Security Support for In-Network Processing in Wireless Sensor Networks

Jing Deng
Department of Computer Sciences
University of Colorado
Boulder, CO, USA
Jing.Deng@colorado.edu

Richard Han
Department of Computer Sciences
University of Colorado
Boulder, CO, USA
rhan@cs.colorado.edu

Shivakant Mishra
Department of Computer Sciences
University of Colorado
Boulder, CO, USA
mishras@cs.colorado.edu

ABSTRACT

The benefits of in-network processing for wireless sensor networks include improved scalability, prolonged lifetime, and increased versatility. This paper addresses the challenges associated with securing in-network processing within WSNs, and proposes a collection of mechanisms for delegating trust to aggregators that are not initially trusted by individual sensor nodes. Security mechanisms are proposed to address the downstream requirement that sensor nodes authenticate commands disseminated from parent aggregators. Conversely, security mechanisms are also proposed to address the upstream requirement that aggregators authenticate data produced by sensors before aggregating. Simulation results in ns2 of the proposed mechanisms for secure in-network processing are presented, as well as implementation on a mote testbed.

Categories and Subject Descriptors

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

General Terms

Design, Security

Keywords

Security mechanisms, Sensor networks, In-network processing

1. INTRODUCTION

Wireless sensor networks (WSN) have the potential to revolutionize the process of information gathering and processing over the next decade. In these networks, a distributed collection of sensor nodes forms a network interconnected by wireless communication links. Each sensor node acts as an information source, sensing and collecting data samples from its environment. Sensor nodes perform routing functions, creating a multi-hop wireless networking fabric that relays data samples to a back-end server via a small number of base stations. To date, applications of sensor networks include habitat monitoring [25][39][40], robotic toys [24], target tracking [19], and battlefield monitoring [2]. In the near future, we expect an even wider range of applications to emerge, including location aware sensor networks in the home and office, assistive technology applications with biomedical sensing, and outdoor deployments of GPS-enabled sensor networks [1] to monitor storms, fires, and marine ecologies.

The primary focus of this paper is on providing security mechanisms to enable the secure operation of *in-network processing*, a key emerging theme in the design and deployment of WSNs. In one form of in-network processing, an intermediate node or aggregator fuses or aggregates sensor data collected from a group of sensor nodes before forwarding the aggregated data on to a base station. For example, in a sensor group comprised of temperature sensors, the aggregator can collect temperature readings from different sensor nodes in a given time interval, locally process these readings, and then forward an average of these readings [41].

The benefits of in-network processing for WSNs include improved scalability, prolonged lifetime, and increased versatility. First, since aggregation reduces the volume of data communicated throughout the sensor network, then the benefits of aggregation include prolonging the lifetime of the WSN, which is one of the most critical factors in the design and deployment of WSNs. Second, since the aggregators themselves may be aggregated to form a multi-level hierarchy, then another benefit of in-network processing is enabling scalability via hierarchy, a technique that has been widely employed in wired networks like the Internet. Third, in-network processing improves the versatility of WSNs so that important applications such as target tracking can be efficiently implemented, e.g. by localized target tracking.

A second form of in-network processing also employs hierarchy, except in the reverse direction, in order to disseminate control/command messages from the base station outwards through the aggregators and downstream towards the sensor leaf nodes. Again, hierarchy provides a scalable and efficient means of communication.

As shown in Figure 1, a given target tracking application for a WSN may need to employ both data aggregation and

Proceedings of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks Fairfax, Virginia

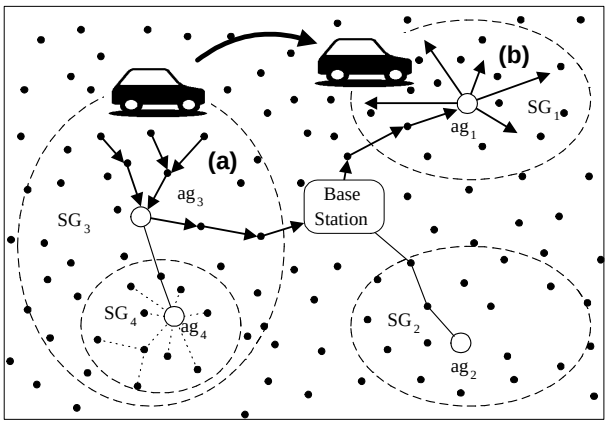

Figure 1: Securing a wireless sensor network with in-network processing in the form of (a) aggregation and (b) dissemination.

data dissemination. For example, Figure 1(a) shows how sensor data is first aggregated. In Figure 1(b), the base station generates and disseminates commands to nearby subsets of sensors to enable target tracking.

Examples of in-network processing for WSNs abound in the literature. In directed diffusion data routing for WSNs [13], in-network processing reinforces routes at intermediate nodes such that sensor data is routed correctly to the sinks that express interest in those events. Placement of in-network filters within a sensor network has also been explored [3]. Both the TinyDB project [23] and the Cougar project [6] offer powerful database tools to support efficient in-network query processing. In addition, in-network acoustic signal processing has been distributed across a heterogeneous hierarchy of sensor nodes with varying capabilities [16].

The focus of this paper is on the challenge of securing in-network processing for WSNs, i.e. of providing mechanisms for securing both upstream data aggregation as well as downstream data dissemination. Secure data aggregation requires at the minimum that intermediate aggregators be able to authenticate the reported data of their downstream nodes. Secure data dissemination requires at the minimum that each sensor node be able to authenticate commands sent from parent nodes. For both upstream and downstream directions, protection is also needed to provide security against eavesdropping, tampering, denial-of-service (DOS) attacks, and the physical compromise of sensor nodes.

WSNs pose additional design challenges in terms of their resource constraints and the broadcast nature of the wireless medium. Limited memory, CPU, communication and battery life limit the applicability of compute-intensive techniques like public-key cryptography. Wireless communication increases the vulnerability of the network to eavesdropping, unauthorized access, spoofing, and replay and denial-of-service (DOS) attacks. For example, the problems with wireless security standards such as Wired Equivalent Privacy (WEP) for wireless Ethernet 802.11b have been well-documented[4]. These problems are exacerbated by the resource constraints imposed on sensor nodes, limiting the strength of defensive security countermeasures such as encryption and authentication.

Related work in secure sensor networks at first largely ignored the role of in-network processing in hierarchical WSNs, though more recently has begun to address this issue. SPINS [30] has proposed general security primitives for resource-constrained sensor networks. These primitives are not specifically tailored to in-network processing in hierarchical WSNs. INSENS[7] proposes an intrusion-tolerant routing mechanism for WSNs rooted in a base station, but does not address either in-network processing or WSNs with an arbitrary multi-level hierarchy. Recently, B. Przydatek, D. Song and A. Perrig addressed secure data aggregation in sensor networks from the point of view of detecting forged aggregation data values [31]. However, this paper does not address the issues of how to build a secure network infrastructure to support hierarchical WSNs, e.g. how to set up trust between aggregators and sensor nodes and how to securely disseminate commands from aggregators.

