A Resilient Packet-Forwarding Scheme against Maliciously Packet-Dropping Nodes in Sensor Networks

Suk-Bok Lee and Yoon-Hwa Choi
Department of Computer Engineering
Hongik University
121-791 Seoul, Korea
{sblee, yhchoi}@cs.hongik.ac.kr

ABSTRACT

This paper focuses on defending against compromised nodes' dropping of legitimate reports and investigates the misbehavior of a maliciously packet-dropping node in sensor networks. We present a resilient packet-forwarding scheme using *Neighbor Watch System* (NWS), specifically designed for hop-by-hop reliable delivery in face of malicious nodes that drop relaying packets, as well as faulty nodes that fail to relay packets. Unlike previous work with multipath data forwarding, our scheme basically employs single-path data forwarding, which consumes less power than multipath schemes. As the packet is forwarded along the single-path toward the base station, our scheme, however, converts into multipath data forwarding at the location where NWS detects relaying nodes' misbehavior. Simulation experiments show that, with the help of NWS, our forwarding scheme achieves a high success ratio in face of a large number of packet-dropping nodes, and effectively adjusts its forwarding style, depending on the number of packet-dropping nodes en-route to the base station.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security, Algorithm, Reliability

Keywords

Sensor Network Security, Reliable Delivery, Packet-dropping Attacks, Secure Routing

1. INTRODUCTION

Wireless sensor networks consist of hundreds or even thousands of small devices each with sensing, processing, and communicating capabilities to monitor the real-world environment. They are envisioned to play an important role in a wide variety of areas ranging from critical military-surveillance applications to forest fire monitoring and the building security monitoring in the near future. In such a network, a large number of sensor nodes are distributed to monitor a vast field where the operational conditions are harsh or even hostile. To operate in such environments, security is an important aspect for sensor networks and security mechanisms should be provided against various attacks such as node capture, physical tampering, eavesdropping, denial of service, etc [23, 33, 38].

Previous research efforts against *outsider* attacks in key-management schemes [4, 13, 32] and secure node-to-node communication mechanisms [24, 32] in sensor networks are well-defined. Those security protections, however, break down when even a single legitimate node is compromised. It turns out to be relatively easy to compromise a legitimate node [14], which is to extract all the security information from the captured node and to make malicious code running for the attacker's purpose.

Even a small number of compromised nodes can pose severe security threats on the entire part of the network, launching several attacks such as dropping legitimate reports, injecting bogus sensing reports, advertising inconsistent routing information, eavesdropping in-network communication using exposed keys, etc. Such disruption by the *insider* attacks can be devastating unless proper security countermeasures against each type of attacks are provided. In reality, detecting all of the compromised nodes in the network is not always possible, so we should pursue *graceful degradation* [35], with a small number of compromised nodes. The fundamental principle for defense against the insider attacks is to restrict the security impact of a node compromise as close to the vicinity of the compromised node as possible.

When the attacker compromises a legitimate node, it may first try to replicate the captured node indefinitely with the same ID and spread them over the network. Against such attacks, a distributed detection mechanism (based on emergent properties [11]) has been proposed by Parno *et al.* [31]. In addition, Newsome *et al.* [30] have presented the techniques that prevent the adversary from arbitrarily creating new IDs for nodes.

Using cryptographic information obtained from a captured node, attackers can establish pairwise keys with any legitimate nodes in order to eavesdrop communication any-

SASN'06, October 30, 2006, Alexandria, Virginia, USA.

where in the network. Localized key-establishment scheme by Zhu *et al.* [46] is a good solution against such an insider attack. Since the scheme does not allow a cloned node (by inside-attackers) to establish pairwise keys with any legitimate nodes except the neighbors of the compromised nodes, the cryptographic keys extracted from the compromised node are of no use for attackers.

Compromised nodes can also inject false sensing reports to the network (i.e. report fabrication attacks [39]), which causes false alarms at the base station or the aggregation result to far deviate from the true measurement. Proposed en-route filtering mechanisms [8, 39, 41, 44, 47] that detect and drop such false reports effectively limit the impact of this type of attacks. Also, proposed secure aggregation protocols [34, 40] have addressed the problem of false data injection, and they ensure that the aggregated result is a good approximation to the true value in the presence of a small number of compromised nodes.

Advertising inconsistent routing information by compromised nodes can disrupt the whole network topology. Hu *et al.* [19, 20] have proposed SEAD, a secure ad-hoc network routing protocol that uses efficient one-way hash functions to prevent any inside attackers from injecting inconsistent route updates. A few secure routing protocols [6, 27] in sensor networks have been proposed to detect and exclude the compromised nodes injecting inconsistent route updates.

Compromised nodes also can silently drop legitimate reports (i.e. selective forwarding attacks [23]), instead of forwarding them to the next-hop toward the base station. Since data reports are delivered over multihop wireless paths to the base station, even a small number of strategically-placed packet-dropping nodes can deteriorate the network throughput significantly. In order to bypass such nodes, most work on secure routing and reliable delivery in sensor networks relies on multipath forwarding scheme [5, 6, 7, 10], or interleaved-mesh forwarding scheme [26, 29, 39, 42].

Among the *insider* attacks described above, this paper focuses on defense against compromised nodes' dropping of legitimate reports and we present a resilient packet-forwarding scheme using *Neighbor Watch System* (NWS) against maliciously packet-dropping nodes in sensor networks. We investigate the misbehavior of a maliciously packet-dropping node and show that an acknowledgement (ACK) that its packets were correctly received at the next-hop node does not guarantee reliable delivery from the security perspective.

NWS is specifically designed for hop-by-hop reliable delivery in face of malicious nodes that drop relaying packets, as well as faulty nodes that fail to relay packets. Unlike previous work [10, 29, 42] with multipath data forwarding, our scheme basically employs single-path data forwarding, which consumes less power than multipath schemes. As the packet is forwarded along the single-path toward the base station, our scheme, however, converts into multipath data forwarding at the location where NWS detects relaying nodes' misbehavior.

NWS exploits the dense deployment of large-scale static sensor networks and the broadcast nature of communication pattern to overhear neighbors' communication for free.

The contribution of this paper is two-fold. First, we investigate the misbehavior of a maliciously packet-dropping node and propose a resilient packet-forwarding scheme, which basically employs single-path data forwarding, in face of such nodes, as well as faulty nodes. Second, our scheme can work with any existing routing protocols. Since it is designed not for securing specific protocols but for universal protocols, it can be applied to any existing routing protocols as a security complement.

The rest of paper is organized as follows. Background is given in Section 2. We present our resilient packet-forwarding scheme in Section 3. An evaluation of the scheme is given and discussed in Section 4. We present conclusions and future work in Section 5.

2. BACKGROUND

2.1 Network Model

Sensor networks typically comprise one or multiple base stations and hundreds or thousands of inexpensive, small, static, and resource-constrained nodes scattered over a wide area. An inexpensive sensor node cannot afford tamper-resistant packaging. We assume that a large number of sensor nodes are deployed in high density over a vast field, such that the expected degree of a node is high; each sensor has multiple neighbors within its communication range. Sensing data or aggregated data are sent along the multihop route to the base station. We assume that each sensor node has a constant transmission range, and communication links are bidirectional.

