Limiting DoS attacks during multihop data delivery in wireless sensor networks

Jing Deng,* Richard Han and Shivakant Mishra

Department of Computer Science,
University of Colorado, Boulder, CO, USA
E-mail: jing@cs.colorado.edu
E-mail: rhan@cs.colorado.edu
E-mail: mishras@cs.colorado.edu
*Corresponding author

Abstract: Denial of Service (DoS) attacks can be easily launched in Wireless Sensor Networks (WSNs). Due to their resource constraints, namely limited energy, memory and bandwidth, WSNs are especially vulnerable to DoS attacks. This paper addresses a particular class of DoS attacks that overwhelm resources along a multihop data delivery path. Since WSNs are typically tree-structured, then a DoS attack on a path will be especially effective in denying routing service to an entire branch of sensor nodes, not just the nodes along the path. This paper proposes a solution using one-way hash chains to protect end-to-end multihop communications in WSNs against such Path-based DoS (PDoS) attacks. The proposed solution is lightweight, tolerates bursty packet losses and can easily be implemented in modern WSNs. This paper reports on performance measured from a prototype implementation.

Keywords: security; Wireless Sensor Networks (WSNs); Denial of service (Dos).

Reference to this paper should be made as follows: Deng, J., Han, R. and Mishra, S. (2006) 'Limiting DoS attacks during multihop data delivery in wireless sensor networks', *Int. J. Security and Networks*, Vol. 1, Nos. 3/4, pp.167–176.

Biographical notes: Jing Deng is a PhD Student in Computer Science at the University of Colorado at Boulder. He received his BE from Univeristy of Electronic Science and Technology of China in 1993 and his ME from the Institute of Computing Technology, Chinese Academy of Science in 1996. He has published ten papers on security in wireless sensor networks and a book chapter on security, privacy and fault tolerance in sensor networks. His research interests include wireless security, secure network routing and security for sensor networks.

Richard Han joined the Department of Computer Science at the University of Colorado at Boulder in August 2001 as an Assistant Professor, He leads the MANTIS wireless sensor networking research project. He has served on numerous technical programme committees for conferences and workshops in the field of wireless sensor networks. He received a National Science Foundation CAREER Award in 2002 and IBM Faculty Awards in 2002 and 2003. He was a Research Staff Member at IBM's Thomas J. Watson Research Centre in Hawthorne, New York from 1997–2001. He received a PhD in Electrical Engineering from the University of California at Berkeley in 1997, and a BS in Electrical Engineering with distinction from Stanford University in 1989. His research interests include systems design for sensor networks, secure wireless sensor networks, wireless networking and sensor-enabled user interfaces.

Shivakant Mishra received the B.Tech in Computer Science and Engineering from the Indian Institute of Technology, Bombay and a PhD in Computer Science from the University of Arizona, Tucson. Currently, he is an Associate Professor in the Department of Computer Science, University of Colorado, Boulder, CO. His current research interests include the design and implementation of secure and highly available wireless sensor networks and system-level support to facilitate highly available and dependable computing over the internet.

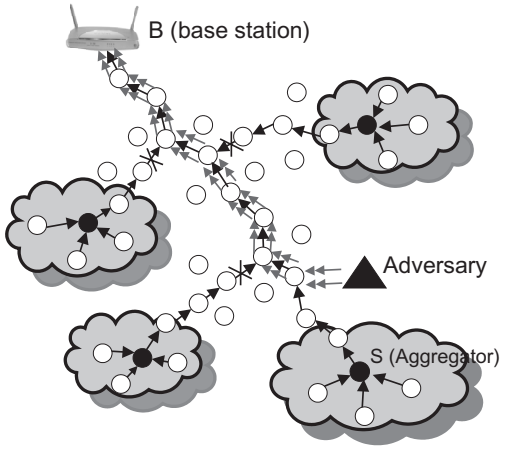
1 Introduction

Wireless Sensor Networks (WSNs) offer the promise of exciting new technological developments. Applications of WSNs are wide-ranging, including environmental monitoring, smart spaces, military deployments, medical systems and robotic exploration.

Large sensor networks are often organised hierarchically via a tree structure to conserve energy and network bandwidth. The tree consists of leaf sensor nodes, aggregator nodes and a base station, as given in Figure 1. Leaf sensor nodes send their sensed data to an aggregator node. Aggregator nodes process and summarise the data from member nodes and send the aggregated result to a base

station. Communication between sensor nodes and their aggregator and between an aggregator and the base station occurs in general via a multihop, end-to-end communication path.

Figure 1 A PDoS attack in end-to-end communication in WSNs



Due to their inherent limitations, WSNs are especially sensitive to Denial of Service (DoS) attacks (Wood and Stankovic, 2002). In contrast to resource-rich networks such as the internet, a WSN is less stable, more resource limited, subject to open wireless communication and prone to the physical risks of in situ deployment. These factors increase the susceptibility of WSNs to DoS attacks.

While an adversary may resort to a localised signal jamming attack, a more effective form of DoS attack against a WSN is to overwhelm nodes that are many hops away by flooding packets, which will quickly exhaust the limited energy, communication bandwidth, memory and CPU of resource-limited sensor nodes. INSENS proposed One way Hash Chains (OHCs) to limit the ability of an attacker to flood to the entire sensor network (Deng et al., 2005). During set up of the routing tables, OHCs limit broadcast flooding of control packets. After routing tables have been securely set up, data packets are confined to securely specified routes and thus cannot flood the entire WSN. However, after the data begins flowing, INSENS does not address how to limit an adversary from flooding replayed or spurious data along any routing path, which will overload all nodes along the path towards a base station.

Other forms of Distributed Denial-of-Service (DDoS) attacks are also possible, analogous to DDoS attacks in the internet (Lu et al., 2005). Such attacks would typically require cooperation among multiple enemy agents. This cooperation can be extended to allow wormholes to be formed (Hu et al., 2003) between two agents in different parts of the network. Such sophisticated attacks require substantial cooperation and thus pose a significant investment of effort by the attacker.

Our observation and the focus of this paper, is that a much simpler yet highly effective class of DoS attacks against WSNs can be launched by a single adversary flooding packets along a multihop data delivery path. Such an attack, which we term a *Path-based DoS* (PDoS) attack, exploits the tree-structured routing of WSNs to cause broad DoS for modest effort. Figure 1 shows the nodes that would be affected by a PDoS attack. Suppose an adversary floods bogus packets along a path towards the base station. Firstly, nodes along the path will quickly become exhausted. Secondly, all nodes on the branch containing the attacked path will also be unable to communicate with the base station, due to the tree-structured topology of a WSN. For example, each of the aggregator clouds shown will be unable to reach the base station. Thus, an adversary launching a PDoS attack in a WSN can disable a much wider region than simply a single path. This problem was important enough to be addressed in several other papers (Ye et al., 2004; Zho et al., 2004), though it was not given an explicit name. We will discuss these solutions in more detail in the related work section.