In this paper, we focus on a simple yet flexible hierarchical model of in-network processing and focus on securing this model. As shown in Figure 1, in this model, aggregators form a multi-hop sensor groups encompassing any number of sensor nodes. Multiple levels of aggregation hierarchy may be formed. Within each sensor group, an aggregator can both aggregate data as well as disseminate commands. This hierarchical network can be dynamically constructed: the aggregator can create or dissolve sub-groups as needed. This network model is flexible enough to support most forms of in-network processing, e.g. decentralized in-network query processing of [3] and TinyDB [23]. This paper makes the further common assumption that resource-constrained sensor nodes must employ symmetric key techniques. As a means of securely bootstrapping the entire network, each node is preconfigured with a custom symmetric key shared only with the base station.

Based on this model, the challenge is to develop a secure and efficient mechanism for delegating trust to aggregators within the WSN. Recall that the sensor nodes do not implicitly trust a fellow node that has been assigned as an aggregator in our model. Instead, the sensor nodes and aggregators at first only trust the base station. Thus, our goal is to leverage the base station as a trusted third-party to find a set of mechanisms that enable sensor nodes to trust aggregators, and vice versa. Three issues must be considered. First, this paper outlines a method for determining secure membership, so that both sensor nodes and aggregators can come to a secure understanding as to which nodes belong to which sensor group. Second, this paper offers a mechanism whereby an aggregator can securely and scalably disseminate commands that are trusted, i.e. authenticated, by individual sensor nodes. Third, this paper provides a mechanism for aggregators to trust the data nodes, again by authentication.

The contributions of this paper are as follows. To commit authority to an aggregator, we present a mechanism for distributing one-way sequence numbers, derived from one-way hash chains, to memory-constrained aggregator nodes. Combined with more traditional MAC authentication, one-way sequence numbers provide an efficient means to quickly authenticate messages from an aggregator with little setup overhead. We also present a lightweight mechanism for establishing pairwise secret keys between sensor nodes and their corresponding aggregators. Finally, we introduce the

notion of "ripple" keys as an efficient mechanism to achieve secure local broadcast within a small network of sensor nodes in an aggregator's domain.

The rest of the paper is organized as follows. Section 2 describes our approach to trust delegation using one-way hash chains. Section 3 describes a lightweight key setup mechanism. Section 4 introduces the ripple key concept. Section 5 details our overall security framework and protocol for distributing trust in a hierarchical WSN. It also briefly describes how to dynamically build a subgroup, and how to add a node to a subgroup. Section 6 reports on a prototype simulation and implementation, and provides a detailed performance evaluation. Section 7 discusses related work, and section 8 concludes the paper.

2. DELEGATION OF AUTHORIZATION

An aggregator of a sensor group needs to *disseminate* appropriate commands *to* its group members to control their operation. In a hostile computing environment, an important requirement is that the sensor nodes must be able to distinguish between a command originating from the aggregator of their sensor group and a forged command originating from some malicious node. Since an aggregator is chosen dynamically after a WSN has been deployed, it is not possible to preconfigure sensor nodes and the aggregators with appropriate cryptographic information to accomplish this. We propose a technique called *delegation of authorization* that allows an aggregator to be loosely authenticated after sending a command to the members of its sensor group. Using this technique, the base station can delegate authority to an aggregator for a limited period of time. This technique exploits two security mechanisms that have been used for building secure WSNs: one-way hash chains [17, 18] and the μTESLA protocol [30].

A one-way hash chain (OHC) is generated by a one-way function F. F satisfies the following two property: 1) Given x, it is easy to compute y, such that $y = F(x)$; and 2) Given y, it is computationally infeasible to compute x such that $y = F(x)$. An OHC is a sequence of numbers, K_n, K_{n-1}, ..., K_0, such that

$$\forall j : 0 < j \leq n : K_{j-1} = F(K_j)$$

To delegate trust to the aggregators, the base station computes separate one-way hash chains, OHC_i: O^i_m, O^i_{m-1}, ..., O^i_0 for each sensor group SG_i. It sends O^i_m to the aggregator of SG_i, and O^i_0 to all members of SG_i. An aggregator ag_i can use OHC_i to loosely authenticate itself in the commands it sends to its group members. In the k^{th} packet sent by ag_i, O^i_k is included. A receiver node authenticates the source of the packet by verifying $F^k(O^i_k) = O^i_0$, where F^k represents applying function F k times. This verification guarantees that ag_i must have generated the number O^i_k in the packet. Because F^{-1} is computationally infeasible, it is impossible for a sensor node other than ag_i to determine what the next number in OHC_i is. In section 4, we will describe the method of authenticating the content of a packet sent by ag_i.

A simple way to disseminate O^i_0 to all members of SG_i is by sending a separate unicast message to each sensor node in SG_i. This approach is of course not scalable. Instead, we propose to use μTESLA for doing this. μTESLA is an efficient and light-weight secure broadcast protocol for wireless sensor network [30]. This protocol requires that the base station and nodes are loosely time synchronized, and each node knows an upper bound on the maximum synchronization error. In addition, the sender of the broadcast packets maintains a one-way hash chain (K_0, K_1, ... K_n) called a *key chain*, and each sensor node is configured with K_0, as in figure 2. A packet broadcast from the base station in time slot t contains a message authentication code (MAC) generated using key K_t. When a sensor node receives this packet, it doesn't know K_t. The base station broadcasts the key K_t after d time slots, where d is of the order of a few time intervals. d should be greater than an reasonable round trip time between the sender and the receivers. On receiving a key K_x in time slot $t + d$, a node verifies that $F^t(K_x) = K_0$. If the key is verified, the node verifies the integrity of the message received earlier by computing the MAC using the key.

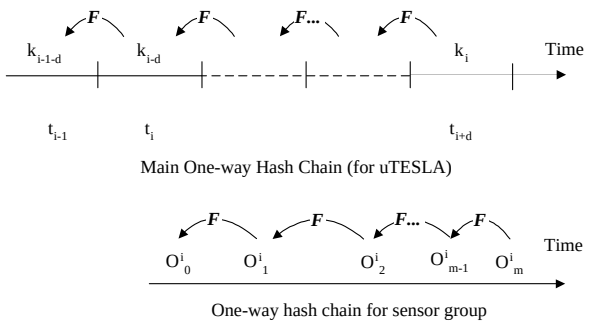

Figure 2: Main One-way hash chain and sensor group one-way hash chain.

To delegate trust to the aggregators, the base station uses μTESLA to broadcast a list containing $< SG_i, ag_i, O^i_0, r_i >$ for each sensor group SG_i to all nodes in the network. Here r_i is a random number that is used for establishing the pairwise shared key (see Section 3). The base station also unicasts O^i_m to the aggregator ag_i. By using μTESLA, each sensor can verify whether the message it received originated from the base station, and whether the contents of the message have been tampered with.