Our sensor network model employs a key-establishment scheme that extends the one in LEAP [46] where the impact of a node compromise is *localized* in the immediate neighborhood of the compromised node, and our scheme is based on it. To evolve from LEAP, we will describe it briefly in Section 2.4.

2.2 Threat Model

The attacks launched from outsiders hardly cause much damage to the network, since the rouge node, which does not possesses the legitimate credentials (e.g. the predistributed key ring from the key pool [13]), fails to participate in the network. On the other hand, there may be multiple attacks from insiders (e.g. dropping legitimate reports, injecting false sensing reports, advertising inconsistent route information, and eavesdropping in-network communication using exposed keys, etc), and the *combination* of such attacks can lead to disruption of the whole network. Thus, proper security countermeasures (specifically designed to protect against each type of the attacks) should be provided.

Among them, in this paper, we focus on defending against compromised nodes' dropping of legitimate reports; Other attacks mentioned above are effectively dealt with by several proposed schemes as described in the previous section.

We consider a packet-dropping node as not merely a faulty node, but also an arbitrarily malicious node. Some previous work [3, 29, 36] on reliable delivery uses an acknowledgement (ACK) that its packets were correctly received at the next-hop node, in order to find out unreliable links. However, in the presence of maliciously packet-dropping nodes, simply receiving ACK from a next-hop node does not guarantee that the packet will be really forwarded by the next-hop node. For example, node u forwards a packet to compromised node v, and node u waits for ACK from node v. Node v sends back ACK to node u, and then node v silently drops the packet. This simple example shows that receiving ACK is not enough for reliable delivery in face of maliciously packet-dropping nodes.

For more reliability, we should check whether the next-hop node really forwards the relaying packet to its proper next-hop node. Fortunately, due to the broadcast nature of communication pattern in sensor networks, we can overhear neighbors' communication for free (for now per-link encryption is ignored). After forwarding a packet to next-hop node v and buffering recently-sent packets, by listening in on node v's traffic, we can tell whether node v really transmits the packet. *Watchdog* [28] mechanism (extension to DSR [22]), *implicit ACK* in M^2RC [29], and *local monitoring* in *DICAS* [25] detect misbehaving nodes in this way. However, this kind of simple overhearing schemes does not guarantee reliable delivery, either.

With arbitrarily malicious nodes, we should be assured that the node, to which the next-hop node forwards the relaying packet, is really a neighbor of the next-hop node. For example, node u forwards a packet to compromised node v, and node u listens in on node v's traffic to compare each overheard packet with the packet in the buffer. Node v transmits the relaying packet whose intended next-hop id marked with any id in the network such as x that is not a neighbor of v. Then node u overhears this packet from node v, and considers it forwarded correctly despite the fact that none actually receives the packet. The packet is eventually dropped without being detected. We refer to this attack as *blind letter attack*.

We consider packet-dropping attacks to be addressed in this paper as ones ranging from the naive case (e.g. a faulty node) to the most malicious one (e.g. a node launching blind letter attack). We focus on developing a solution to such attacks.

2.3 Notation

We use the following notation throughout the paper:

- u, v are principals, such as communicating nodes.
- R_u is a random number generated by u.
- f_K is a family of pseudo-random function [12].
- $MAC(K, M_1|M_2)$ denotes the message authentication code (MAC) of message - concatenation of M_1 and M_2, with MAC key K.

2.4 Key-Establishment Scheme in LEAP

LEAP supports the establishment of four types of keys for each sensor node - an *individual key* shared with the base station, a *pairwise key* shared with its neighbor, a *cluster key* shared with its surrounding neighbors, and a *group key* shared by all the nodes in the networks.

It assumes that the time interval T_{est} for a newly deployed sensor node to complete the *neighbor discovery phase* (e.g. tens of seconds) is smaller than the time interval T_{min} that is necessary for the attacker to compromise a legitimate node (i.e. $T_{min} > T_{est}$). Some existing work [1, 39] has made similar assumptions, which are believed to be reasonable.

The four steps for a newly added node u to establish a pairwise key with each of its neighbors are as follows:

1. Key Pre-distribution. Each node u is loaded with a common initial key K_I, and derives its master key $K_u = f_{K_I}(u)$.
2. Neighbor Discovery. Once deployed, node u sets up a timer to fire after time T_{min}, broadcasts its id, and waits for each neighbor v's ACK. The ACK from v is authenticated using the master key K_v of node v. Since node u knows K_I, it can derive $K_v = f_{K_I}(v)$.

$$u \longrightarrow *: \quad u, R_u.$$
$$v \longrightarrow u: \quad v, MAC(K_v, R_u|v).$$

3. Pairwise Key Establishment. Node u computes its pairwise key with v, K_{uv}, as $K_{uv} = f_{K_v}(u)$. Node v also computes K_{uv} in the same way. K_{uv} serves as their pairwise key.
4. Key Erasure. When its timer expires, node u erases K_I and all the master keys of its neighbors. Every node, however, keeps its own master key, in order to establish pairwise keys with later-deployed nodes.

Once erasing K_I, a node will not be able to establish a pairwise key with any other nodes that have also erased K_I. Without K_I, a cloned node (by an attacker compromising a legitimate node after T_{min}) fails to establish pairwise keys with any nodes except the neighbors of the compromised node. In such a way, LEAP localizes the security impact of a node compromise.

3. A RESILIENT PACKET-FORWARDING SCHEME USING NEIGHBOR WATCH SYSTEM

In this section, we present our resilient packet-forwarding scheme using Neighbor Watch System (NWS). NWS works with the information provided by Neighbor List Verification (NLV) to be described in Section 3.2.

3.1 Neighbor Watch System

Our scheme seeks to achieve hop-by-hop reliable delivery in face of maliciously packet-dropping nodes, basically employing single-path forwarding. To the best of our knowledge, proposed works so far rely on multipath forwarding or diffusion-based forwarding, exploiting a large number of nodes in order to deliver a single packet. ACK-based technique is not a proper solution at all as explained in the previous section.

With NWS, we can check whether the next-hop node really forwards the relaying packet to the actual neighbor of the next-hop node. The basic idea of our scheme is as follows:

1. **Neighbor List Verification.** After deployment, during neighbor discovery phase, every node u gets to know of not only its immediate neighbors, but also the neighbors' respective neighbor lists (i.e. u's neighbors' neighbor lists). The lists are verified using *Neighbor List Verification* to be described in Section 3.2. Every node stores its neighbors' neighbor lists in the neighbor table.
2. **Packet Forwarding to Next-hop.** If node u has a packet to be relayed, it buffers the packet and forwards the packet (encrypted with cluster key of node u so that neighbors of node u can overhear it) to its next-hop node v. As in LEAP, a cluster key is a key shared by a node and all its neighbors, for passive participation.

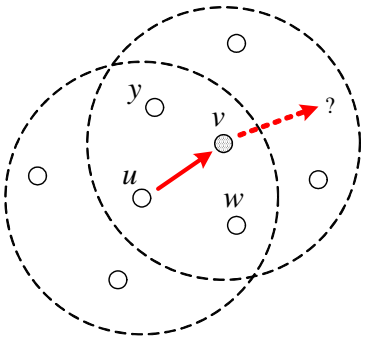


Figure 1: Neighbor Watch System. Sub-watch nodes w and y, as well as primary-watch node u listen in on v's traffic.