To defend against a PDoS attack, an intermediate node must be able to detect spurious packets or replayed packets and then reject them. One way to detect spurious packets is to have the source node establish a separate shared key with other sensor nodes in the communication path. The sender node then uses each key to separately generate authentication/integrity information for each packet to satisfy each node along the path. However, the highly restricted packet size in WSNs (e.g. 29 bytes for data in TinyOS packet) makes it difficult to include such a large amount of verification information in a sender's packet, for example, an 8-byte Message Authentication code (MAC) (Perrig et al., 2002) for each node in the path. In addition, this imposes an onerous burden on the sender, who must know a priori each node in the path in order to send the relevant verification information. Alternatively, a sender could use a single 'path' key that is shared with each node along the path, thus requiring only one MAC for each packet. This approach is vulnerable to compromise any of the sensor nodes along a path, because the attacker will then have the path key and be able to flood legitimate packets along the path in a PDoS attack. One way to detect replay of duplicate packets is to have each intermediate node store a history of all packets they have forwarded. However, both memory and computation limitations of a sensor node make this solution infeasible.

Another possible PDoS defence is to limit the number of packets an intermediate node can forward per second, namely rate control. However, given the asymmetric nature of WSNs, nodes at different locations need to forward different numbers of packets per second. For example, nodes near a base station will typically need to forward more packets per second than the nodes far from the base station. Furthermore, different types of sensor nodes have different packet sending rate requirements, for example, aggregator nodes reporting different types of events, nodes undergoing dynamic reprogramming, etc. In addition, when a routing path changes, the rate control setting for some nodes need to be updated. Security, efficiency and scalability issues suggest that a rate control solution is non-trivial.

In this paper, we propose a general lightweight secure mechanism to defend against PDoS attacks on intermediate nodes in a multihop end-to-end data path in WSNs. This mechanism configures a one-way hash chain in each node along a path, enabling each intermediate node to detect a PDoS attack and prevent the propagation of spurious or replayed packets. In this mechanism, every packet sent by an end point includes a new one-way hash chain number. An intermediate node forwards a packet only if the included

OHC number is verified to be new. This OHC-based solution is more resilient to compromise than the approach of sharing a single path key since an adversary who obtains the current and earlier OHCs cannot generate a legitimate next OHC number and therefore cannot flood the path with bogus packets or replayed packets. This OHC-based solution also requires minimal storage. Our approach has the advantage that it is general, that is, the solution applies to any multihop data communication path used for unicast or reliable end-to-end data delivery.

This paper makes four contributions. Firstly, this Section identified an important DoS attack that is relatively easy to launch and can severely impact both computation and communication in a WSN. Secondly, it proposes an efficient and lightweight mechanism based on OHCs to defend a sensor network against PDoS attacks. While OHCs have been used to solve security problems in the internet (Perrig et al., 2000), wireless ad hoc networks (Hu et al., 2002) and WSNs (Deng et al., 2003, 2005; Perrig et al., 2002; Zhu et al., 2003), our unique approach is to apply OHCs to protect *unicast paths* from easily launched DoS attacks. Thirdly, Sections 3.2 describe novel and robust mechanisms to *maintain* OHCs given high WSN packet loss rates (Zhou and Govindan, 2003), irregular spatial wireless ranges (Zhou et al., 2004), and frequently time varying transmission ranges (Woo et al., 2003). These mechanisms include bootstrapping an OHC in intermediate nodes, refreshing an OHC after a loss of a burst of packets and adapting to path changes. Finally, in Section 5.2.2, the proposed OHC solution has been implemented and quantitatively evaluated on modern sensor nodes in terms of its storage and generation costs, demonstrating the feasibility of our solution.

2 Related work

DoS attacks in WSNs are a critical security issue. Different types of DoS attacks in different layers of a sensor network protocol stack are discussed in Wood and Stankovic (2002), and some countermeasures to defend against them are proposed. Security problems of different sensor network routing protocols are analysed and mechanisms to enhance the security of sensor network routing are proposed in Karlof and Wagner (2003).

Perrig et al. proposed the μ TESLA protocol to securely broadcast messages in a WSN (Perrig et al., 2002). This protocol uses an OHC number as the key to generate a MAC of a broadcast message. A different OHC number is allocated for each time slot and this number is used to generate MACs for the packets sent in that time slot. To tolerate packet losses, μTESLA has been extended by introducing multi-level one-way hash chains Liu and Ning (2003). A higher-level OHC is used to bootstrap low-level OHCs. We adopt the idea of using a higher-level OHC to maintain low-level OHCs in our solution to tolerate a sequence of packet losses. However, we use a different mechanism to maintain low-level OHCs and our OHC maintenance scheme does not require time synchronisation.

Hu et al. (2002) proposed a secure on-demand routing protocol for ad hoc networks, in which an OHC is used to thwart malicious routing request floods. When an initiator node broadcasts a ROUTE REQUEST message, it attaches an OHC number on the message. Other nodes can check the authenticity of the packet by verifying the OHC number. OHCs were used in INSENS to limit broadcast floods for control routing updates in WSNs (Deng et al., 2005). In contrast, our approach employs OHCs to defend against DoS attacks on *unicast* messages that follow a path. Problems unique to unicast messages must be addressed, for example, maintaining OHCs when many packets are lost and how to generate and store OHCs in a highly resource-constrained sensor node.

Recently, en-route filtering schemes have been proposed for intermediate nodes to filter false data generated by malicious aggregator nodes as well as intruders engaged in what we have termed PDoS attacks (Ye et al., 2004; Zhu et al., 2004). The basic idea is that the intermediate nodes share some keys with the member nodes in a sensor node group or cluster. Member nodes generate MACs for the reported data using the shared keys. Intermediate nodes can verify the MACs before forwarding packets. In the SEF scheme proposed by Ye et al., the Bloom filter is used to reduce the size of MACs and ensure their security. The intermediate nodes and member nodes use randomly predistributed keys to generate and verify MACs. In this scheme, it is highly likely that the false data will be dropped by one of the intermediate nodes and would not reach the base station. However, there are several problems with the SEF scheme. Firstly, SEF uses a probabilistic approach. It cannot guarantee that every spurious packet will be filtered out on the path. In addition, statistically, a spurious packet will be forwarded for a certain number of nodes before it is filtered out. Secondly, the message overhead of SEF is still large. The size of the Bloom filter is 14 bytes long, which is about half of data payload of a TinyOS packet.

In the interleaved key scheme proposed by Zhu et al. (2004), member nodes and intermediate nodes set up interleaved keys using randomly predistributed keys. These interleaved keys and hop-by-hop authentication ensure that the base station will detect any false packets when no more than a certain number (t) of nodes are compromised. The problem of the interleaved key scheme is that there is no efficient mechanism to authenticate two nodes to each other through multiple hops. In addition, the communication overhead of the pairwise key establishment for multihop nodes is large and the process is slow.

