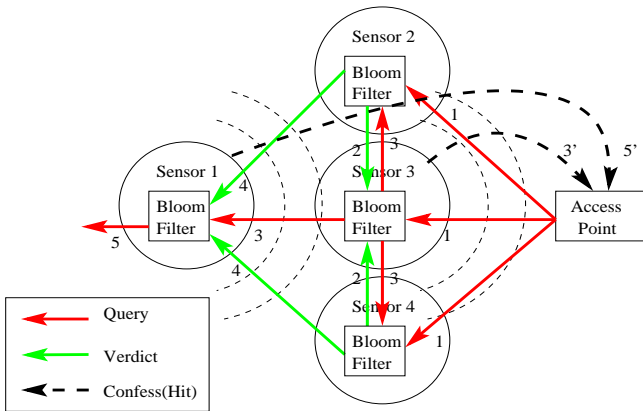


**Figure 4: Space-Time Bloom Filter for Packet Tracking**

### 5.3 Packet Tracing

In order to describe the packet traceback operations in CAPTRA, we again look the partial network in Fig. 4 as an example for illustrations.

Suppose that, in Fig. 4, the access point initiates a traceback query on packet  $\text{pkt}$  due to a security breach detected by an IDS. The access point sends out the traceback request with the packet digest  $\text{pkt.id}$  in the newly defined message  $\text{TRAC}_{REQ}$  with  $\text{pkt.id}$  as payload. Because the access point can look up its own Bloom filter for a hit, and find out the sender of the packet with digest  $\text{pkt.id}$ , the traceback request is sent directly to the packet predecessor — sensor 3, and is overheard by the sensors around the access point.



**Figure 5: Packet Tracing Process using Space-Time Bloom Filter**

Due to the broadcast nature of the wireless medium, sensors 2 and 4 overhear the traceback request in Fig. 5. Because sensors 2 and 4 overheard about packet  $\text{pkt}$  when it was forwarded 4, they must have recorded  $\text{pkt.id}$  plus the sender ID in their Bloom filters. If there is a positive hit, sensor 2 or 4 sends the  $\text{TRAC}_{VERD}$  packet to sensor 3 to reinforce the tracing request.

When enough verdicts are collected after a period of time, including the Bloom filter hits at sensors 2, 3 and 4, and the access point, sensor 3 sends back a  $\text{TRAC}_{CONF}$  packet to the access point for attack source route construction. The

threshold of the number of verdicts for further traceback is a tunable parameter, and is at least 2 because the traceback request sender and receiver also generate verdicts. Note that each traceback request is a verdict to the receiver of the traceback request.

In addition, sensor 3 generates further traceback request  $\text{TRAC}_{REQ}(\text{pkt.id})$  based on its current neighbors, and in Fig. 5, a new request is generated to sensor 1, after which the verdict generation process follows the same as above. In rare cases, multiple one-hop neighbors of sensor 3 appear to be the predecessor of the packet, each of them receives the traceback request. If a node is wrongly convicted of the packet transfer, it is expected the wrong traceback request gradually die out because of lack of evidence further away from the convicted node.

### 5.4 Loop Avoidance in Packet Traceback

Without addition mechanisms, the above traceback protocol can cause loops while forwarding the traceback request, one instance in Fig. 5 being that sensor 1 sends the traceback request again to sensor 3 because sensor 1 has heard about sensor 3's transmission of the packet, which causes unnecessary traceback activities.

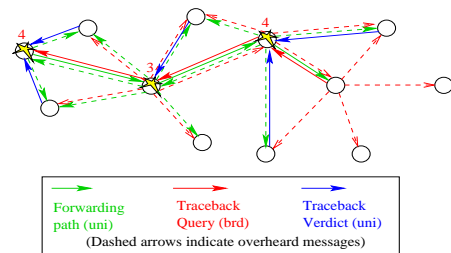
To avoid the potential looping problem, the traceback request is appended with distance information, indicating the distance from the current sender of traceback request to the origin of the traceback request.

If a node receiving the traceback request has a shorter distance to the request origin than the sender of the request, the traceback request is discarded. Note that the criteria based on the reverse path to the origin may be incorrect due to network mobility.

### 5.5 Termination of Tracing Query

When a node has no other one-hop suspect but itself who has transfer the packet, the tracing ends.