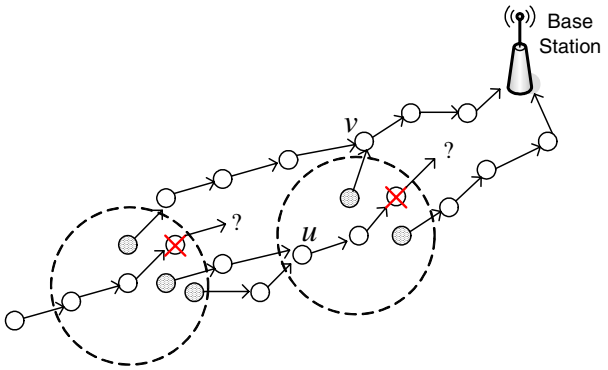


Figure 2: An example of our packet-forwarding scheme. Only the nodes that relay the packet are presented. With the help of sub-watch nodes (grey ones), our scheme bypasses two packet-dropping nodes en-route to the base station.

3. **Designation of Watch Nodes.** Overhearing the packet from node u to node v, among neighbors of node u, the nodes that are also neighbors of node v (in Figure 1, nodes w and y) are designated as *sub-watch nodes* and store the packet in the buffer. Other nodes (that are not neighbors of node v) discard the packet. Node u itself is a *primary-watch node*. A primary-watch node knows which nodes are sub-watch nodes, since every node has the knowledge of not only its neighbors but also their respective neighbor lists.

4. **Neighbor Watch by Sub-Watch Node.** Sub-watch nodes w and y listen in on node v's traffic to compare each overheard packet with the packet in the buffer. To defend against blind letter attack, each of them also checks whether the packet's intended next-hop is a verified neighbor of node v, by looking up the neighbor table. If all correct, the packet in the buffer is removed and the role of the sub-watch node is over.
   If the packet has remained in the buffer for longer than a certain timeout, sub-watch nodes w and y forward the packet (encrypted with their respective cluster keys) to their respective next-hop nodes other than node v. Then the role of a sub-watch node is over (each of them is now designated as a primary-watch node for the packet it has forwarded).

5. **Neighbor Watch by Primary-Watch Node.** Primary-watch node u does the same job as sub-watch nodes. The only difference, however, is that it listens in on not only node v's traffic, but also sub-watch nodes w's and y's. If the packet is correctly forwarded on by at least one of them (nodes v, w, or y), primary-watch node u removes the packet in the buffer and the role of the primary-watch node is over.
   Otherwise, after a certain timeout, primary-watch node u forwards the packet (encrypted with its cluster key) to its next-hop other than node v.

As the packet is forwarded on, this procedure (except for Neighbor List Verification) of NWS is performed at each hop so that hop-by-hop reliable delivery can be achieved with mainly depending on single-path forwarding. On the other hand, in the previous approaches [29, 39, 42], when forwarding a packet, a node broadcasts the packet with no designated next-hop, and all neighbors with smaller costs[1] or within a specific geographic region continue forwarding the packet anyway. For example, in Figure 1, if nodes v, w, and y have smaller costs than node u in the previous approaches, they all forward[2] the packet from node u. In our scheme, however, sub-watch nodes w and y are just on watch in designated next-hop node v, instead of unconditionally forwarding the packet. If no packet-dropping occurs en-route to the base station, the packet may be forwarded along single-path all the way through.

However, a packet-dropping triggers the multipath forwarding for the dropped packet. If the designated next-hop node v in Figure 1 has not forwarded the relaying packet to its certified neighbor by a certain timeout, sub-watch nodes w and y forward the packet to their respective next-hop. At the point, the packet is sent over multiple paths. Since the location where the packet-dropping occurs is likely in an unreliable region, this prompt reaction of the conversion to multipath forwarding augments the robustness in our scheme. The degree of multipath depends on the number of the sub-watch nodes. Figure 2 shows an example of our packet-forwarding scheme, bypassing two packet-dropping nodes en-route to the base station. If a node utilizes a cache [16, 21] for recently-received packets, it can suppress the same copy of previously-received one within a certain timeout, as nodes u and v in Figure 2.

Our scheme requires that a relaying packet should be encrypted with a cluster key of a forwarding node, in order that all its neighbors can decrypt and overhear it. In fact, per-link encryption provides better robustness to a node compromise, since a compromised node can decrypt only the packets addressed to it. Thus, there exists a tradeoff between resiliency against packet-dropping and robustness to a node compromise. However, encryption with a cluster key provides an intermediate level of robustness to a node compromise [24] (a compromised node can overhear only its immediate neighborhood), and also supports *local broadcast* (i.e. resiliency against packet-dropping), so that we can achieve graceful degradation in face of compromised nodes.

[1]The cost at a node is the minimum energy overhead to forward a packet from this node to the base station.

[2]It is the broadcast transmission with no designated next-hop, and, if needed, the packet should be encrypted with a cluster key in order for all neighbors to overhear it.

To make our scheme work (against blind letter attack), we must address the problem of how a node proves that it really has the claimed neighbors. It is the identical problem of how a node verifies the existence of its neighbors' neighbors. Apparently, a node has the knowledge of its direct neighbors by *neighbor discovery* and *pairwise key establishment* phases. However, in the case of two-hop away neighbors, as in Figure 1, malicious node v can inform its neighbor u that it also has neighbor node x (any possible id in the network) which in fact is not a neighbor of node v. Node u has to believe it, since node x is not a direct neighbor of node u, and only the node v itself knows its actual surrounding neighbors. Then, how do we verify the neighbors' neighbors? The answer to this critical question is described in the next subsection.

3.2 Neighbor List Verification

To verify neighbors' neighbors, we present *Neighbor List Verification* (NLV) which extends the pairwise key establishment in LEAP. During neighbor discovery in LEAP, two messages are exchanged between neighbors to identify each other. On the other hand, NLV adopts *three-way handshaking* neighbor discovery, in order to identify not only communicating parties but also their respective neighbors.

NLV has two cases of neighbor discovery. One is that neighbor discovery between two nodes that are both still within the initial T_{min}[3] (referred as *pure* nodes). The other is that neighbor discovery between a newly-deployed node within the initial T_{min} and an existing node over the initial T_{min} (referred as an *adult* node).

Neighbor Discovery between Pure Nodes. Neighbor list verification process between pure nodes is quite simple. If a pure node broadcasts its neighbor list before the elapse of its initial T_{min}, we can accept the list as verifiable. Thus, the key point here is to keep track of each other's T_{min}, and to make sure that both broadcast their respective neighbor lists before their respective T_{min}. The following shows the three-way handshaking neighbor discovery between pure node u and v:

$$u \longrightarrow * : \quad u, R_u.$$
$$v \longrightarrow u : \quad \underbrace{v, T_v, R_v}_{M_v}, MAC(K_v, R_u|K_u|M_v).$$
$$u \longrightarrow v : \quad \underbrace{u, T_u}_{M_u}, MAC(K_{uv}, R_v|M_u).$$

where T_v and T_u are the amount of time remaining until T_{min} of v and T_{min} of u, respectively. Once deployed, node u sets up a timer to fire after time T_{min}. Then, it broadcasts its id, and waits for each neighbor v's ACK. The ACK from every neighbor v is authenticated using the master key K_v of node v. Since node u knows K_I[4], it can derive $K_v = f_{K_I}(v)$. The ACK from node v contains T_v, the amount of time remaining until T_{min} of node v. If T_v is a non-zero value, node v claims to be a pure node. K_u in MAC proves node v to be a pure node, since pure node v should know K_I and derive $K_u = f_{K_I}(u)$. Node u records $\tilde{T_v}$ (T_v added to the current time of node u) in the entry for node v in the neighbor table. Node u computes its pairwise key with v, $K_{uv} = f_{K_v}(u)$.[5] Node u also generates $MAC(K_v, v|u)$ (which means that v certifies u as an immediate neighbor), and stores it as a *certificate*.