The one-way hash chain provides a very convenient way to not only delegate trust to an aggregator, but also limit the length of this trust. The size of the one-way hash chain that the base station generates for an aggregator ag_i automatically determines the number of packets that ag_i can send to its sensor group members. So, if the length of this chain is m, ag_i can send at most $m - 1$ separate commands to the sensor nodes. After ag_i has used up all numbers in OHC_i, the base station can assign a new OHC_i to ag_i using the same method. In addition to limiting the length of trust delegated to an aggregator, it is possible to restrict the type of commands that an aggregator may send using the delegated OHC. For example, the sensor nodes can be preconfigured with a set of commands, and they accept only one of the commands from this set from the aggregator.

In our protocol, if the aggregator is compromised, the damage is confined to the aggregator's own sub-group of sensors. This is because the compromised aggregator cannot predict the sequence of numbers in the one-way hash chain of any other aggregators. However, the malicious aggregator can still generate forged aggregation data. This

problem was addressed in [31].

Because of memory limitations, a sensor node cannot store a complete OHC, even though the chain is much shorter than the main OHC maintained by the base station. To solve this problem, we can make tradeoffs between computation and storage. An aggregator need only save the seed and some intermediate numbers in its OHC. The aggregator can generate the sequence numbers between intermediate numbers by applying the one-way function repeatedly. One strategy for determining the intermediate numbers is to select more numbers that will be used in the near future, and sparsely select fewer numbers that will be used much later. We can also select intermediate numbers based on a skiplist, as suggested in [11].

3. LIGHTWEIGHT SHARED SECRET KEY ESTABLISHMENT

To provide support for ensuring privacy and integrity of data packets sent *from* sensor nodes *to* their aggregators, a separate pairwise secret key shared between each sensor node and its aggregator is required. We call this key a *subkey* of a sensor node. This shared key can also defend against the Sybil attack [8].

In our design, the base station generates a subkey for each sensor node. We assume that each sensor node is preconfigured with a custom secret key $\mathcal{K}_s$ that it shares with the base station only. Given $\mathcal{K}_s$ and a random number r, a subkey $K_{s,r}$ for s is generated as follows:

$$K_{s,r} = G(\mathcal{K}_s, r)$$

Function G has the following properties:

1. An adversary that knows function G and random number r cannot compute $K_{s,r}$ without knowing $\mathcal{K}_s$;
2. An adversary that knows subkeys, K_{s,r_0}, K_{s,r_1}, ..., K_{s,r_i} cannot compute K_{s,r_j}, where r_j is different from r_0, r_1, ..., or r_i;
3. An adversary that knows subkeys, K_{s,r_0}, K_{s,r_1}, ..., K_{s,r_i} cannot compute $\mathcal{K}_s$.

Function G can be implemented using a random number generator, as suggested in [30]. A subkey $K_{s,r}$ generated using the method outlined above is safe, because it does not compromise the secrecy of $\mathcal{K}_s$, even if an adversary is able to hack $K_{s,r}$.

After generating the subkeys of sensor nodes, the base station needs to distribute them to the respective nodes and aggregators in a secure manner. Again, a simple way to do this is to send a separate unicast message to each node. However, as observed before, this approach is not scalable. It is not possible to use a broadcast mechanism such as μTESLA, because the information (subkey) to be communicated is different for each node. We propose a three-step, light-weight mechanism shown in Figure 3 to distribute these subkeys.

First, the base station chooses a unique random number r_i for each sensor group SG_i, and generates subkeys for each sensor node in SG_i using the method described above. It broadcasts r_i using the μTESLA protocol as described in Section 2. After receiving this broadcast message, each sensor node knows the random number r_i used to generate subkeys for nodes in sensor group SG_i. However, they do not know which sensor group they belong to.

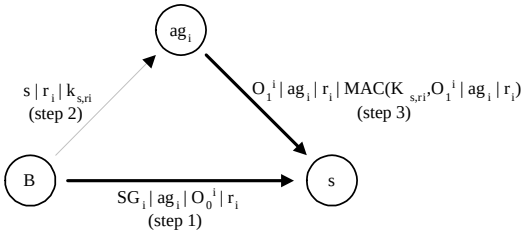

Figure 3: Secret Key Establishment

Second, the base station send a separate unicast message to each aggregator. Full details of this unicast message are described in Section 5. The unicast message sent to aggregator ag_i includes the subkeys of all sensor nodes of sensor group SG_i. Contents of this message are encrypted using the pairwise shared secret key between ag_i and the base station to preserve the confidentiality of the message, and a MAC of the message contents generated using the pairwise shared secret key is appended to ensure the integrity of the message.

Third, ag_i sends the following unicast message to each sensor node s in its sensor group.

$$ag_i \rightarrow s : O_1^i | ag_i | r_i | MAC(K_{s,r_i}, O_1^i | ag_i | r_i)$$

On receiving this message, s learns that it belongs to sensor group SG_i. It first computes its subkey K_{s,r_i} by computing $G(\mathcal{K}_s, r_i)$. It then verifies the integrity of the unicast message by computing a MAC on $O_1^i | ag_i | r_i$ using K_{s,r_i}. If verified, it completes authentication that this message indeed originated from ag_i by checking if $F(O_1^i) = O_0^i$.

4. EFFICIENT SECURE BROADCAST IN A SMALL GROUP: RIPPLE KEY

While the delegation of authorization discussed in Section 2 allows an aggregator to loosely authenticate itself in the commands it sends, a related issue is how is a command propagated to all sensor nodes within a sensor group. One simple way to do this is to send a separate unicast message to each member of the group. However, since the same command is being sent to all members, a secure broadcast mechanism such as μTESLA can be used. While μTESLA protocol provides support for secure broadcast, there are two disadvantages of using it to disseminate commands with in a sensor group. First, there is a need for loose time synchronization among the sensor nodes. Second, the delayed release of the μTESLA key can extend the overall time for command dissemination. One way to address this is to set very short time slots. However, time synchronization becomes difficult for shorter time slots. Furthermore, since one sequence number from the key chain is used up by each time slot, shorter time slots will result in faster exhaustion of the key chain.

We assume that all nodes in a sensor group are located close to one another. For example, the distance between farthest node to the aggregator is less than 5 hops. We think this is a reasonable assumption since it is the best way for in-network processing to save energy and data transmission. Based on this, we propose a method for command dissemination with in a sensor group that does not rely on unicast messages, and does not require time synchronization. We

call this method the *ripple command dissemination* method. We assume that the nodes in a sensor group are geographically close together. As shown in Figure 4, a sensor group is organized into multiple layers called *ripples*. A ripple is defined as the set of all nodes that are at the same distance (number of hops) from ag. Ripple 1 comprises of all nodes that are exactly one hop away from ag, ripple 2 comprises of all nodes that are exactly two hops away from ag, and so on. For each ripple, a secret key called a *ripple key* is generated by ag. A ripple key KR_j for ripple j is shared between ag and the members of ripple j. To disseminate a command, ag sends a separate *ripple message* for each ripple in its sensor group. A ripple message for ripple k is a broadcast message forwarded exactly once by intermediate nodes until the message reaches the nodes in the ripple k. The format of a ripple message for ripple j is as follows:

$$Mc_j : Content|O_i|MAC(KR_j, Content|O_i)$$

Here O_i is an OHC number of aggregator ag for loose authentication. ag uses a different OHC number for each ripple message, even for the same command being disseminated to nodes in a different ripple. When a sensor node at layer j gets Mc_j, it can verify whether the packet is sent from ag using O_i, and it can verify the integrity of the packet by computing a MAC on $Content|O_i$ using KR_j. Intermediate nodes that are not in ripple j cannot tamper with the contents of a ripple message without detection, because they don't know KR_j.