In contrast to SEF and interleaved keys, our PDoS solution filters out bogus packets immediately, wherever they are injected. Unlike (Ye et al., 2004), our mechanism is deterministic and guarantees that, that the bogus packets will be filtered out with only a small message overhead. Unlike (Ye et al., 2004; Zhu et al., 2004), our mechanism is easy to bootstrap, is lightweight and flexible in the face of routing path changes and is extensible enough to protect general forms of unicast communication against PDoS attacks, for example, reliable end-to-end communication between a base station and a sensor node.

Some fault tolerance routing mechanisms can make it harder on the attacker to deny service to the WSN. For instance, in Mint routing (Woo et al., 2003), a node has multiple parent nodes and it selects one parent node to

forward its data based on the connectivity quality between itself and the parent node. So if the one parent node is overwhelmed by a PDoS attack, the node can select another parent node to send its data. However, the PDoS attack can still be launched against Mint routing; the only added requirement is that the attacker possess more energy, that is, it will simply take longer for the attacker to exhaust the network. The nodes on the attacked path will still be overwhelmed by a PDoS attack. Even worse, nodes along alternative paths will also eventually be overwhelmed.

3 A lightweight defence against path-based DoS attacks

3.1 Assumptions

We assume a standard WSN system model in which sensor nodes forward data via a tree-structure routing topology to a base station. The sensor nodes and base station are stationary after deployment. Data packets are forwarded along an end-to-end multihop data communication path between a sensor node S and a base station B, namely $S \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_m \rightarrow B$, where $n_1 \ldots n_m$ are the intermediate nodes. The path between S and B has already been securely set up by a protocol such as INSENS (Deng et al., 2005). Our goal above and beyond the secure routing, which protects the control packets used to set up routing, is to prevent PDOS attacks of flooded *data* packets along these established paths. The paths may change over time for a variety of reasons. Section 5.2 addresses how our solution adapts to path changes caused by fluctuating radio ranges. The underlying secure routing scheme is assumed to be able to adjust to topology changes caused by node failure and/or duty cycle sleeping. S and B share a secret key that they use to protect the confidentiality, integrity and authenticity of the data exchanged.

An adversary can eavesdrop upon, modify or block any packets transmitted along the path from S to B. She can also inject any number of spurious packets along this path. If the adversary compromises an intermediate node n_k, she can determine all keys stored in n_k, and control every packet passed through n_k. To launch a PDoS attack, an adversary can inject bogus packets, compromise an intermediate node or compromise a source node S. In general, DoS attacks would be easy to defend against if we knew where the adversary launched an attack. When an adversary launches a DoS attack from a fixed sender S, a base station can use its shared key with S or the OHC to identify misbehaviour from a malicious S, and inform intermediate nodes not to forward any more packets for S. However, replay PDoS attacks can be initiated from anywhere along a path. As a result, in this paper, we focus on intermediate nodes or outside sources capable of launching PDoS attacks.

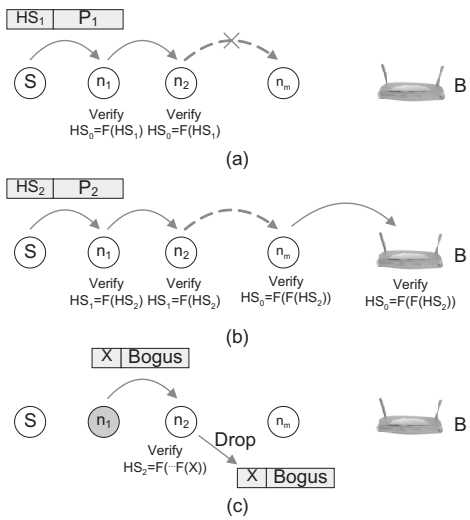
Our goal here is to address only PDoS-style attacks. Verifying whether the content of an aggregation result is correct is beyond the scope of this paper and has been addressed elsewhere (Przydatek et al., 2003). We also do not focus on localised jamming or blocking attacks that an adversary may launch (Deng et al., 2003; Wood et al., 2003) or more exotic attacks to routing schemes, such as the rushing attack or wormhole attack Karlof and Wagner (2003).

3.2 Basic scheme using one-way hash chains

A one-way hash chain is employed as an efficient and simple solution on resource-constrained sensor nodes for mitigating DOS attacks along paths. A one-way hash chain (Lamport, 1979) is a sequence of numbers generated by a one-way function F that has the property, that for a given x it is easy to compute $y = F(x)$. However, given F and y, it is computationally infeasible to determine x, such that $x = F^{-1}(y)$. An OHC is a sequence of numbers K_n, K_{n-1},..., K_0, such that $\forall i : 0 \leq i < n$, $K_i = F(K_{i+1})$. To generate an OHC, we first select a random number K_m as the seed and successively apply function F on K_m to generate other numbers in the sequence.

To defend against a PDoS attack, each source node S (mostly S is an aggregator node) maintains a unique one-way hash chain $\mathrm{HS} : < \mathrm{HS}_n, \mathrm{HS}_{n-1}, \ldots, \mathrm{HS}_1, \mathrm{HS}_0 >$. When S sends a packet to the base station through multiple hops, it includes an OHC *sequence number* from HS in the packet: it attaches HS_1 in the first packet, HS_2 in the second packet, and so on. To validate an OHC number, each intermediate node $n_1, \ldots, n_m$ maintains a verifier V_S for node S. Initially, V_S is set to HS_0. When S sends its ith packet, it includes HS_i with the packet. When an intermediate node n_k receives this packet, it verifies if $V_S = F(\mathrm{HS}_i)$. If so, n_k validates the packet, forwards it to the next intermediate node and sets V_S to HS_i. In general, n_k can choose to apply the verification test iteratively up to a fixed number w times, checking at each step whether $V_S = F(F...(F(\mathrm{HS}_i)))$. If the packet is not validated after the verification process has been performed w times, n_k simply drops the packet. Figure 2 demonstrates this mechanism. The reason for performing the verification process more than once is to tolerate packet losses. In particular, by performing the verification process w times, up to a sequence of w packet losses can be tolerated, where the value of w depends on the average packet loss rate of the network.

Figure 2 Defending against PDoS attacks with a one-way hash chain



This OHC-based scheme brings several advantages. Firstly, it constrains PDoS attacks from an adversary, since an adversary cannot generate the next valid OHC number, while replayed old OHC numbers will be dropped immediately.

Secondly, the memory and computational costs of OHC execution are quite lightweight, as we will show. Thirdly, this scheme tolerates packet losses. Finally, our approach does not require tight time synchronisation, unlike SEF (Ye et al., 2004) or interleaved keys (Zhu et al., 2004). A source node can send its message at any time without needing to be tightly synchronised with any intermediate node.