The ACK from node u also contains T_u, the amount of time remaining until T_{min} of u. This ACK is authenticated using their pairwise key K_{uv}, which proves node u a pure node and u's identity. Node v then records $\tilde{T_u}$ (T_u added to the current time of v) in the entry for u in the neighbor table. It also generates $MAC(K_u, u|v)$ and stores it as a certificate. Then, the three-way handshaking is done.

Every pure node u broadcasts its neighbor list just prior to T_{min} of u. Each receiving neighbor v checks whether the receiving time at v is prior to $\tilde{T_u}$ in the neighbor table. If yes, the neighbor list of u is now certified by each neighbor v.

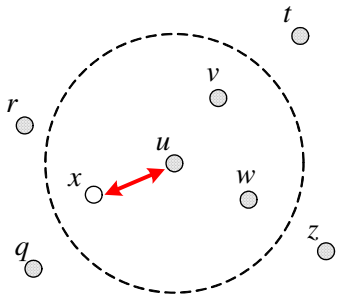


Figure 3: Neighbor Discovery between Pure node x and Adult node u. Grey and white nodes represent adult and pure nodes, respectively.

Neighbor Discovery between A Pure Node and An Adult node. After most nodes have completed *bootstrapping* phase, new nodes can be added in the network. Consider Figure 3. The issue here is how adult node u can assure its existing neighbors (v and w) of the existence of its newly-added neighbor x. This is a different situation from the above *neighbor list verification* case between two pure nodes. Thus, the messages exchanged during the three-way handshaking are somewhat different in this case. The following shows the three-way handshaking neighbor discovery between pure node x and adult node u:

$$x \longrightarrow * : \quad x, R_x.$$
$$u \longrightarrow x : \quad \underbrace{u, T_u, R_u, v, \overbrace{MAC(K_v, v|u)}^{certificate}, w, \overbrace{MAC(K_w, w|u)}^{certificate}}_{M_u}, MAC(K_u, R_x|M_u).$$
$$x \longrightarrow u : \quad \underbrace{x, T_x, \overbrace{MAC(K_x, x|u)}^{certificate}, v, \overbrace{MAC(K_v, x|u)}^{one-time\ cert.}, w, \overbrace{MAC(K_w, x|u)}^{one-time\ cert.}}_{M_x}, MAC(K_{xu}, R_u|M_x).$$

Newly-added node x sets up a timer to fire after time T_{min}. Then, it broadcasts its id, and waits for each neighbor u's

[3] T_{min} is the time interval, necessary for the attacker to compromise a legitimate node as in LEAP [46].

[4] Each node u is loaded with a common initial key K_I, and derives its master key $K_u = f_{K_I}(u)$. After time T_{min}, node u erases K_I and all the master keys of its neighbors.

[5] Node v also computes K_{uv} in the same way. K_{uv} serves as their pairwise key.

ACK. The ACK from every neighbor u is authenticated using the master key K_u of node u. Since node x knows K_I, it can derive $K_u = f_{K_I}(u)$. The ACK from node u contains T_u, the amount of time remaining until T_{min} of u. If T_u is zero, node u is an adult node that may already have multiple neighbors as in Figure 3. Node u reports its certified neighbor list (v and w) to x by including their respective certificates in the ACK. Node x verifies u's neighbor list by examining each certificate, since x can generate any certificate with K_I. If all correct, x computes its pairwise key with u, $K_{xu} = f_{K_u}(x)$. Node x also generates $MAC(K_u, u|x)$ and stores it as a certificate.

The ACK from x also contains T_x, the amount of time remaining until T_{min} of x. This ACK is authenticated using their pairwise key K_{xu}, which proves node x a pure node and x's identity. Node u then records $\check{T}_x$ (T_x added to the current time of u) in the entry for x in the neighbor table. Since adult node u cannot generate $MAC(K_x, x|u)$ by itself, pure node x provides the certificate for u in the ACK. Node x also provides *one-time certificates*[6] for each of u's certified neighbors (v and w). Then, the three-way handshaking is done.

After that, adult node u broadcasts one-time certificates (from newly-discovered pure node x), in order to assure u's existing neighbors (v and w) of the discovery of new neighbor x. The packet containing one-time certificates is as follows:

$$u \longrightarrow * : \underbrace{u, x, v, \overbrace{MAC(K_v, x|u)}^{one-time\ cert.}, w, \overbrace{MAC(K_w, x|u)}^{one-time\ cert.}, K_u^A}_{M_u}$$
$$, MAC(K_u^c, M_u).$$

where x is a new neighbor of u, K_u^A is a local broadcast authentication key in u's one-way key chain, K_u^c is the cluster key of u. Each receiving neighbor v of u verifies u's new neighbor x by examining the one-time certificate designated for v, $MAC(K_v, x|u)$[6]. If ok, node x is now certified by each neighbor v of u. Then, one-time certificates can be erased, since they are of no use any more.

Broadcast authentication only with symmetric keys such as cluster key K_u^c fails to prevent an *impersonation attack*, since every neighbor of u shares the cluster key of u. Thus, we employ the reverse disclosure of one-way key chain K_u^A as in LEAP.

Just prior to T_{min} of x, pure node x broadcasts its neighbor list. Each receiving neighbor u of x checks whether the receiving time at u is prior to $\check{T}_x$ in the neighbor table. If yes, the neighbor list of x is now certified by each neighbor u.

In summary, through the proposed three-way handshaking neighbor discovery process, pure node u identifies each immediate neighbor v and v's certified neighbor list (if v is an adult node), and keeps track of T_{min} of v. Just prior to T_{min} of u, node u broadcasts its direct neighbor list so that every neighbor of u accepts the list as verifiable. Then, node u becomes an adult node. After that, if newly-added node x initiates neighbor discovery with adult node u, node u identifies pure node x, keeps track of T_{min} of x, provides u's certified neighbor list to x, and, in return, takes one-time certificates from x. Node u then broadcasts these one-time certificates, in order to assure u's existing neighbors of the discovery of new neighbor x. Thus, every time adult node u discovers newly-added node x through three-way handshaking, node u informs (by broadcasting) its existing neighbors of the discovery of new neighbor x. Also, whenever receiving neighbor list information from pure neighbor x, node u checks whether the receiving time at u is prior to $\check{T}_x$ in the neighbor table. If yes, u now accepts the neighbor list of x as verifiable.

[6]One-time certificate, for instance $MAC(K_v, x|u)$, assures v that x is an immediate neighbor of u. It is generated by pure node x with master key of v.

Table 1: An example of the Neighbor Table of u.