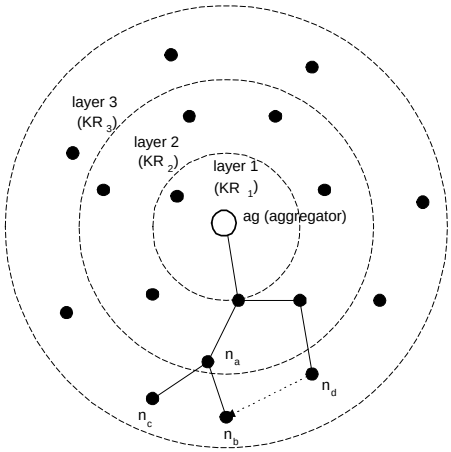

Figure 4: Ripple Command Dissemination.

A disadvantage of the ripple command dissemination method is that it requires an aggregator to send the same command multiple times (although it can send all of them in a short time), once for each ripple. Also, a separate ripple key for each ripple needs to be maintained. If the number of ripples in a sensor group is small, e.g. less than 5, we expect that the ripple command dissemination method will provide improved performance, as shown in Section 6. If a sensor group contains too many layers, the ripple command dissemination method can become inefficient. In such a case, it is useful to set up sub sensor groups.

Although the ripple command dissemination method is efficient, it is subject to a rushing attack[12]. A malicious node can send forged messages to nodes in the same layer. For example, in figure 4, if node n_b and n_d are in the same layer, when node n_d gets the message, it can change the content part of the message and generate a new MAC, and send the packet to n_b. If n_b hasn't already received the original message from aggregator, n_b will be unable to identify the message as forged. One protection against this attack would be to require that each sensor node share a secret key with all of its direct downstream nodes. The content of the command message is encrypted by this secret key, so that the format of the packet is:

$$E(Ks, Mc_j)$$

Although encryption is not used to protect the integrity of the data, since Mc_j contains a MAC, it is very difficult for an adversary to spoof the contents of $E(Ks, Mc_j)$ ([26]). If n_d and n_b are in the same layer but have different upstream nodes, n_d will have a very difficult time spoofing the message sent to n_b because n_d doesn't have the secret key between n_b and its upstream node n_a. If two nodes share the same upstream node, e.g. n_b and n_c share upstream parent n_a, they will receive the same broadcast command packet almost at the same time, so the chances of n_c attacking n_b is very small. However, if there are multiple malicious nodes in different layers and they cooperate together, they can spoof the message to the outer layer nodes.

Given this weakness, we propose to employ ripple keys in situations where nodes are geographically close and it is difficult to launch cooperative attacks with multiple malicious sensor nodes. We can use ripple command dissemination for the commands which require quick reaction from member nodes, such as query commands, and μTESLA for other critical commands that don't have a deadline. For example, if an aggregator wants to set up a subgroup, it can use μTESLA to bootstrap the one-way hash chain.

5. BUILDING A SECURE HIERARCHICAL WSN: AN INTEGRATED SOLUTION

The security mechanisms introduced in the preceding sections are integrated into a single solution for setting up a secure and hierarchical WSN in four rounds, as shown in Figure 5. In round 0, the base station discovers the topology of the complete sensor network and performs some initial preparation. In round 1, the base station securely broadcasts a list containing group id and aggregator id of all sensor groups to each node in the network using μTESLA. The intent of this round is to inform each sensor node of the existence of all sensor groups. In round 2, the base station sends a separate unicast message to each aggregator. The intent of this message is to inform an aggregator of the topology of its sensor group, and delegate trust to the aggregator for a limited period of time. Finally, in round 3, each aggregator sends a set of separate unicast messages to the members of its sensor group to set up appropriate cryptographic keys and forwarding tables to facilitate secure in-network processing.

5.1 Round 0 - Preparation

To securely discover the network topology and set up the routing tables of all sensor nodes, we rely on an appropriate secure routing protocol for WSN, such as INSENS [7]. INSENS provides support for discovering the topology of a sensor network and building routing tables in the presence of security attacks such as tampering with transmitted

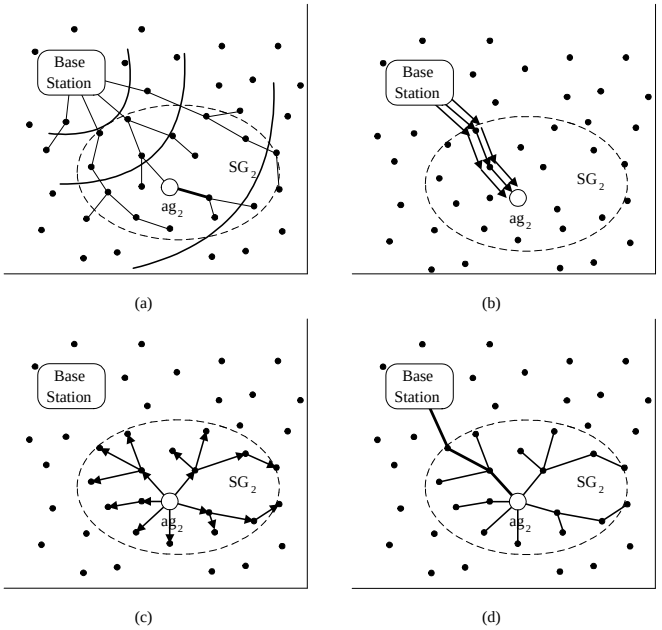

Figure 5: Building a Secure and Hierarchical WSN

data, compromising sensor nodes, and launching DOS attacks. Using such protocol, a WSN as shown in Figure 5(a) is built. In this network, each sensor node has a path to the base station. Based on the network topology, the base station configures the hierarchical network by dividing it into sensor groups, assigning a unique group id to each sensor group, and choosing an aggregator for each sensor group.

5.2 Round 1 - Group Announcement

In this round, the base station uses the delegation of authorization mechanism described in Section 2 to inform each node the identities of all sensor groups, e.g. their aggregator ids, initial sequence number of OHC of each group, and random number used to generate subkeys. For a sensor group SG_i, the broadcast message contains a list L_{SG_i} containing

$$L_{SG_i} : SG_i|ag_i|O_0^i|r_i$$

If the base station wants to set up several sensor groups at the same time, it can put lists for all sensor groups in one message. The μTESLA protocol protects the authenticity and integrity of the message. If the base station doesn't set up groups too often, the overhead of this broadcast is small. Figure 5 (a) shows this round. After receiving this message, each sensor can verify whether the message it received originated from the base station, and whether the contents of the message have been tampered with. At the end of group announcement round, a sensor node has the complete list of sensor groups in the network, but still doesn't know which sensor group it belongs to. The following is the message format for this round.

$$\begin{array}{lcl} B \rightarrow ALL & : & L_{SG_i}|MAC(K_t, L_{SG_i}) \\ B \rightarrow ALL & : & K_t \end{array}$$

5.3 Round 2 - Trust Commitment

In this round, the base station sends all information that an aggregator needs to securely build a sensor group. The information includes internal routing information, one-way hash chain(s), and the subkeys of the member nodes.