One possible attack in this scheme is that a malicious node can listen to and block all packets sent from the source node and in addition, collect all the OHC numbers included in these packets. These accumulated numbers can be used to generate a flash flood against subsequent intermediate nodes by sending a burst of spurious packets in a very short period of time. Since the subsequent intermediate nodes have not seen these OHC numbers, they will validate the corresponding packets and forward them. However, such an attack is limited in two aspects. Firstly, the adversary will have to wait for a relatively long period of time to collect a large number of valid OHC numbers that it is blocking. Secondly, the adversary can send only as many packets as the number of OHC numbers it has collected, that is, such an attack can be sustained for only a short period of time.

If the packets sent by a source node arrive out of order, for example, a source node sends packet p_1 first and p_2 secondly and somehow an intermediate node n_i gets p_2 before getting p_1, then n_i will drop p_1 since n_i will think that p_1 has an old OHC number. Although out-of-order arrival is common in internet routing, since packets from the same source to the same destination may be routed through different paths, we argue that this is not a large problem in sensor networks because standard routing paths between a source node and destination node are typically unique. In addition, each node will forward packets based on a First In, First Out (FIFO) policy. We will discuss the effect of changes in the routing topology later.

4 Bootstrapping the initial one-way hash chain number

Our solution requires that every intermediate node be configured with the initial OHC number ($V_S = \mathrm{HS}_0$) before communication can begin. One advantage of this OHC scheme is that we only need to protect authenticity, not confidentiality, of the initial OHC number. To bootstrap the initial OHC number, the base station can apply either a public key scheme (Malan et al., 2004) or a μTESLA secure broadcast mechanism (Perrig et al., 2002).

As mentioned in Section 3.2, the path between the base station B and the sensor node S is assumed to have been already set up as: $B \rightarrow n_m \rightarrow n_{m-1} \rightarrow \cdots \rightarrow n_1 \rightarrow S$. In our public key scheme, the base station posses a private key PK_s and every node has the corresponding public key PK_p. To bootstrap HS_0, the base station sends a message containing HS_0 and a signature of HS_0 signed with PK_s to the nodes n_1 to n_m and S in the path. When a node n_k receives this message, it can use PK_p to verify the authenticity of HS_0 and forwards the message to the next node n_{k-1} if the verification is a success. Malan et al. (2004) implemented an elliptic curve public key scheme on Berkeley motes. Their experiment showed that the encryption/decryption process costs 30–40 s. Compared to the μTESLA option described below, the public key approach is slow. As a result, we employ public keys only during bootstrapping and not during per packet verification. Public key bootstrapping of OHCs has the advantage over μTESLA that loose time synchronisation is not required.

To apply the μTESLA protocol, all nodes in the network are loosely time synchronised. When base station B bootstraps a one-way hash chain, B generates a packet containing HS_0, the ID of the destination node S and a MAC for the packet using key K_i, where K_i is the number in the key chain number corresponding to time slot t_i. The packet format is:

$$\text{bsp} : B|S|\mathrm{HS}_0|\mathrm{MAC}_{K_i}(B|S|\mathrm{HS}_0)$$

In the time slot t_i, B sends bsp to n_m. n_m records HS_0 and forwards the packet to n_{m-1}; n_{m-1} records HS_0 and forwards the packet to n_{m-2}; and so on, until the packet reaches S. To authenticate HS_0, B releases the key K_i in time slot t_{i+d}. On receiving this key, an intermediate node can verify the integrity and source authentication of HS_0.

Notice that the bsp message does not flood to the whole network, which saves data bandwidth and would not bring any attacks against the network even if the nodes on the other side of the network do not receive K_i at t_{i+d}. Since the messages that are MACed by K_i are supposed to be sent out at time slot t, an adversary cannot launch any attacks with K_i when he gets K_i at t_{i+d} (Perrig et al., 2002).

5 One-way hash chain maintenance

5.1 Refreshing a broken OHC

An intermediate node performs the verification process up to w times. This allows the node to tolerate a sequence of up to w packet losses. However, if a sequence of more than w packets are lost, an intermediate node will be unable to recover, that is it would not be able to validate any later packets and will simply drop them. We call this problem a broken OHC problem. An adversary can exploit this limitation by jamming the communication medium around an intermediate node for a sufficient time period that will result in more than w packet losses. In this way, an adversary can block the communication between a source node S and a base station B by only launching jamming attacks for a short time.

Simply increasing the value of w can help intermediate nodes tolerate more packet losses, but cannot defend against an adversary's jamming attack. To address the broken OHC problem, we periodically bootstrap a new OHC number (the OHC number most recently sent by the source node) in the intermediate nodes. This way, even if a sequence of more than w packets are lost, intermediate nodes can set up a new value for V_S using this periodic bootstrapping mechanism and validate subsequent packets. As a result, if an adversary wants to attack this enhanced scheme and block communication between S and B, she is forced to repeatedly launch jamming attacks by either jamming all packets or by periodically jamming bootstrap messages. She cannot block

all future communication by simply jamming for a short duration.

There are two problems in bootstrapping a new OHC number that we need to address. Firstly, how can intermediate nodes authenticate the new OHC number? Secondly, if an adversary intercepts the new OHC number, how do we stop the adversary from flooding spurious packets to upstream intermediate nodes that have not received the new OHC number? For example, suppose the last OHC number that an intermediate node n_k received is HS_i and the most recent OHC number that S has sent out is HS_j, where $j - i > w$. Suppose HS_{j+1} is being bootstrapped using the periodic bootstrapping mechanism. If an adversary intercepts HS_{j+1}, she can generate $j + 1 - i$ spurious packets containing valid OHC numbers ($\text{HS}_{i+1}, \ldots, \text{HS}_{j+1}$) and send them in a flash flood to n_k. n_k will validate these OHC number and forward the corresponding spurious packets to the next node.

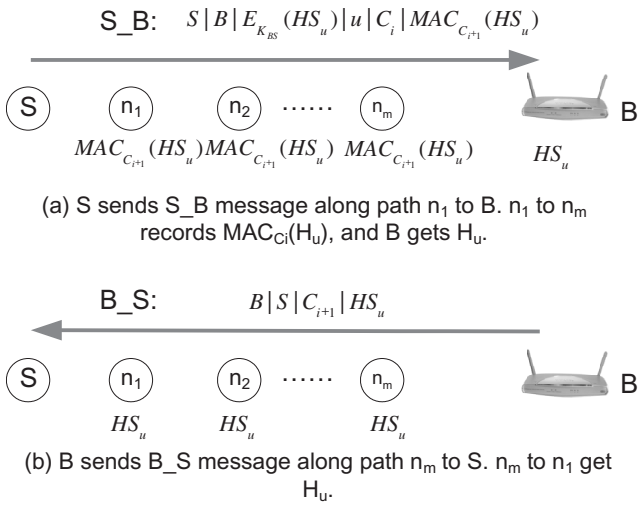
To address the first problem, a second one-way hash chain is introduced, called the *control* one-way hash chain (denoted by OHC_C), to authenticate the newly bootstrapped OHC number. OHC_C is only used for periodically bootstrapping a new one-way hash chain number for data transmission. In this section, the OHC used for data communication is denoted by OHC_D, to differentiate it from OHC_C. To address the second problem, the new OHC_D number is bootstrapped first in nodes closer to the base station, that is, the new OHC_D number is bootstrapped in node n_k before node n_j, where $k > j$. This way, if node n_k is compromised, it cannot use the new OHC number to generate spurious packets and forward them to nodes n_{k+1}, n_{k+2}, etc. This is because nodes n_{k+1}, n_{k+2}, ... would have already received the new OHC number, and so drop the spurious packets.