Neighbor ID	Certificate	Verified Neighbor List
v	$MAC(K_v, v\|u)$	u, w, t
w	$MAC(K_w, w\|u)$	u, v, z
x	$MAC(K_x, x\|u)$	u, r, q

Through the above neighbor list verification in the bootstrapping phase, every node gets the knowledge of its neighbors' certified neighbors. Our Neighbor Watch System makes use of this information to prevent blind letter attack. With this knowledge, watch nodes are able to check whether the relaying packet's intended next-hop is a verified neighbor of the forwarding node.

3.3 Neighbor Table Maintenance

The information obtained through neighbor list verification (e.g. its direct neighbors, corresponding certificates, neighbors' neighbor lists, etc) is stored in the neighbor table of each node. Table 1 shows an example of the neighbor table of node u. In densely-deployed sensor networks, the expected degree of a node is high. However, in this example, for simplicity, node u has only three neighbors v, w, and x as in Figure 3.

The entries in the neighbor table are accessed and maintained with immediate neighbor IDs. For example, if node u overhears the packet sent from w to v, node u begins to listen in on v's traffic as a sub-watch node (since the neighbor table of u has both v's and w's entries in it). Unless v forwards the packet to a node of the *Verified Neighbor List* in v's entry by a certain timeout, sub-watch node u will forward the packet to its next-hop other than v; many existing routing protocols [5, 18, 21, 27, 37, 43] enable each node to maintain multiple potential next-hop. Once forwarding the packet, sub-watch node u becomes a primary-watch node and begins to listen in on its next-hop's traffic as described above.

If newly-added node y initiates the three-way handshaking with u, node u provides its neighbor list to y by sending certificates in the neighbor table. Node u, in return from node y, takes the certificate for y and one-time certificates for u's existing neighbors. Then, node u stores the certificate in the new entry for y. However, node u does not store the one-time certificates but broadcasts them to its neighbors. If new neighbor y broadcasts its neighbor list within T_{min}, node u stores the list in the entry for y.

If node u is compromised, not only cryptographic key information but also certificates in the neighbor table are exposed. However, the attacker cannot misuse these certificates for other purposes. Since a certificate only attests neighborship between two specific nodes, it cannot be applied to any other nodes. In fact, it can be made even public. However, *colluding* nodes can deceive a pure node anyway,

by fabricating a bogus certificate. We will describe this limitation in Section 4.4.

4. EVALUATION

In this section, we evaluate the communication and storage cost, and analyze the security of our resilient forwarding scheme (Neighbor Watch System) as well as Neighbor List Verification. We then present the simulation results of our forwarding scheme.

4.1 Communication Cost

Unlike the previously proposed diffusion-based reliable-forwarding schemes [21, 29, 39, 42] that exploit a large number of nodes to deliver a single packet, our scheme requires only the designated next-hop node to relay the packet, under the supervision of watch nodes. We note that, like overhearing by watch nodes in our scheme, those diffusion-based schemes require each node to listen to all its neighbors, since they forward a packet by broadcasting with no designated next-hop. With a smaller number of relaying nodes, our scheme makes a report successfully reach the base station. Thus, the average communication cost of our forwarding scheme for delivery of a single packet is smaller than those of the previous schemes.

Our neighbor list verification during the bootstrapping phase requires the three-way handshaking neighbor discovery. Unlike the neighbor discovery between two pure nodes, the size of the messages exchanged between a pure and an adult node varies with the degree of the adult node. A large number of certificates caused by the high degree can be overburdensome to a single TinyOS packet which provides 29 bytes for data. Considering 8-byte certificates and a 4-byte[7] message authentication code (MAC), the adult node is able to include at most two neighbors' information in a single TinyOS packet. Thus, when the entire neighbor list cannot be accommodated within a single packet, the node should allot the list to several packets and send them serially. In a network of size N with the expected degree d of each node, the average number of packets invoked by a newly-added node per each node is nearly $(d-1)^2/2(N-1)$.

Therefore, as node density d grows, the total number of packets transmitted from adult nodes to a newly-added node increases. However, neighbor discovery between a pure and an adult node occurs much less than between two pure nodes, since most neighbor discoveries throughout the network are between two pure nodes in the early stage of the network. Neighbor discovery between a pure and an adult node occurs generally when a new node is added to the network.

4.2 Storage Overhead

In LEAP, each node keeps four types of keys and a manageable length of hash chain, which is found to be scalable. In our scheme, each node needs to additionally store its direct neighbors' certificates and their respective neighbor lists as in Table 1. Thus, for a network of the expected degree d and the byte size l of node ID, the additional storage requirement for each node is $d \cdot (8 + ld)$ bytes.

Although our storage requirement for these neighbor lists is $O(d^2)$, for a reasonable degree d, memory overhead does not exceed 1 KB (a Berkeley MICA2 Mote with 128 KB flash memory and 4 KB SRAM). For example, when $d = 20$ and $l = 2$, a node needs 960 bytes of memory to store such information.

If node density of a network is so high that the required space for those neighbor lists significantly increases and the storage utilization becomes an issue, we can employ a storage-reduction technique such as Bloom filter [2]. For example, when $d = 30$ and $l = 2$, a node requires 2,040 bytes of additional space mainly for the neighbor lists. Instead of storing neighbors' neighbor lists, applying each of the neighbor lists (480 bits) to a Bloom filter (of 5 hash functions mapping to a 256 bit vector), a node needs the reduced space of 1,200 bytes for such information (with the false positive probability = 0.02).

[7]4-byte MAC is found to be not detrimental in sensor networks as in TinySec [24] which employs 4-byte MAC.

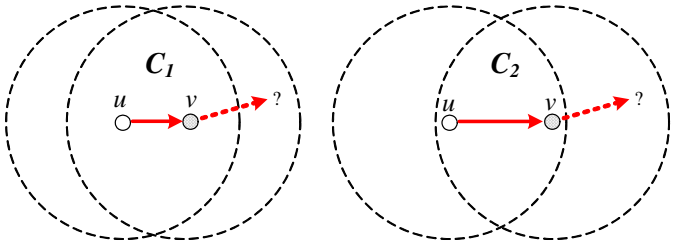


Figure 4: Examples of critical area C_1 and C_2.

4.3 Resilience to Packet-Dropping Attacks

In face of maliciously packet-dropping nodes, the higher degree of multipath we provide, the more resiliency our scheme achieves against such attacks. The average degree of multipath depends on the number of sub-watch nodes around a packet-dropping node. Sub-watch nodes should be located in the region within the communication range of both forwarding node u and designated next-hop v. We refer to such a region as *critical area*. As in Figure 4, if nodes u and v are located farther away, the size of critical area C_2 gets smaller than that of C_1, and the probability (p_c) that at least one sub-watch node exists in the critical area goes down. The probability (p_c) is

$$p_c = 1 - (1-c)^{d-1},$$

where c is the ratio of the critical area size to the node's communication range, and the expected degree d of the node.

To determine the appropriate degree d, we set the smallest critical area C_2 in Figure 4 as a lower bound case ($c = 0.4$). Figure 5 shows that, even in the lower bound critical area, with $d = 6$ and $d = 10$, probability p_c is above 0.9 and above 0.99, respectively.