Some optimizations are possible in CAPTRA. In case the suspect which receives the traceback request is an uncooperative node, and refuses to respond to the request for packet tracing, the request generator may be able to conclude that the suspect has actually transfer the packet according to the verdicts from their shared one-hop neighbors, *e.g.*, sensors 1 and 3 in Fig. 5.



**Figure 6: Illustration of Data Packet Forwarding and Traceback**

Putting together the packet tracking and tracing operations in CAPTRA, Fig. 6 illustrates the packet tracking footprints while the packet is being forwarded from its source to its destination, and the packet tracing process after the packet is found unclear. The numbers above each node indicate the number of verdict received.

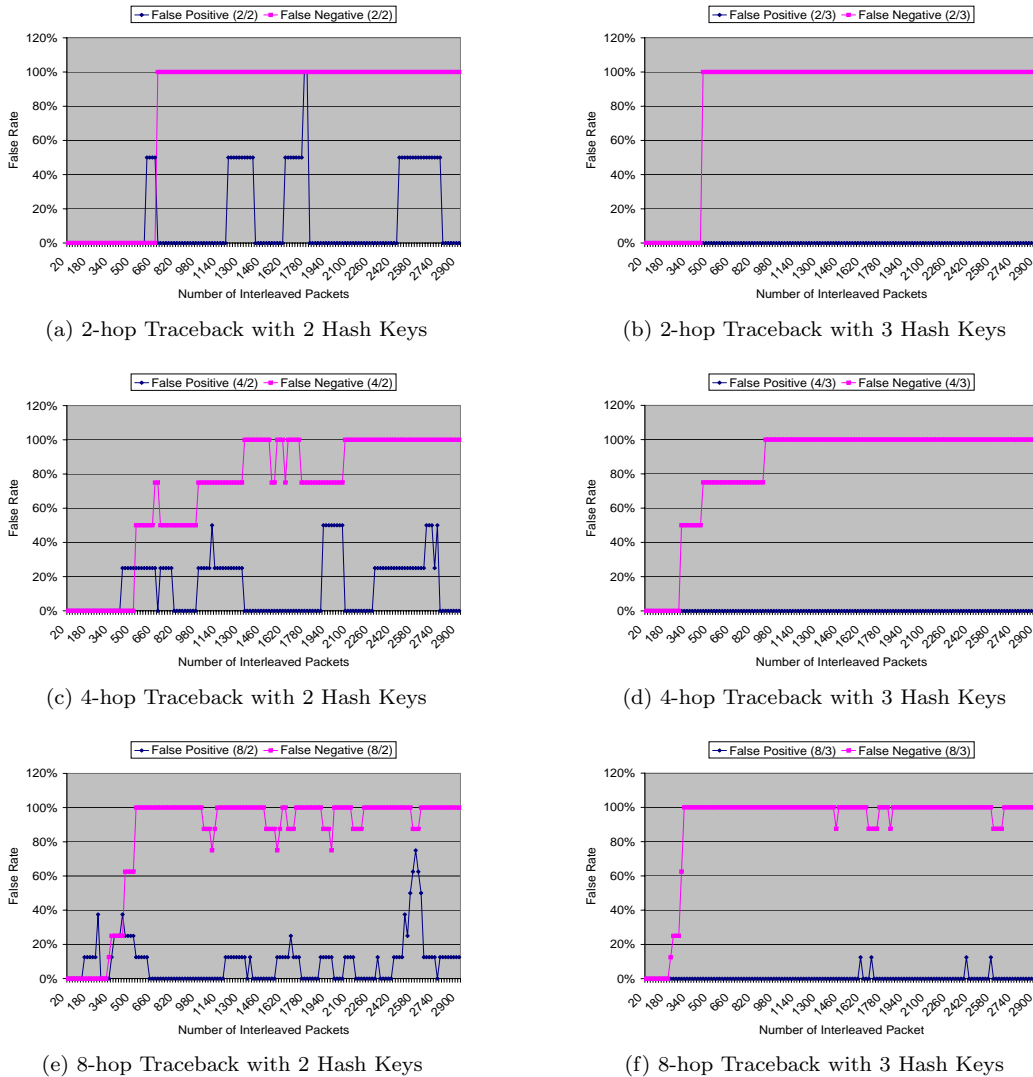


Figure 7: The False Positive and Negative Rates in Multihop Packet Traceback

## 6. EVALUATIONS

### 6.1 Simulation Setup

We used J-Sim [27] to simulate the packet traceback protocol in wireless sensor networks. J-Sim is a network simulator constructed entirely in JAVA. The MAC layer of the wireless sensors was modified to implement CAPTRA, RTS/CTS was turned off to simulate a WSN. For simplicity, the hashing function uses the senders MAC address as the `pkt.tx`, and the receiver’s MAC address as  $i$  in Eq. 1.