Our solution to refresh a broken one-way hash chain combines these two mechanisms as follows. The source node S and base station B share a second one-way hash chain $\text{OHC}_C :< C_m, C_{m-1}, \ldots, C_0 >$. All intermediate nodes are bootstrapped with the initial number of OHC_C, that is, C_0, using the mechanism described in Section 3.2. As given in Figure 3, a roundtrip exchange is employed consisting of two messages, a RQST S_B and an RACK B_S, to bootstrap a new OHC_D number HS_u in the intermediate nodes. The first message (S_B) sent by S to B contains a hash value of HS_u ($\text{MAC}_{C_{i+1}}(\text{HS}_u)$), index of HS_u in OHC_D (u), the next new OHC_C number (C_i) and an encrypted form of HS_u ($E_{K_{BS}}(\text{HS}_u)$). The hash value is computed using C_{i+1}, which is the next new OHC_C number after C_i, and encryption is done using a secret key K_{BS} shared between S and B. C_i is included to loosely authenticate the source of the message. The format of an S_B message is as follows:

$$\text{RQST}S\_B : S|B|E_{K_{BS}}(\text{HS}_u)|u|C_i|\text{MAC}_{C_{i+1}}(\text{HS}_u)$$

When an intermediate node receives this message, it authenticates the source by verifying C_i as the next OHC_C number, given that C_{i-1} or earlier was received in a previous B_S message. If authenticated, this node saves the MAC verification for HS_u, and forwards the message to the next node. When B receives this message, it first authenticates the source. If authenticated, it decrypts $E_{K_{BS}}(\text{HS}_u)$ and obtains the new OHC_D number HS_u. At this point, B knows that every intermediate node have already received the MACed HS_u.

Figure 3 Refreshing a broken OHC number



(a) S sends S_B message along path n_1 to B. n_1 to n_m records $\text{MAC}_{\text{Ci}}(\text{H}_\text{u})$, and B gets H_u.

(b) B sends B_S message along path n_m to S. n_m to n_1 get H_u.

The next stage in Figure 3(b) consists of B releasing the plaintext of HS_u to each node along the path via an RACK. B sends a message (B_S) to S that has the following format:

$$\text{RACK}B\_S : B|S|C_{i+1}|\text{HS}_u$$

When an intermediate node receives this message, it first authenticates the source by verifying that C_{i+1} is the next new OHC number in OHC_C. It then computes $\text{MAC}_{C_{i+1}}(\text{HS}_u)$ using C_{i+1} and HS_u included in this message and compares it with the hashed value of HS_u received earlier in message S_B. If there is a match, the intermediate node assigns HS_u to V_S and forwards the message to the next node. When S receives B_S message, it knows that every intermediate node has received HS_u, so it can use $\text{HS}_{\text{u}+1}$ for the next packet.

This refresh mechanism is resilient to a variety of attacks. Firstly, the use of the control hash chain C_i prevents nodes from flooding forged RQST and RACK messages to intermediate nodes. Secondly, the sequence of disclosure means that a node n_k learns HS_u before n_j if $k > j$, that is, if n_k is closer to B. However, n_k cannot use this knowledge to launch a PDoS attack on n_j since traffic goes from n_j to n_k.

5.2 *Resilience to path changes*

Due to irregularity of radio coverage (Zhou et al., 2004) and frequent changes in the data transmission range (Woo et al., 2003), the end-to-end routing paths in WSNs can change during an end-to-end communication. For example, by monitoring routing information broadcast by its neighbour nodes, a node n_k may detect that it cannot reach n_{k+1}, but that it can reach n_{k+2} via another node a_1. When a routing path changes, new nodes joining the path will need to securely receive the OHC number and initialise their verifier V_S.

One approach to deal with a path change is to employ the bootstrap protocol every time the path changes. However, this method is costly. In addition, it exposes the protocol to new DOS attacks: an adversary can simply jam one node on the path, causing the path to change. This forces the base station to rebootstrap a new OHC for all nodes on the path. An adversary can repeatedly jam along the path, forcing the base station to repeatedly reboostrap. To defend against this

DOS attack, we propose two mechanisms that both reduce the frequency of bootstrapping.

5.2.1 OHC proactive bootstrapping

The high redundancy of WSNs enables most sensor nodes to find another node near the failed node, for example, the failed node's neighbour or even nodes two to three hops away, to repair a path (Woo et al., 2003). If nodes near the path can be bootstrapped and refreshed with the OHC, then re-bootstrapping the OHC can be postponed and need not occur every time the path is changed. When base station B and nodes along the designated path n_1 to n_m bootstrap the initial OHC number, their neighbour nodes can receive these messages and receive the authenticated initial OHC number. Similarly, these nodes can also receive the refreshed OHC number. These nodes can be chosen for a new path and will be able to authenticate OHC numbers from the source node without rebootstrapping.

We can extend this scheme by proactively bootstrapping OHC numbers to nodes that are several hops away from node n_1 to n_m, B and S. When these nodes' neighbour nodes receive the bsp broadcast, they will rebroadcast the message $bsp|K$ to their neighbour nodes, where K is a propagation factor. Its value should be one or two. When a node receives this message, it reduces K by one. If the result is larger than one, that node will continue to rebroadcast the message with new K. In this way, the bsp message is flooded to neighbourhoods of the nodes on the path from B to S in a limited way, namely within $K + 1$ hops. Using the same method, the second message of μTESLA and messages for refreshing OHC number are also flooded to these expanded neighbourhoods. These nodes can join the new path and defend against PDoS attacks without rebootstrapping. This mechanism requires neighbourhood nodes to consume additional memory to save OHC values. Since these would not be used very often, they can be stored on more plentiful flash memory or EEPROM.

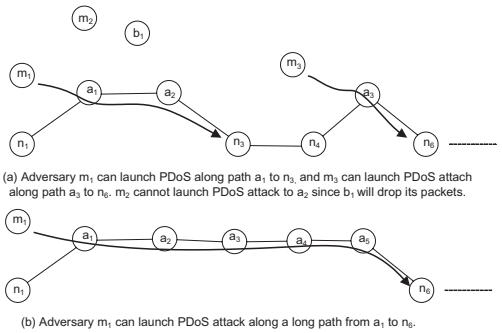
To successfully launch a DOS attack against proactive bootstrapping, an adversary has to destroy many more nodes and has to continually move from one place to another place and destroy nodes. This significantly increases the cost of the attack. The public key and/or μTESLA provide further protection against saving the wrong initial OHC number.