Since, in a network of degree d, the probability that there exist m sub-watch nodes in the critical area of the ratio c is

$$p(m) = \binom{d-1}{m} c^m (1-c)^{d-m-1},$$

the expected number of sub-watch nodes, m, in the critical area is given by

$$E[m] = (d-1)c.$$

Thus, in the lower bound ($c = 0.4$) critical area, when $d = 10, 15, 20$, the number of sub-watch nodes (i.e. the degree of multipath) is 3.6, 5.6, 7.6 on average, respectively. This

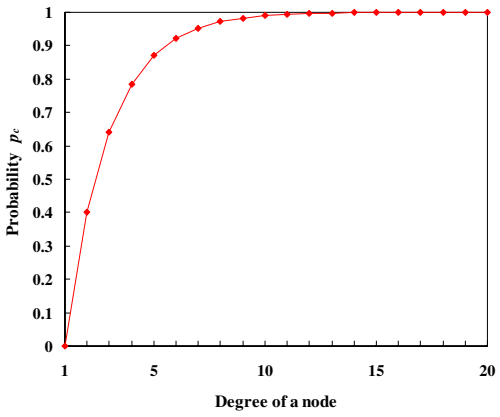


Figure 5: Probability (p_c) that at least one subwatch node exists in the lower bound ($c = 0.4$) critical area.

shows that the higher degree of each node has, our scheme has the higher degree of multipath and resiliency against packet-dropping nodes.

4.4 The Security of Neighbor List Verification

Our Neighbor List Verification(NLV) keeps the nice properties of LEAP. Adult nodes fail to establish pairwise keys with any adult nodes in arbitrary locations, so that the impact of a node compromise is localized. NLV performs the three-way handshaking neighbor discovery, instead of two-message exchange in LEAP. The three-way handshaking enables each node to verify not only its direct neighbors but also their respective neighbor lists.

Moreover, this this three-way handshaking can be a potential solution to deal with irregularity of radio range [15, 37, 45]. In reality, due to the noise and some environmental factors, radio range of each node is not exactly circular. So, communication links among nodes are asymmetric; node u can hear node v which is unable to hear u. With two-message exchange, only the node initiating the neighbor discovery is assured of the link's bidirectionality. By the three-way handshaking, both of neighbors can be assured of their symmetric connectivity.

With NLV, only the verified lists are stored and utilized for our packet-forwarding scheme. NLV verifies the neighbor list of an adult node with certificates. These certificates merely attest neighborship between two specific nodes. Even if a node is compromised, the attacker fails to abuse the certificates of the captured node for other purpose.

However, collusion among compromised nodes can fabricate bogus certificates in order to deceive a newly-added node. For example, consider two colluding nodes u and v at the different locations. When compromised node u discovers newly-added node x, node u provides x with u's neighbor list (maliciously including v in it). Even though node v is not an actual neighbor of u, colluding node v can generate the bogus certificate for u, $MAC(K_v, v|u)$. Then, x falsely believes that v is a direct neighbor of u. This attack, however, affects only the one newly-added node x. Thus, when compromised node u tries to launch the blind letter attack[8], other surrounding adult neighbors of u can still detect it anyway.

The more serious case is that colluding nodes exploit a newly-added node to generate bogus one-time certificates. For example, consider two colluding nodes u and v that share all their secret information as well as all their certificates. When newly-added node x initiates the three-way handshaking with u, compromised node u pretends to be v and provides x with v' neighbor list. Then, x in return provides u with one-time certificates for each neighbor of v; these one-time certificates falsely attest that v has new neighbor x. Node u sends this information to v over the covert channel. Then, v broadcasts these one-time certificates, and neighbors of v falsely believe that x is a direct neighbor of v.

Unfortunately, we do not provide a proper countermeasure to defend against this type of *man-in-the-middle* attacks. However, we point out that this type of attacks has to be launched in the passive manner. The adversary has to get the chance of discovery of a newly-added node. In other words, compromised nodes wait for the initiation of the three-way handshaking from a newly-added node. Since the attacker does not know where the new nodes will be added, it has to compromise a sufficient number of legitimate nodes in order to increase the probability of discovery of newly-added nodes.

As an active defense against such man-in-the-middle attacks, we can apply a *node replication* detection mechanism such as Randomized or Line-Selected Multicast [31], which revokes the same ID node at the different location claims. To successfully launch such man-in-the-middle attacks, two colluding nodes should pretend to be each other so that each of them claims to be at two different locations with the same ID. Location-binding key-assignment scheme by Yang *et al.* [39] with a little modification also can be a good solution to such attacks. Since it binds secret keys with nodes' geographic locations, the key bound to the particular location cannot be used at any arbitrary locations. Adopting this, NLV can check whether the claimed neighbors are really located within geographically two hops away.

4.5 Simulations

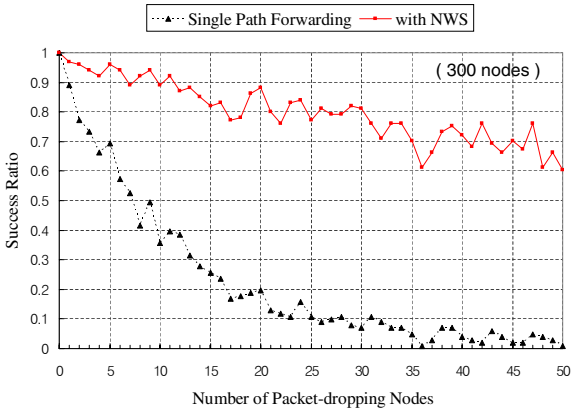
To further evaluate the performance of our resilient forwarding scheme, we run simulations of our scheme in the presence of packet-dropping nodes on a network simulator, *ns-2* [9].

4.5.1 Simulation Model

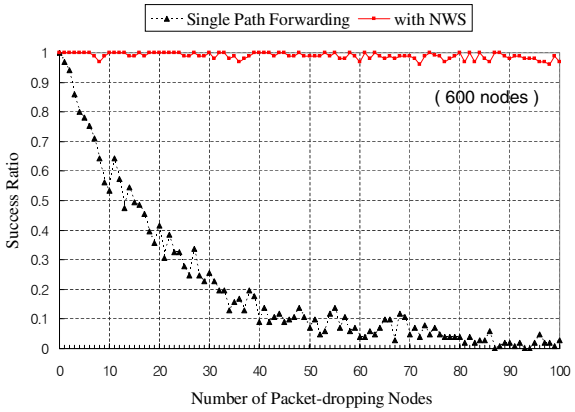
In our simulations, we deploy N sensor nodes uniformly at random within $500 \times 500m^2$ target field, with $N = 300$ and 600. Each sensor node has a constant transmission range of $30m$, so that the degree of each node is approximately 10 ($N = 300$) and 20 ($N = 600$) on average. We position a base station and a source node in opposite corners of the field, at a fixed point (50, 50) and (450, 450), respectively. They are located approximately 18 hops away from each other.

We distribute compromised nodes over an inner square area with $200m$ each side (from $150m$ to $350m$ of each side of the $500 \times 500m^2$ target area). Thus, compromised nodes are strategically-placed in between the base station and the source node. In the simulations, those compromised nodes drop all the relaying packets.