In the simulations, twenty wireless sensors, numbered from 0 to 19, are deployed in a linear fashion, spaced 8 meters apart. Each node has a 25-meter transmission range, so that each node has up to 7 neighbors. The propagation and path-loss model use the free-space model. AODV is used as the underlying routing protocol. Each node contains a Bloom filter of size 4096 bits, which is refreshed using the “50% Golden Rule”. The number of hash functions for each packet,  $k$ , is a system-wide variable, and varies in two simulation scenarios from 2 to 3 for comparison purposes.

The sensor traffic source is attached to node 0, and des-

tinued to node 19. A CBR traffic generation model with 512 byte per packet at a rate of 10 packets per second was used. We change the location node 0 in the network so as to change the hop distance to node 19 in order to evaluate the performance packet tracing with regard to the hop distance.

In each simulation round, the packet tracking capacity of the network is tested by collecting the false positive and false negative rates when different amount of traffic is forwarded between the times when the packet is generated at the source and when the packet is being traced back from node 19. As discussed earlier, the CAPTRA voting system is used to convict a node, and have a node confess. In the simulations, a node confesses as a packet forwarder when two nodes convicts it.

### 6.2 Simulation Results and Analysis

Fig. 7 presents the false positive and false negative rates under the various simulation scenarios. Two types of Bloom filters are implemented and compared side-by-side, in which one type of Bloom filter uses 2 hash functions in all the simulation scenarios, and the other uses 3 hash functions in

each. To calculate false positive rates we simply counted the number of false positives by the sensors of the WSN and divided by the hop count. Similarly, for false negative rates, we had the number of false negatives divided by hop count.

In Fig. 7 (a) (c) and (e), the traceback performance is measured for 2-, 4- and 8-hop source route of a particular packet. As we can see, when the number of hops increases, the network has shorter and shorter memory of the traced packet, indicated by the less number of interleaved packets from the time when the traced packet is generated to the time when the packet is traced back. In addition, the duration of the network memory about the packet is depended on the network density, and the size of the Bloom filters, which are 4096 bits in our simulations.

On the other hand, Fig. 7 (b) (d) and (f) show the false positive and false negative rates of similar simulations when the number of hash functions is three. Comparing with the other three corresponding sub-diagrams, these three settings presents more stable traceback performance, but shorter memory of the traced packet due to the faster fill-up rate of the Bloom filters. Therefore, the choice of Bloom filters depends on the application trade-offs between accuracy of traceback and the duration of the traceback validity.

There are unstable false rates on the curves of the false positive and false negative rates. This is due to the Bloom filter collisions. Due to space limitations, we skipped various other settings that variate and mix different Bloom filter setup strategies.

## 7. CONCLUSIONS

We have introduced a new perspective on expanding Bloom filter dimensions, which led to the Space-Time Bloom Filter (STBF) for coordinated packet traceback (CAPTRA) in wireless sensor networks (WSNs). CAPTRA takes advantage of the broadcast nature of the wireless transmissions for coordinated traceback, and is especially suitable for WSNs due to the small memory and computational requirements in maintaining STBF. The space-time Bloom filter construction enables distributed maintenance and asynchronous and gradual refreshing of the Bloom filters so that the WSN keeps robust and gradually fading memory of the packet traversal event. These unique organization and utilization of Bloom filters in CAPTRA allows our future application of packet traceback in adaptive routing in WSNs.