5.2.2 Lazy OHC bootstrapping

One observation is that modest changes in the path do not require immediate bootstrapping. When only one new node is added in the path, the OHC need not be bootstrapped in the new node(s) immediately. This is because the extent of a PDoS attack will be limited to only the new node(s). Other nodes that have V_S already set up can still verify packets and hence are still resilient to stopping PDoS attacks.

As given in Figure 4(a), when new nodes (a_1, a_2 and a_3) that have just joined the path are sparsely distributed, an adversary can flood only some of these nodes (a_1 and a_2), but cannot flood the other nodes (e.g. n_4) that are separated by old nodes. However, if several new nodes form a long path, as shown in Figure 4(b), a PDoS attack can cause more damage. To avoid the situation where a long path of new nodes exists, the bootstrapping process can be performed periodically, or when the base station finds that the number of new nodes in the path exceeds some threshold or when the length of a path formed by the new nodes exceeds some threshold.

Figure 4 Routing path repair. Nodes $n_1, n_2, \ldots, n_6$ are old nodes that already have an OHC number, and nodes a_1, $a_2, \ldots, a_5$ are new nodes in the path that don't have an OHC number



6 Evaluation

To evaluate the feasibility of our mechanism in current WSN platforms, we need to measure the resource consumption for generating, storing and verifying one-way hash chains in resource constrained sensor nodes. To do this, we implemented an OHC generation and verification algorithm on Berkeley MICA2 motes. Furthermore, to evaluate the practicality of our mechanism, it is important to understand its performance overhead, which is the extra delay introduced in communication. To do this, we simulated our solution in a multihop network. The overhead of bootstrapping is one message passed from base station B to node S in the public key approach, and two messages passed from B to S in the μTESLA approach. The OHC refresh also consists of two messages passed.

6.1 One-way hash chain verification

The Berkeley MICA2 mote has a 7.3 MHz processor with 128 KB flash memory, 4.0 KB RAM, and a Chipcon CC1000 radio at 19.2 Kbps. We adopt the method of generating OHCs by a block cipher encryption algorithm (Deng et al., 2003). To measure the resource requirements of the OHC verification function, we adopted the implementation of skipjack in TinySEC (Karlof et al., 2004).

We measured the time for one OHC verification operation to be 1.49 MS. Considering the slow speed of wireless links (19.2 kbps), this verification time of 1.49 ms is quite reasonable. This shows that the proposed mechanism for preventing PDoS attacks is a viable mechanism that can easily be supported by current sensor nodes such as motes.

If 8 byte OHC numbers are used, an intermediate node needs to store only 16 bytes for each transmission link. For nodes that are on many paths, we store the OHC numbers in flash memory, which has 128 KB, and cache frequently used OHCs in SRAM, of size 4 KB. According to Dai and

Han (2004), reading/writing a page costs only < 250 µs/14 ms. We think that 14 ms is short enough for a node to write data to flash during its non-I/O cycle. If a node has to receive/send a packet every tens of milliseconds, it will exhaust its battery in a few days.

Since a single OHC number is included in each packet, without counting setup overhead, the message overhead is 8 bytes per packet. This is less that the 14-byte per packet overhead of the SEF protocol (Ye et al., 2004). As suggested in (Karlof et al., 2004), we can use 4 bytes of MAC for end-to-end security. So the total security overhead is 12 bytes. Notice that end-to-end security overhead may still be required for SEF, since the Bloom filter only provides probabilistic security protection. node.

6.2 *One-way hash chain generation on a source node*

The method of generating and storing a long OHC in a sensor node is not straight forward. Naive algorithms require either too much memory to store every OHC number or too much time to compute the next OHC number. None of these algorithms are practical on resource-constrained sensor nodes. Recently, some efficient OHC generation algorithms for resource-constrained platforms have been proposed (Coppersmith and Jakobsson, 2002; Jakobsson, 2002; Sella, 2003). After a comparison of their performance, we implemented the fractal graph traversal algorithm (Coppersmith and Jakobsson, 2002) on Berkeley motes. This algorithm stores only some of the intermediate numbers, called pebbles, of an OHC and uses them to compute other numbers. If the size of an OHC is n (there are total n numbers in this OHC), the algorithm performs approximately $\frac{1}{2}\log_2 n$ one-way function operations to compute the next OHC number and requires a little more than $\log_2 n$ units of memory to save pebbles.

Another important factor is the length of an OHC that is needed for a source node. The typical length is between 2^{11} and 2^{22}. If the length of an OHC is 2^{22} and a node uses one OHC number per second, it will take more than a month to exhaust all numbers from this chain. Figure 5 shows the storage requirements for storing pebbles for different lengths of an OHC. This includes a skipjack-based one-way function and OHC generation based on (Coppersmith and Jakobsson, 2002). We see that a node needs about 930 bytes to maintain an OHC of length 2^{22}. This includes 256 bytes lookup table for skipjack, which can be shared with other applications.

Figure 6 shows the average time and maximum time required to generate one OHC number. The average time was measured as the total time for computing a complete OHC with the fractal traversal algorithm, and then averaged for generating a single number. As analysed in Jakobsson (2002), the maximum time for generating an OHC number is approximately equal to $\delta \times 0.5 \times \log_2 n$, where n is the size of the OHC and δ is the time for performing a one-way function. Here we chose δ as 1.49 ms. When the size of the OHC is 2^{22}, the implementation shows that a Berkeley mote MICA2 requires about 10.3 ms on average to generate an OHC and about 16.5 ms in the worst case. Considering that it takes about 40–50 ms to send a 36-byte packet on motes, we believe that this computing time for generating an OHC number is practical.