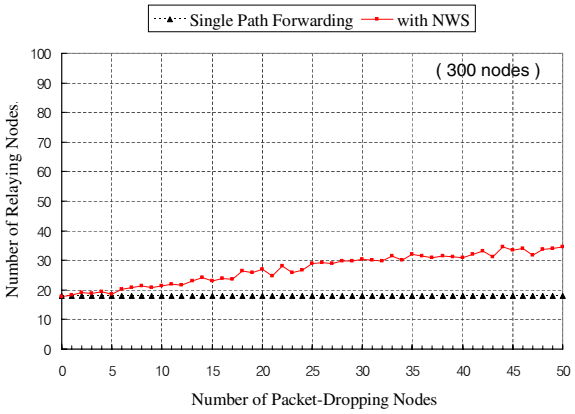
[8]Compromised node u transmits the relaying packet with its next-hop id as v, so that x considers it forwarded correctly.



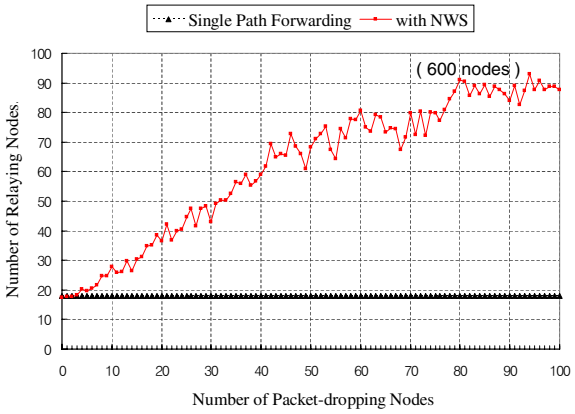
(a) Success ratio ($N = 300$, $x = 0 \sim 50$)



(b) Success ratio ($N = 600$, $x = 0 \sim 100$)



(c) The number of relaying nodes with $N = 300$



(d) The number of relaying nodes with $N = 600$

Figure 6: Simulation Results (averaged over 100 runs).

We use the typical TinyOS beaconing [17] with a little modification as a base routing protocol in our simulations. We add a hop count value in a beacon message[9]. To have multiple potential next-hops, when receiving a beacon with the same or better hop count than the parent node's, each node marks the node sending the beacon as a potential next-hop.

Each simulation experiment is conducted using 100 different network topologies, and each result is averaged over 100 runs of different network topologies.

4.5.2 *Simulation Results*

In the presence of compromised node dropping all the relaying packets, we measure the success ratio (i.e. the percentage of the packets that successfully reach the base station from the source) and the number of relaying nodes by the primitive single-path forwarding and with NWS in a network of size N, with $N = 300$ and 600.

[9]The base station initiates the beacon-broadcasting, which floods through the network, in order to set up a routing tree.

Figure 6(a) shows the success ratio in face of x packet-dropping nodes (varying x=0 to 50) in a 300-sensor-node network with the approximate degree $d = 10$. Although the success ratio gently decreases with x, it keeps up above 0.8 even with $x = 30$, with the help of NWS. This tendency of decreasing success ratio can be attributed to the degree $d = 10$ (3.6 sub-watch nodes on average) as well as an increasing number of packet-dropping nodes. Due to the strategically-placement of compromised nodes in our simulations, as x increases on, it is likely that a forwarding node's all potential sub-watch nodes themselves are packet-dropping nodes. Figure 6(c) shows the number of nodes that relay the packet from the source to the base station in the same experiments. Since the source is located about 18 hops away from the base station, the number of relaying nodes only with the single-path forwarding remains at 18. With NWS, the number of relaying nodes increases with x, in order to bypass an increasing number of packet-dropping nodes. In face of such nodes, our scheme converts single-path forwarding into multipath data forwarding, with the

help of sub-watch nodes around such packet-dropping nodes. Utilizing a cache for recently-received packets can suppress the same copy within a certain timeout, which reduces the number of relaying nodes.

Figure 6(b) shows the success ratio in a 600-sensor-node network with the approximate degree $d = 20$ with x packet-dropping nodes (varying x=0 to 100). Unlike that with $N = 300$, the success ratio stays constantly at around 0.99 even with $x = 100$, with the help of NWS. This tendency of high success ratio can be mainly attributed to the degree $d = 20$ (7.6 sub-watch nodes on average in the lower bound case), which is found to be high enough to bypass a large number of packet-dropping nodes. Figure 6(d) shows the number of relaying nodes from the source to the base station in the same experiments. With NWS, the increase in the number of relaying nodes with x is more conspicuous than that with $N = 300$, since more than twice as many as sub-watch nodes help forward the packets so that it can bypass a large number of packet-dropping nodes anyway.

In the simulation results, we note that our forwarding scheme dynamically adjusts its forwarding style, depending on the number of packet-dropping nodes en-route to the base station. As in Figures 6(c) and 6(d), while there exist none or a small number of packet-dropping nodes on the way, our scheme works almost like the single-path forwarding with the help of a few additional relaying nodes. On the other hand, when confronting a large number of packet-dropping nodes, our scheme makes full use of the help from additional relaying nodes, in order to successfully deliver the packet to the base station at any cost to the best efforts.

5. CONCLUSIONS AND FUTURE WORK

In this paper we focus on defending against compromised nodes' dropping of legitimate reports. We have presented a resilient packet-forwarding scheme using *Neighbor Watch System* (NWS) against maliciously packet-dropping nodes in sensor networks. In face of such nodes, NWS is specifically designed for hop-by-hop reliable delivery, and the prompt reaction of the conversion from single-path to multipath forwarding augments the robustness in our scheme so that the packet successfully reach the base station.

In future work, we plan on further improving NLV to defend against the *man-in-the-middle* attacks, collusion among compromised nodes. Such attacks can be prevented by using a master key derived with not only a node ID but also its geographic information. We will also seek to address $O(d^2)$ storage requirement for the neighbors' neighbor lists. Finally, we would like to perform an intensive experimental evaluation to compare our scheme with other reliable delivery protocols [10, 29, 42].

6. ACKNOWLEDGMENTS

This work was supported by grant No.R01-2006-000-10073-0 from the Basic Research Program of the Korea Science and Engineering Foundation.

7. REFERENCES

[1] R. Anderson, H. Chan, and A. Perrig, *Key Infection: Smart Trust for Smart Dust*, IEEE ICNP 2004

[2] Burton H. Bloom, *Space/Time Trade-offs in Hash Coding with Allowable Errors*, Communication of the ACM, vol. 13, 422-426, 1970

[3] B. Carbunar, I. Ioannidis, and C. Nita-Rotaru, *JANUS: Towards Robust and Malicious Resilient Routing in Hybrid Wireless Networks*, ACM workshop on Wireless security (WiSe'04), Oct. 2004

[4] H. Chan, A. Perrig, and D. Song, *Random Key Predistribution Schemes for Sensor Networks*, IEEE Symposium on Security and Privacy, pp. 197-213, May 2003.

[5] B. Deb, S. Bhatnagar, and B. Nath, *ReInForM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks*, IEEE Local Computer Networks (LCN 2003), pp. 406-415, Oct. 2003.

[6] J. Deng, R. Han, and S. Mishra, *A Performance Evaluation of Intrusion- Tolerant Routing in Wireless Sensor Networks*, 2nd International Workshop on Information Processing in Sensor Networks (IPSN 03), pp. 349-364, Apr. 2003.