For each sensor group SG_i, the base station first generates a subkey for each member node as described in section 3. The base station then sends a unicast message to the aggregator ag_i. This message includes $< topology_i, ohc_i, r_i, key\_list_i >$. Here, $topology_i$ is the topology of the sensor group SG_i. Topology includes all connectivity information among the nodes in SG_i. For example, if nodes s_a and s_b are in SG_i and can communicate directly with each other, we put $< s_a, s_b >$ in $topology_i$. ohc_i is the one-way hash chain that the aggregator will use. ohc_i must contain O_m^i that ag_i can use to generate other numbers in the one-way hash chain. In addition, ohc_i can contain some intermediate numbers in OHC to save computing cost and memory usage of ag_i. key_list_i is the list of all subkeys generated by the base station for the nodes in SG_i. Finally, the base station uses the encryption key of ag_i to encrypt the message, the MAC key of ag_i to protect message integrity, and sends the message to ag_i. The format of the message is

$$B \rightarrow ag_i : E(\mathcal{K}_{e_{ag_i}}, Cmt_{ag_i})|MAC(\mathcal{K}_{m_{ag_i}}, Cmt_{ag_i})$$

Here $Cmt_{ag_i} = topology_i|ohc_i|r_i|key\_list_i$. $\mathcal{K}_{e_{ag_i}}$ and $\mathcal{K}_{m_{ag_i}}$ are ag_i's encryption key and MAC key, respectively.

The size of this message could be much larger than the maximum packet size of a common sensor network platform. To send this message, base station needs to segment it into many packets and sends them to ag_i. The whole message is protected by MAC so adversary cannot spoof the content of this message. Figure 5 (b) shows this round. After getting the whole message, an aggregator has the complete topology information of its sensor group, and enough cryptographic data to securely establish communication with in its sensor group.

5.4 Round 3 - Building Each Sensor Group

The goal of this round is to provide four pieces of information to each node in a sensor group: the sensor group it belongs to, its forwarding table for sending and routing data, the pairwise shared key it shares with the aggregator, and the ripple key of the ripple it belongs to. Since aggregator has sensor group topology information, then the aggregator can compute paths and routing tables for each member node. The simple way to do this is to use a breadth-first search algorithm to compute the shortest paths from each member node to the aggregator.

To securely send this information to member nodes, ag_i sends two unicast messages to each node in a breadth-first manner, i.e. it sends two unicast messages to each node in ripple 1 first, to each node in ripple 2 next, and so on. In the first message sent to node s, it includes $< O_1^i, ag_i, r_i, MAC >$. The significance of the information included in this message was discussed in Section 3. The overall result of this first message is that a node now knows the sensor group it belongs to, and its subkey that it shares with its aggregator.

In the second unicast message to s, ag_i includes $< ft_s, KR_j >$, where ft_s is the forwarding table of s and KR_j is the ripple key of the ripple s belongs to. This message is encrypted by the encryption key $k_{e_{s,r_i}}$ generated from subkey of s K_{s,r_i}, and key $k_{m_{s,r_i}}$ is used to generated the MAC. $k_{m_{s,r_i}}$ is also generated from K_{s,r_i}. After receiving the second message,

s can set up its routing tables and the ripple key. It is now ready to route packets for its downstream nodes.

Figure 5 (c) shows this round for building sensor group. Formats of the two unicast messages sent by ag_i are as follows (assume that s is in ripple j):

$$ag_i \rightarrow s \quad : \quad O^i{}_1 | ag_i | r_i | MAC(K_{s,r_i}, O^i{}_1 | ag_i | r_i)$$
$$ag_i \rightarrow s \quad : \quad E(k_{e_{s,r_i}}, fts | KR_j) | MAC(k_{m_{s,r_i}}, fts | KR_j)$$

After this round, the hierarchical WSN has been set up, and secure in-network processing can commence. At this stage, each aggregator ag_i has a one-way hash chain for broadcasting commands to its group. Each sensor node s has a ripple key KR_j, initial number O^i_0, and a subkey K_{s,r_i}. ag_i can use ripple keys to securely send commands to its group as discussed in Section 4. Each sensor node can securely send its sensed data to its aggregator using its subkey K_{s,r_i}.

5.5 Group Dynamic Maintenance

Because of resource constraints, the OHC maintained by an aggregator should not be too long. When this chain is exhausted, a new OHC needs to be set up for the subgroup. To set up a new OHC, the base station or upper level aggregator broadcasts the initial number of the new OHC, and then unicasts the seed and other numbers of the new OHC to the aggregator.

The mechanisms proposed in this paper also support dynamically generating new subgroups. The process of building each subgroup is similar to rounds 1 to 3 for building groups. First, the aggregator generates a new OHC, and broadcasts the initial number of the new OHC to its subgroup. Second, the aggregator unicasts the seed of the new OHC, node connection information of the subgroup, and sub-subkeys to the sub-aggregator of subgroup. The sub-aggregator of a subgroup unicasts subgroup information to each member node in the subgroup, with the similar process described in round 3.

If a node wants to join a subgroup, it needs to know the OHC number used by aggregator, and it needs to share a secret key with aggregator. In addition, it needs the forwarding table for data routing within the group. To add a node into a group, we propose a Kerberos-like protocol[28]. Let's assume new node s wants to join a top-level group SG. The aggregator ag's upper level aggregator is the base station B. First, s detects its neighbors information and the group ID of SG. Then s unicasts the aggregator ID, its neighbors information, and its identity to SG's aggregator ag. ag forwards this message to base station B. B verifies the information from s. If the information is valid, B unicasts s's subkey k_{s,r_i}, and connection information of s to ag. B also encrypts the subkey index r_i with s's encryption key, and unicasts it to ag:

$$B \rightarrow ag \quad : \quad E(\mathcal{K}_{e_{ag}}, k_{s,r_i} | Cnt_s) | E(\mathcal{K}_{e_s}, r_i) | MAC$$

Where Cnt_s is the list of s's neighbor nodes in SG. ag then unicasts the forwarding table of s, the OHC number, and the encrypted subkey index to s:

$$ag \rightarrow s \quad : \quad E(k_{e_{s,r_i}}, fts | O | E(\mathcal{K}_{e_s}, r_i)) | MAC$$

With this information, s can securely receive OHC number O, the subkey k_{s,r_i}, and authenticate ag. The aggregator ag also unicasts the new routing tables to the nodes who need to route s's data, and the nodes who need s to route their data. For s to join a subgroup that is multiple levels down from base station, each level's aggregator needs to forward s's request message to its upper level aggregator, until the message reaches the base station. When the base station sends the $ACCEPT$ information, it sends the uppermost level subkey of s, and each aggregator will generate the subkey of s at that level. Finally, s can get its subkey shared with its subgroup aggregator ag. This solution is not efficient, and not very secure. For example, the DoS attack can be launched by forging join request. More efficient and secure mechanisms to support a node joining and node leaving from a sensor group are part of our future work.