Figure 5 Memory consumption in one-way hash chain generation

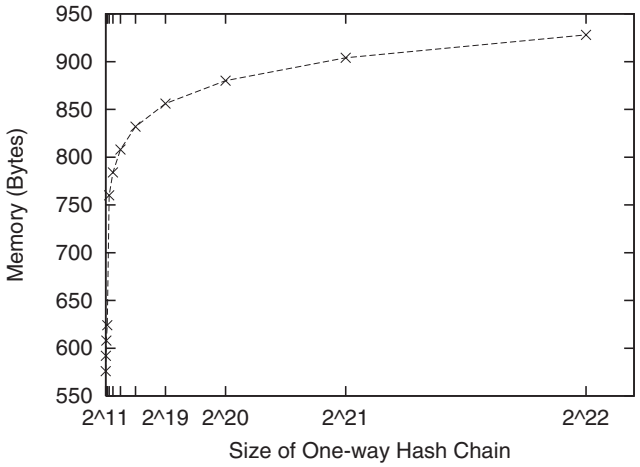
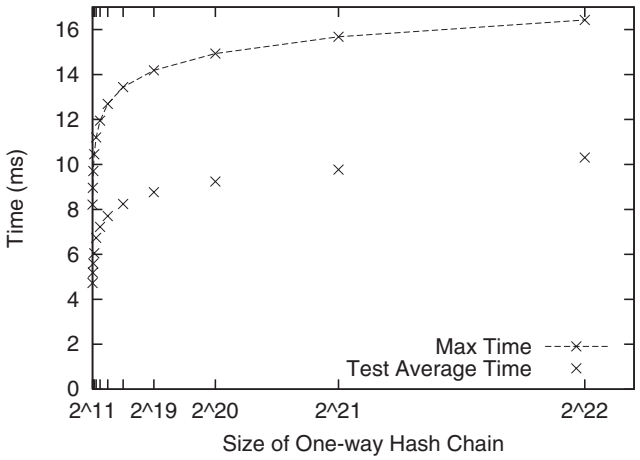


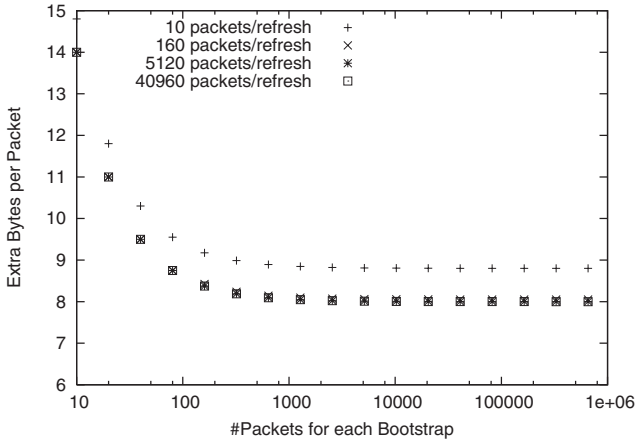
Figure 6 Time consumption in one-way hash chain generation



6.3 *Simulation of additional overhead*

In our solution, every data packet contains a OHC number, which is 8 bytes of overhead compared with the no anti-PDOS attack solution. In addition, the resetup of OHCs with the µTESLA protocol and refreshing broken OHCs also costs extra overhead. We simulated the total data overhead. We simulated OHC set up for from every 10 messages to every 10^6 messages, and we simulated refreshing the OHC for every 10 messages to every 4×10^5 messages. Figure 7 illustrates that when we perform OHC setup and refreshment very frequently, for exmple, for every 10 messages, the total data overhead is about 14 bytes per packet, which is the same as the SEF algorithm. However, if we perform these operations infrequently, for example, set up OHC every 10,000 messages and refresh OHC every 160 messages, the amortised overhead of our scheme will reduce to about 8 bytes per packet.

Figure 7 Average extra bytes of overhead per packet for OHC bootstrapping and refresh



6.4 Simulation of multihop data transmission delay overhead

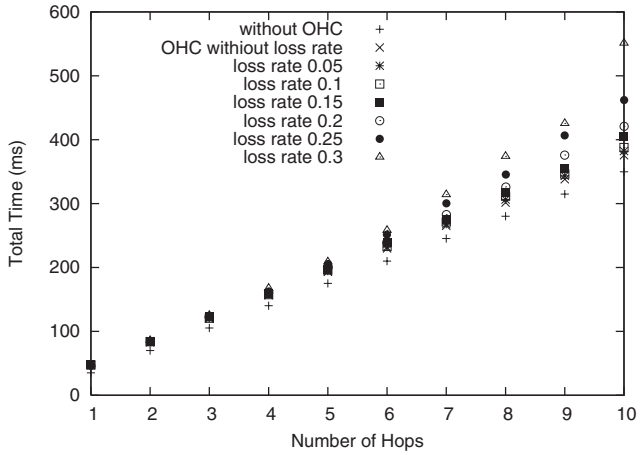
Using the OHC generation and verification times, we simulated the data transmission overhead of our scheme. In this experiment, a sensor node sends data to a base station via a path whose length varies from 1 to 10 hops. Data transmission time of each hop is randomly set between 30 and 40 ms. We simulated this experiment without our scheme and with our scheme. When our scheme was used, we experimented with different packet loss rates ranging from 0 to 0.3 at each hop. Our scheme adds overhead in data transmission by performing OHC generation at the source node and OHC verification at the intermediate nodes. We chose $\frac{1}{2} \times \log_2 2^{22} \times 1.5 = 16.5$ms as the overhead of OHC generation, which is almost the maximum time to generate a OHC number in a 2^{22} long OHC. The time needed for verification depends on the packet loss rate. In the presence of packet loss, intermediate nodes need to apply the one-way function F several times to validate a received packet. We manually introduce packet loss rates on each hop, ranging from 0 to 0.3. Note that the total end-to-end packet loss rate is adversely affected by multihop transmission. For example, if each hop has a loss rate of 0.3, then with 97% possibility a packet would not reach the base station via a 10-hop path. If each hop has a loss rate of 0.05, a packet only has a probability of 0.7 to reach the base station via a 7-hop path.

Figure 8 illustrates the overhead of our scheme for various packet loss rates. We see that when the packet loss rate is 0, our scheme requires an additional 16 ms in a 1-hop path and about 30 ms in a 10-hop path. If the packet loss rate is 0.05 and there are 7 hops from the source to the base station, our scheme adds only about 22 ms overhead (16 ms for OHC generation and 6 ms for OHC verification). We believe that this overhead is quite reasonable for current WSNs.

7 Discussion

Our OHC-based mechanism is applicable not just to unicast paths, but also can be extended to counteract PDoS attacks against a reliable end-to-end connection and multipath routing in WSNs. A reliable protocol would be useful for sending commands or even dynamic code updates to a sensor node from a base station. A typical reliable ARQ protocol will send and/or retransmit data and acknowledgements. Such a protocol is highly susceptible to a PDoS attack, since replaying data and/or duplicate acks is considered a legitimate part of the protocol. PDoS attacks can be inhibited if the two end points share two OHCs, one for (re)transmitting data packets in one direction and the other for (re)transmitting ack/nack packets in the reverse direction. Every packet sent by one end point contains a unique OHC number. Even if a packet is retransmitted, the retransmitted version of the packet has a new distinct OHC number. This allows an intermediate node to distinguish between a packet retransmitted by the source and a retransmitted packet replayed by an adversary, because the adversary cannot attach a valid OHC number in the replayed packet.