[7] J. Deng, R. Han, and S. Mishra, *Intrusion Tolerance and Anti-Traffic Analysis Strategies for Wireless Sensor Networks*, IEEE International Conference on Dependable Systems and Networks (DSN), pp. 594-603, 2004.

[8] J. Deng, R. Han, and S. Mishra, *Defending against Path-based DoS Attacks in Wireless Sensor Networks*, ACM Workshop on Security of Ad-Hoc and Sensor Networks (SASN'05) , Nov, 2005.

[9] K. Fall and K. Varadhan (editors), *NS notes and documentation*, The VINT project, LBL, Feb 2000, http://www.isi.edu/nsnam/ns/

[10] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, *Highly Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks*, Computing and Communications Review (MC2R) Vol 1., pp. 11-25, 2002.

[11] V. D. Gligor, *Security of Emergent Properties in Ad-Hoc Networks*, International Workshop on Security Protocols, Apr. 2004.

[12] O. Goldreich, S. Goldwasser, and S. Micali, *How to Construct Random Functions*, Journal of the ACM, Vol. 33, No. 4, 210-217, 1986

[13] L. Eschenauer and V. D. Gligor, *A Key-Management Scheme for Distributed Sensor Networks*, 9th ACM Conference on Computer and Communication Security (CCS), pp. 41-47, Nov. 2002.

[14] C. Hartung, J. Balasalle, and R. Han, *Node Compromise in Sensor Networks: The Need for Secure Systems*, Technical Report CU-CS-990-05, Department of Computer Science University of Colorado at Boulder, Jan. 2005

[15] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, *An Energy-Efficient Surveillance System Using Wireless Sensor Networks*, ACM MobiSys'04, June, 2004

[16] W.R. Heinzelman, J. Kulik, H. Balakrishnan, *Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*, ACM MobiCom99, pp. 174.185, 1999.

[17] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, *System Architecture Directions for Networked Sensors*, ACU ASPLOS IX, November 2000.

[18] X. Hong, M. Gerla, W. Hanbiao, and L. Clare, *Load Balanced, Energy-Aware Communications for Mars Sensor Networks*, IEEE Aerospace Conference, vol.3, 1109-1115, 2002.

[19] Y.-C. Hu, D. B. Johnson, and A. Perrig, *SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks*, IEEE Workshop on Mobile Computing Systems and Applications, pp. 3-13, Jun. 2002.

[20] Y.-C. Hu, A. Perrig, and D. B. Johnson, *Efficient Security Mechanisms for Routing Protocols*, NDSS 2003, pp. 57-73, Feb. 2003.

[21] C. Intanagonwiwat, R. Govindan and D. Estrin, *Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*, MobiCom'00, Aug. 2000.

[22] D. Johnson, D.A. Maltz, and J. Broch, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks* (Internet-Draft), Mobile Ad-hoc Network (MANET) Working Group, IETF, Oct. 1999.

[23] C. Karlof and D. Wagner, *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures*, The First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 113-127, May 2003

[24] C. Karlof, N. Sastry, and D. Wagner, *TinySec: A Link Layer Security Architecture for Wireless Sensor Networks*, ACM SensSys'04, pp. 162-175, Nov. 2004.

[25] I. Khalil, S. Bagchi, and C. Nina-Rotaru, *DICAS: Detection, Diagnosis and Isolation of Control Attacks in Sensor Networks*, IEEE SecureComm 2005, pp. 89 - 100, Sep. 2005

[26] Y. Liu and W. K.G. Seah, *A Priority-Based Multi-Path Routing Protocol for Sensor Networks*, 15th IEEE International Symposium on Volume 1, 216 - 220, 2004

[27] S.-B. Lee and Y.-H. Choi, *A Secure Alternate Path Routing in Sensor Networks*, Computer Communications (2006), doi:10.1016/j.comcom.2006.08.006.

[28] S. Marti, T.J. Giuli, K. Lai, and M. Baker, *Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*, ACM/IEEE International Conference on Mobile Computing and Networking, pp. 255-265, 2000

[29] H. Morcos, I. Matta, and A. Bestavros, *M^2RC: Multiplicative-Increase/Additive-Decrease Multipath Routing Control for Wireless Sensor Networks*, ACM SIGBED Review, Vol. 2, Jan 2005.

[30] J. Newsome, E. Shi, D. Song, and A. Perrig, *The Sybil Attack in Sensor Networks: Analysis and Defenses*, IEEE IPSN'04, pp. 259-268, Apr. 2004.

[31] B. Parno, A. Perrig, and V. D. Gligor, *Distributed Detection of Node Replication Attacks in Sensor Networks*, the 2005 IEEE Symposium on Security and Privacy, pp. 49-63, May 2005.

[32] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, *SPINS: Security Protocols for Sensor Networks*, ACM MobiCom'01, pp. 189-199, 2001.

[33] A. Perrig, J. Stankovic, and D. Wagner, *Security in Wireless Sensor Networks*, Communications of the ACM, 47(6), Special Issue on Wireless sensor networks, pp.53- 57, Jun. 2004

[34] B. Przydatek, D. Song, and A. Perrig, *SIA: Secure Information Aggregation in Sensor Networks*, 1st International Conference on Embedded Networked Sensor Systems, 255-256, 2003

[35] E. Shi and A. Perrig, *Designing Secure Sensor Networks*, Wireless Communications, IEEE Volume 11, Issue 6, pp. 38-43, Dec. 2004.

[36] D. Tian and N.D. Georganas, *Energy Efficient Routing with Guaranteed Delivery in Wireless Sensor Networks*, IEEE Wireless Communications and Networking (WCNC 2003), IEEE Volume 3, 1923 - 1929, March 2003

[37] A. Woo, T. Tong, and D. Culler, *Taming the Underlying Challenges of Reliable Multhop Routing in Sensor Networks*, ACM SenSys03, Nov, 2003

[38] A. Wood and J. Stankovic, *Denial of Service in Sensor Networks*, IEEE Computer, Vol.35, 54-62, Oct. 2002

[39] H.Yang, F. Ye, Y. Yuan, S. Lu and W. Arbough, *Toward Resilient Security in Wireless Sensor Networks*, ACM MobiHoc'05, 34-45, May 2005

[40] Y. Yang, X. Wang, S. Zhu, and G. Cao *SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks*, ACM MobiHoc'06 May 2006

[41] F. Ye, H. Luo, S. Lu and L. Zhang, *Statictial En-route Filtering of Injected False Data in Sensor Networks*, IEEE INFOCOM, 2004

[42] F. Ye, G. Zhong, S. Lu and L. Zhang, *GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks*, ACM Wireless Networks (WINET), March 2005

[43] Y. Yu, R. Govindan, and D. Estrin, *Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks*, UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.

[44] W. Zhang and G. Cao, *Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach*, IEEE INFOCOM'05. Vol. 1, 503-514, March 2005

[45] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, *Impact of radio irregularity on wireless sensor networks*, the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys04), June, 2004

[46] S. Zhu, S. Setia, and S. Jajodia, *LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks*, The 10th ACM Conference on Computer and Communications Security (CCS '03), 62-72, 2003

[47] S.Zhu, S. Setia, S. Jajodia, and P. Ning, *An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks*, IEEE Symposium on Security and Privacy, 2004