6. PERFORMANCE EVALUATION

To evaluate the performance of security support of in-network processing, we have simulated a prototype on ns2 [29], and implemented cryptographic primitives (one-way hash chain generator and MAC) on Berkeley motes [10]. Network setup overhead, performance of data aggregation and command dissemination, and storage requirements for an aggregator are measured from the simulated prototype, while computation and memory requirements for cryptographic algorithms are measured from the implementation on motes.

6.1 Network Setup Overhead

To measure the overhead of network setup, we constructed a multi-level hierarchical WSN as shown in Figure 6(a). We placed the base station in the center of the network comprised of 250 sensor nodes. The network was first divided into four top-level sensor groups (H1: one-level hierarchical WSN). In a two-level hierarchical WSN (H2), each top-level sensor group was further divided into four second-level sensor groups (4+16 aggregators). In a three-level hierarchical WSN (H3), each second-level sensor group was further divided into four third level sensor groups (4+16+64 aggregators). The area covered by all sensor groups at any particular level was assumed to be same, and the position of an aggregator was chosen to be in the center of its sensor group. We experimented with multi-level hierarchical WSN scenarios: H1, H2, and H3. Figure 6(b) shows the network structure of H1, H2, and H3.

Network setup involves three rounds of message exchanges as discussed in Section 5. We have measured the number of packets exchanged to set up a multi-level hierarchical network in H1, H2, and H3. The number of packets is counted hop by hop. For example, if a packet takes 3 hops to the base station, it is counted as 3 packets. The size of the simulated packet was 36 bytes of which 29 bytes were used for data. This corresponds to the packet size of Berkeley motes [34] with link-layer security [35]. Messages in round 2 were segmented into multiple packets. For each test, we randomly generated 50 network topologies. The numbers reported in figure 7 are corresponding averages.

Figure 7 shows the number of packets needed to set up H1, H2, and H3. The overall network overhead increases with the number of levels, and hence the number of sensor groups. Network overhead increases in rounds 1 and 2 and decreases in round 3 with increasing number of levels. The reason for the increase in network overhead in round 1 is that round 1 is repeated at each level. Recall that round 1 consists of a μTESLA broadcast. While the number of packets exchanged in this broadcast protocol does not depend on

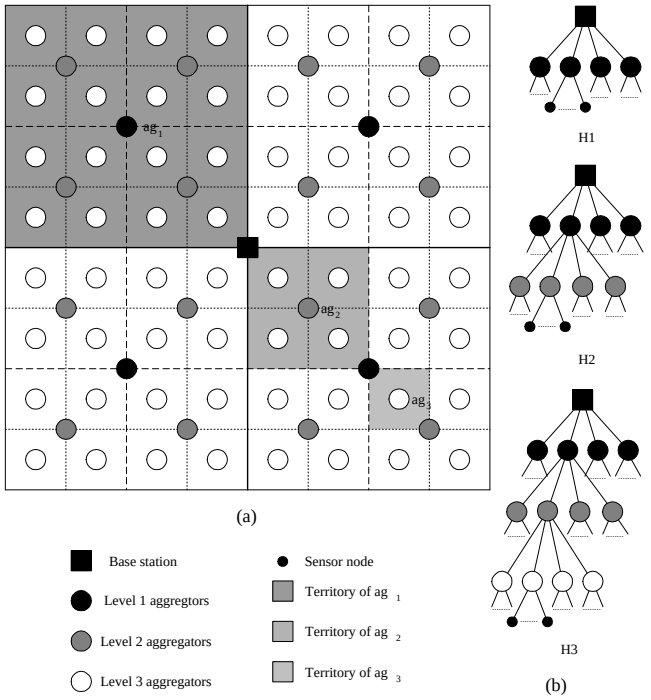

Figure 6: A three-level WSN.

the number of sensor groups at a single level, each aggregator needs to do it to set up smaller sensor groups with in its sensor group. The reason for the increase in network overhead in round 2 with the number of levels is the increase in the total number of sensor groups. Recall that the base station sends a separate unicast message to each aggregator in level 1 sensor groups, aggregators at level 1 send a separate unicast message to each aggregator in level 2 sensor groups, and so on. Finally, the reason for the decrease in network overhead in round 3 with increasing number of levels is the smaller-sized area of each individual sensor group. When the number of sensor groups is large, the size (area) of each sensor group is small. As a result, the unicast messages sent by an aggregator to the sensor nodes in this round take fewer number of hops on average. The important point to notice is

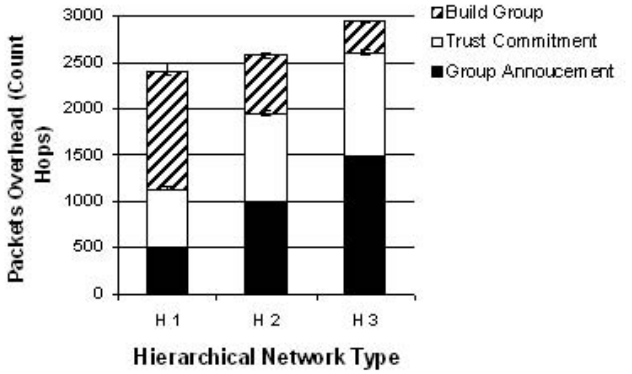

Figure 7: Network setup overhead.

that the overall network setup overhead is relatively small. As we will see in the next set of experiments, this overhead is easily offset by the reduction in the number of packets that in-network processing provides subsequently.

6.2 In-Network Processing Performance

To measure the performance of in-network processing we conducted an experiment in which all sensor nodes report sensor data to their respective aggregators, each aggregator computes a single sensor value (MIN, MAX, average, etc.) from these sensor data received from its group members, and forwards it to its aggregator, and so on. Figure 8 illustrates the performance measured from this experiment. The network setup in this experiment is same as in Section 6.1. The graph plots the total number of packets exchanged in the network as function of the number of messages (sensor data) sent by each node. Numbers reported in this graph include the packets exchanged as a part of the initial network setup. For comparison, we have also included the total number of packets exchanged if there is no hierarchy in the WSN.

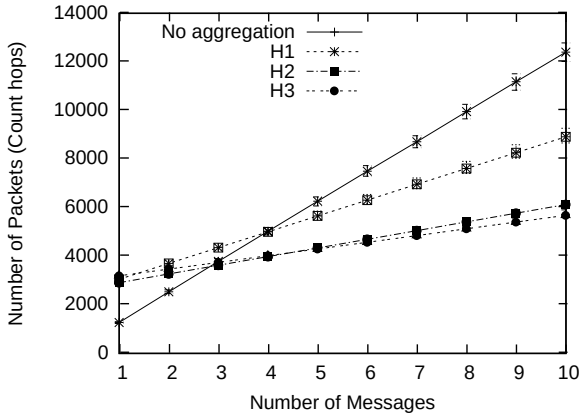

Figure 8: Performance of In-Network Processing.