Figure 8 Delay overhead of OHC verification



Multipath routing (Deng et al., 2003; Ganesen et al., 2002; Karloff et al., 2002; Ye et al., 2005) improves the robustness and reliability of data communications in WSNs. Bootstrapping proceeds as before, except along multiple paths. If the multiple paths are disjoint, maintenance of OHC is similar to single path routing. An intermediate node forwards a packet only once. In addition, different paths can use different OHCs. But in interleaved multiple paths, an intermediate node may receive the same packet from different nodes and may forward them more than once. In this case, every node only forwards a packet containing the same OHC number a limited number of times equal to the number of paths, thereby forestalling a PDoS attack.

8 Conclusion

In WSNs, an adversary can launch with little effort a PDoS attack that will have a severe widespread effect on the WSN, disabling nodes on all branches downstream of the path, due to the tree-structured topology of WSNs. In this paper, we have proposed a lightweight and efficient mechanism using one-way hash chains that allows intermediate nodes to defend against PDoS attacks by detecting replayed and spurious packets. We have proposed a novel and robust set of mechanisms to bootstrap and maintain one-way hash chains given packet loss and topology

changes. Our implementations show that our scheme is feasible in current sensor network platforms and incurs modest overhead.

Acknowledgements

We would like to thank Dr. Helger Lipmaa for providing us the implementation of fractal graph traversal algorithm and thank Dr. Marco Gruteser for his helpful suggestions. We also would like to thank anonymous reviewers for their valuable comments.

References

Coppersmith, D. and Jakobsson, M. (2002) 'Almost optimal hash sequence traversal', *Sixth International Financial Cryptography 2002 (FC'02)*, Bermuda.

Dai, H. and Han, R. (2004) 'Elf: an efficient log-structured flash file system for micro sensor nodes', *Second International Conference on Embedded Networked Sensor Systems*, Baltimore, MD.

Deng, J., Han, R. and Mishra, S. (2003) 'The performance evaluation of intrusion-tolerent routing in wireless sensor networks', *IEEE Second International Workshop on Information Processing in Sensor Networks (IPSN'3)*, Palo Alto, CA.

Deng, J., Han, R. and Mishra, S. (2005) 'INSENS: intrusion-tolerant routing for wireless Sensor Networks', *ELsevier Journal on Computer Communications, Special Issue on Dependable Wireless Sensor Networks*, Vol. 29, Issue 2, pp.216–230.

Ganesan, D., Govindan, R., Shenkar, S. and Estrin, D. (2002) 'Highly resilient, energy efficient multipath routing in wireless sensor networks', *Mobile Computing and Communica-tion Review (MC2R)*, Vol. 1, No. 2.

Hu, Y., Perrig, A. and Johnson, D. (2003) 'Ariadne: a secure on-demand routing protocol for ad hoc networks', *Eighth Annual International Conference on Mobile Computing and Networking (Mobicom'02)*.

Hu, Y-C., Perrig, A. and Johnson, D.B. (2003) 'Packet leashes: a defense against Wormhole Attacks in Wireless Networks', *IEEE Infocom 2003*.

Jakobsson, M. (2002) 'Fractal hash sequence representation and traversal', *2002 IEEE International Symposium on Information Theory (ISIT'02)*, Switzerland.

Karlof, C., Li, Y. and Polastre, J. (2002) 'Arrive: algorithm for robust routing in volatile environments', *Technical Report UCBCSD-02-1233*, Computer Science Department, University of California at Berkeley.

Karlof, C., Sastry, N. and Wagner. D. (2004) 'TinySec: a link LayerSecurity architecture for wireless sensor networks', *Second ACMConference on Embedded Networked Sensor Systems (SenSys 2004)*, November, pp.162–175.

Karlof, C. and Wagner, D. (2003) 'Secure routing in wireless sensor networks: attacks and countermeasures', *Ad hoc networks*, Vol. 1, Nos. 2–3.

Lamport, L. (1979) 'Constructing digital signatures from one-way function', *Technical Report SRI-CSL-98*, SRI International.

Liu, D. and Ning, P. (2003) 'Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks', *Tenth Annual Network and Distributed System Security Symposium*, San Diego, CA.

Lu, K., Fan, J., Greco, J., Wu, D., Todorovic, S. and Nucci, A. (2005) 'A novel anti-ddos system for large-scale internet', *ACM SIGCOMM 2005, Work-in Progress Session*, Philadelphia, PA.

Malan, D.J., Welsh, M. and Smith, M.D. (2004) 'A public-key infrastructure for key disribution in tinyos based on elliptic curve cryptography', *First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, Santa Clara, CA.

Perrig, A., Canetti, R., Tygar, J. and Song, D.X. (2000) 'Efficient authentication and signing of multicast streams over lossy channels', *IEEE Sysposium on Security and Privacy*, pp.56–73.

Perrig, A., Szewczyk, R., Wen, V., Culler, D. and Tygar, J. (2002) 'Spins: security protocols for sensor networks', *Wireless Networks Journal (WINET)*, Vol. 8, No. 5, pp.521–534.

Przydatek, B., Song, D. and Perrig, A. (2002) 'SIA: secure information aggregation in sensor networks', *ACM SenSys'03*, Los Angeles, CA.

Sella, Y. (2003) 'On the computation-storage trade-offs of hash chain traversal', *Seventh International Financial Cryptography Conference*, Le Gosier, Guadeloupe.

Woo, A., Tong, T. and Culler, D. (2003) 'Taming the underlining challenges of reliable multihop routing in sensor networks', *SenSys'03*, Los Angeles, CA.

Wood, A. and Stankovic, J. (2002) 'Denial of service in sensor networks', *IEEE Computer*, Vol. 35, No. 10, pp.54–62.

Wood, A.D., Stankovic, J.A. and Son, S.H. (2003) 'Jam: a jammed-area mapping service for sensor networks', *Twenty Fourth IEEE Real-time Systems Symposium (RTSS'03)*, Cancun, Mexico.

Ye, F., Zhong, G., Lu, S. and Zhang, L. (2005) 'Gradient broad-cast: a robust data delivery protocol for large scale sensor networks', *ACM Wireless Networks (WINET)*, Vol. 11, No. 3, pp.285–298.

Ye, F., Luo, H., Lu, S. and Zhang, L. (2004) 'Statistical en-route detection and filtering of injected false data in sensor networks', *IEEE INFOCOM 2004*.

Zhao, J. and Govindan, R. (2003) 'Understanding packet delivery performance in dense wireless sensor networks', *SenSys'03*, Los Angeles, CA.

Zhou, G., He, T., Krishnamurthy, S. and Stankovic, J.A. (2004) 'Impact of radio irregularity on wireless sensor networks', *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys'04)*, Bostom, MA.

Zhu, S., Setia, S. and Jajodia, S. (2003) 'LEAP: efficient security mechanisms for large-scale distributed sensor networks', *CCS'03*, Washington, DC.

Zhu, S., Setia, S., Jajodia, S. and Ning, P. (2004) 'An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks', *2004 IEEE Symposium on Security and Privacy*, Privacy, Oakland, CA.