## 8. REFERENCES

- [1] IEEE Std 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical report, IEEE, Jul. 1997.
- [2] S.M. Bellovin, Marcus Leech, and Tom Taylor. ICMP Traceback Messages. Technical report, Internet Draft, IETF, Mar. 2001.
- [3] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, Jul. 1970.
- [4] A. Broder and M. Mitzenmacher. Network applications of Bloom filters: a survey. In *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.
- [5] H.Y. Chang, P. Chen, A. Hayatnagarkar, R. Narayan, P. Sheth, N. Vo, C. L. Wu, S.F. Wu, L. Zhang, X. Zhang, F. Gong, F. Jou, C. Sargor, and X. Wu. Design and Implementation of A Real-Time Decentralized Source Identification System for Untrusted IP Packets. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, Jan. 2000.
- [6] H.Y. Chang, R. Narayan, C. Sargor, F. Jou, S.F. Wu, B.M. Vetter, F. Gong, X. Wang, M. Brown, and J.J. Yuill. DECIDUOUS: Decentralized Source Identification for Network-Based Intrusions. In *Proceeding of 6th IFIP/IEEE International Symposium on Integrated Network Management*, pages 702–714, 1999.
- [7] Inc. CrossBow Technology. <http://www.xbow.com>, 2005.
- [8] D. Dean, M. Franklin, and A. Stubblefield. An Algebraic Approach to IP Traceback. In *Proceedings of Network and Distributed System Security Symposium*, Feb. 2001.
- [9] T.W. Doeppner, P. N. Klein, and A. Koyfman. Using Router Stamping to Identify the Source of IP Packets. In *7th ACM Conference on Computer and Communications Security*, pages 184–189, Athens, Greece, Nov. 2000.
- [10] J. R. Douceur. The Sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS)*, Mar. 2002.
- [11] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proceeding of SIGCOMM*, 1998.
- [12] M.T. Goodrich. Efficient Packet Marking for Large-Scale IP Traceback. In *9th ACM Conf. on Computer and Communications Security (CCS)*, pages 117–126, 2002.
- [13] A. Kumar, J. Xu, E.L. Li, and J. Wang. Space-code bloom filter for efficient traffic flow measurement. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 167–172, Miami Beach, FL, 2003.
- [14] J. Li, M. Sung, J. Xu, and L. Li. Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, May 2004.
- [15] A. Mankin, D. Massey, C. Wu, S. F. Wu, and L. Zhang. On Design and Evaluation of Intention-Driven ICMP Traceback. In *Proceedings of IEEE International Conference on Computer Communications and Networks (IC3N)*, 2001.
- [16] M. Mitzenmacher. Compressed Bloom Filters. *IEEE/ACM Transactions on Networks*, 10(3):613–620, Oct. 2002.
- [17] K. Park and H. Lee. On the Effectiveness of Probabilistic Packet Marking for IP Traceback. In *Proceedings of SIGCOMM*, pages 15–26, 2001.
- [18] S.C. Rhea and J. Kubiawicz. Probabilistic Location and Routing. In *INFOCOM*, 2002.
- [19] R. Rivest. RFC 1321 - The MD5 Message-Digest Algorithm. Technical report, MIT Laboratory for Computer Science and RSA Data Security, Inc., Network Working Group, Apr. 1992.
- [20] Y. Matias S. Cohen. Spectral Bloom Filters. In *SIGMOD Conference on Management of Data*, pages 241–252, 2003.
- [21] L.A. Sanchez, W.C. Milliken, A.C. Snoeren, F. Tchakountio, C.E. Jones, S.T. Kent, C. Partridge, and W.T. Strayer. Hardware Support for a Hash-Based IP Traceback. In *Proceedings of DARPA Information Survivability Conference and Exposition*, Jun. 2001.
- [22] S. Savage, D. Wetherall, A. Karlin, , and T. Anderson. Practical Network Support for IP Traceback. In *Proceedings of ACM SIGCOMM Conference*, Aug. 2000.
- [23] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer. Hash-based IP Traceback. In *Proceedings of ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, pages 3–14, 2001.
- [24] D.X. Song and A. Perrig. Advanced and Authenticated Marking Scheme for IP Traceback. In *Proceedings of IEEE INFOCOM Conference*, 2001.
- [25] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of 9th Usenix Security Symposium*, Aug. 2000.
- [26] S. Templeton and K. Levitt. Detecting spoofed packets. In *Proceedings of The Third DARPA Information Survivability Conference and Exposition (DISCEX)*, 2003.
- [27] H. Tyan. J-Sim. <http://www.j-sim.org/>.
- [28] B. Vetter, F. Wang, and S.F. Wu. An Experimental Study of Insider Attacks for the OSPF Routing Protocol. In *IEEE International Conference on Network Protocols (ICNP)*, pages 293–300, Oct. 1997.
- [29] M. Waldvogel. GOSSIB vs. IP Traceback Rumors. In *Proceedings of 18th Annual Computer Security Applications Conference (ACSAC)*, Dec. 2002.
- [30] X. Wang and D.S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *ACM Conference on Computer and Communications Security*, pages 20–29, 2003.