Although the last experiment shows that there is a fair bit of traffic for setting up the keys, one way hash chains, and forwarding tables to build a group, it is clear from this graph that in-network processing results in a significant reduction in the number of packets exchanged. While the overhead due to initial network setup results in more packets being exchanged at first in multi-level WSN compared to no hierarchy, in-network processing consumes less network bandwidth after just a small number of sensor reports, e.g. after just 5 or 6 sensor data packets reported. Another observation is that by increasing the number of levels, the number of packets exchanged decreases. However, the rate of this decrease diminishes for larger number of levels. For example, there is very little difference in the number of packets exchanged between two-level and three-level WSNs.

6.3 Aggregator Storage Requirements

An aggregator needs to store various cryptographic keys (ripple keys, subkeys), one-way hash chain, and topology information of its sensor group. Memory required to store a one-way hash chain is typically small, because not all numbers in the chain need to be stored. Similarly, since the number of ripples in a sensor group is expected to be low, the

storage requirement for storing ripple keys is also low. On the other hand, the storage requirement for storing subkeys and topology information increases with the sensor group size (number of nodes). We have measured this requirement for a number sensor group sizes. We maintained the same network density in all sensor groups. For each group size, we randomly generated 50 different networks. A subkey was assumed to be 8 bytes long. To store topology information, an aggregator stores the identity of all neighbor nodes for each sensor node.

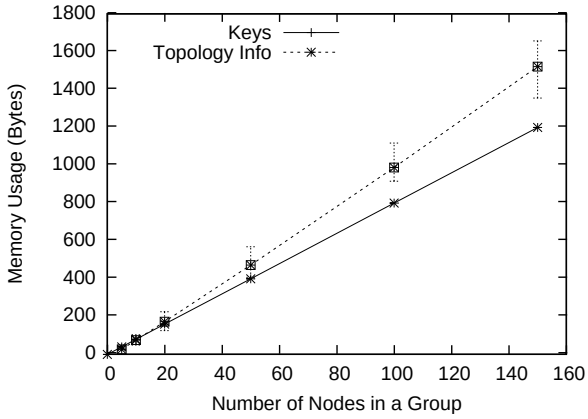

Figure 9: Aggregator Storage Requirement.

Figure 9 plots the memory required in an aggregator as a function of the number of nodes in its sensor group (group size). We observe that storage requirements for storing pairwise shared keys and topology information increases linearly with group size. Suppose there are n nodes in a group, the storage required to store pairwise shared keys is $8n$ bytes. Because the density was the same in all networks we experimented with, the storage requirement for storing topology information was also linear.

A Berkeley mote has 4 KB RAM and 512K EEPROM. Since network topology information is not always needed, it can be stored in EEPROM. If the size of the lowest level sensor group is about 100 nodes, its aggregator will have enough space to save the subkeys in RAM. For the top level aggregator, although it has shared subkeys with all down stream nodes, it interacts only with direct down stream aggregators most of the time. So, it can save most of the shared subkeys that are only occasionally needed in EEPROM.

6.4 Aggregator Command Dissemination

As discussed in Section 4, there are two mechanisms an aggregator can use to send a command to all nodes in its sensor group. One is to use μTESLA and the other is to use ripple keys. We have measured the number of packets exchanged to disseminate a command using these two mechanisms for different sensor group sizes with the same density. Figure 10 shows the result. For comparison, we have also included the number of packets exchanged if the aggregator sends a separate unicast message to each node instead.

The figure shows that both μTESLA and ripple key approaches incur much less overhead than the unicast approach. The ripple key approach outperforms μTESLA for the small network sizes we experimented with. Thus the ripple key approach requires smaller number of packet exchanges than

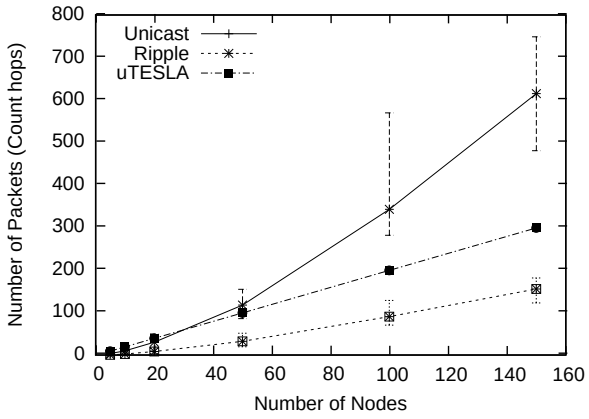

Figure 10: Performance of Aggregator Command Dissemination.

Table 1: Computation and Memory Requirements of Cryptographic Algorithms.

	OHC	CBC MAC	Total
Speed (ms)	4.18	2.22	
Code (Bytes)	1678	1738	1948
Data (Bytes)	136	128	264

μTESLA. Further advantages of the ripple key approach are that it does not require any time synchronization, and does not suffer from delayed key release, which will increase the total time required to disseminate commands. However, if the group size is significantly large, the μTESLA approach will result in smaller number of packets exchanged.

6.5 Resource Requirements of Cryptographic Algorithms

We have implemented the cryptographic algorithms for generating a one-way hash chain and message authentication codes (MAC) using RC5 on Berkeley motes. Motes have a 4MHz processor with 128K flash memory, 4K RAM, and an RFM monolithics TR 1000 radio at 19.2Kbps. We used standard CBC mode for generating MACs. To generate a one-way sequence number chain K_n, K_{n-1}, ... K_0, we used the following algorithm. The base station first chooses a random key K_n and encrypts a well-known plaintext using this key. The resulting cipher is K_{n-1}, which is then used to generate K_{n-2} similarly. This process continues until K_0 has been generated.

Computation and memory usage of these algorithms are shown in Table 1. These measurements show that the computation and memory requirements of the cryptographic algorithms needed to build a hierarchical WSN and support in-network processing are fairly low, and can easily be supported by current sensor nodes such as motes. The program of cryptographic algorithms occupy about 2K flash memory, and they use 264 bytes of RAM for data storage.

7. RELATED WORK

Sensor network security is a critical issue in sensor network research [27]. S. Slijepcevic et. al [33], A. Wood and J. A. Stankovic[36], C. Karlof and D. Wagner[14] provide survey

papers for secure sensor network routing, and discuss many attacks on WSNs. The TinySec[35] project provides link layer security support for Berkeley motes. We referred to the RC5 library of TinySec for our experiments.

In the field of ad hoc wireless networking, previous work on secure routing employs public key cryptography to perform authentication [22][15][32][27]. Unfortunately, resource constraints in WSNs limit the applicability of these current public/asymmetric key standards.

A. Perrig et. al[30] address secure communication in resource constrained sensor networks, introducing two low-level secure building blocks, SNEP and μTESLA. The one-way hash chain in our paper is inspired by μTESLA. D. Liu and P. Ning[20] propose an efficient distribution of key chain commitments for μTESLA. Multi-layer one-way hash chains are proposed for one-way hash chain generation and maintenance in WSNs. Our paper proposes that the base station sends a one-way hash chain to the aggregator to commit authority and enable the aggregator to control its subgroup. Our approach only utilizes μTESLA for bootstrapping the initial number during the setup phase of each subgroup. Both one-way hash chains and ripple keys are employed to secure aggregator commands. This scheme is more efficient than μTESLA for the small subgroups since neither time synchronization nor a delayed release key are required. As far as we know, our approach is the first to address *building* security mechanisms for hierarchical sensor networks with in-network data processing.

J. Deng, R. Han and S. Mishra [7] propose INSENS intrusion tolerant routing for WSNs. INSENS employs multipath routing to tolerate intrusions to the sensor network. INSENS proposes to use one-way hash chains, instead of μTESLA, to authenticate the network setup messages. Our approach to secure in-network processing is agnostic to the underlying routing scheme, and can build on top of INSENS' secure routing scheme.

L. Lazos and R. Poovendran[21], and D. Bruschi, E. Rosti [38] proposed secure multicast for wireless networks. [21] proposed an energy-efficient way of building secure multicast groups for resource-constrained wireless networks. Since our work has not addressed the problem of nodes joining and leaving from a sensor group, our interest is to apply such secure multicast techniques to enhance our future work on secure data dissemination.

8. CONCLUSION AND FUTURE WORK

We have described the design and implementation of a secure, hierarchical wireless sensor network that provides support for secure in-network processing. Our approach delegates trust to aggregators within the network and enables each sensor node within an aggregator's subgroup to trust its aggregator parent. This is accomplished via a collection of techniques, including a novel method for delegating authorization using one-way hash chains and the μTESLA protocol, light-weight shared key establishment, and a novel ripple command dissemination method. We have simulated the proposed network architecture in ns2 and experimented with it under different computing scenarios. We have also implemented some of the proposed protocols on a WSN comprised of Berkeley motes. Performance evaluation from the prototype simulation and implementation shows that the proposed design and protocols for secure in-network processing provide good performance and are lightweight in their resource requirements.

Recently, L. Eschenauer et.al [9], and H. Chan et.al [5] proposed several random key pre-distribution mechanisms for distributed key management in sensor networks. These mechanisms operate without the cooperation of a base station. In the future, we plan to adopt such schemes to design secure, efficient, and DoS attack resistant support for sensor nodes to join or leave a subgroup. In addition, we plan to reduce computation load on the base station, as well as communication traffic between the base station and aggregator nodes, by applying random key pre-distribution mechanisms. Finally, we plan to integrate our security support for in-network aggregation with some popular in-network aggregation platforms such as TinyDB.

9. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments.

10. REFERENCES

[1] H. Abrach, S. Bhatti, J. Carlson, H. Dui, J. Rose, A. Sheth, B. Shucker, J. Deng, R. Han, "MANTIS: System Support for Multimodal Networks of In-Situ Sensors", In Proc. of 2nd ACM Workshop on Wireless Sensor Networks and Applications (WSNA'03), San Diego, CA, Sep, 2003.

[2] ARGUS Advanced Remote Ground Unattended Sensor Systems, Department of Defense, U.S. Air Force, http://www.globalsecurity.org/intell/systems/arguss.htm.

[3] B.J.Bonfils, P. Bonnet, "Adaptive and Decentralized Operator Placement for In-Network Query Processing", IPSN'03, April, 2003.

[4] N. Borisov, I. Goldberg, D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11", ACM MobiCom 2001, pp. 180-188.

[5] H. Chan, A. Perrig, D. Song, "Random Key Predistribution Schemes for Sensor Networks", Appears in IEEE Symposium on Security and Privacy 2003.

[6] Cougar Project: http://cougar.cs.cornell.edu

[7] J. Deng, R. Han and S. Mishra, "The Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks", In Proc. of IEEE 2nd International Workshop on Information Processing in Sensor Networks, IPSN'03, LNCS 2634.

[8] J. Douceur, "the Sybil Attack," In Proc. of the IPTPS02 Workshop, Cambridge, MA (USA), March 2002.

[9] L. Eschenauer, V.D. Gigor, "A Key-Management Scheme for Distributed Sensor Networks", Conference on Computer and Communications Security, CCS'02, Washington DC, USA, November, 2002.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Cullar, K. Pister, "System architecture directions for network sensors",ASPLOS 2000, Cambridge, November 2000.

[11] Y.C. Hu, A. Perrig, D.B. Johnson, "Efficient Security Mechanisms for Routing Protocols", In Proc. of the Tenth Annual Network and Distributed System Security Symposium, NDSS'03, San Diego, CA, February 2003.

[12] Y.C. Hu, A. Perrig, D.B. Johnson, " Rushing Attacks and Defense in Wireless Ad Hoc Network Routing

Protocols", In Proc. of 2nd ACM Workshop on Wireless Security (WiSe'03), San Diego, CA, Sep, 2003.

[13] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robuts Communication Paradigm for Sensor Networks," 6th Conf. on Mobile Computing and Networking, August, 2000, USA.

[14] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", First IEEE International Workshop on Sensor Network Protocols and Applications, May 2003.

[15] J. Kong, H. Luo, K. Xu, D. Gu, M. Gerla and S. Lu, "Adaptive Security for Multi-layer Ad Hoc Networks," Special Issue of Wireless Communications and Mobile Computing". August, 2002.

[16] R. Kumar, V. Tsiatsis, M. Srivastava, "Computation Hierarchy for In-network Processing", In Proc. of 2nd ACM International Workshop on Wireless Sensor Networks & Applications (WSNA'03), San Diego, CA, Sep, 2003.

[17] L. Lamport, "Constructing digital signatures from one-way function", technical report SRI-CSL-98, SRI International, Oct 1979.

[18] L. Lamport, "Password Authentication with Insecure Communication", Communication of the ACM, 24:11, Nov 1981.

[19] J. Liu, J.E. Riech, and F. Zhao, "Collaborative in-network processing for target tracking", EURASIP, Journal on Applied Signal Processing, March, 2003.

[20] D. Liu and P. Ning, "Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks", The 10th Annual Network and Distributed System Security Symposium. San Diego, California. February 2003.

[21] L. Lazos, R. Poovendran, "Energy-aware secure multicast communication in ad-hoc networks using geographic location information", ICASSP 2003, Hong Kong, China, April 2003.

[22] H. Luo, J. Kong, P. Zerfos, S. Lu and L. Zhang, "Self-securing Ad Hoc Wireless Networks," IEEE ISCC 2002, Italy, July 2002.

[23] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks", OSDI, December 2002.

[24] F. Martin, B. Mikhak, and B. Silverman, "MetaCricket: A designer's kit for making computational devices", IBM Systems Journal, vol. 39, 2000.

[25] A. Mainwaring, J. Polastre, R. Szewczyk D. Culler, J. Anderson, "Wireless Sensor Networks for Habitat Monitoring", First ACM Workshop on Wireless Sensor Networks and Applications (WSNA) 2002, pp. 88-97.

[26] A.J. Menezes, P.C.Oorschot, S.A. Vanstone, "Handbook of Applied Cryptography", CRC Press LLC, 1996.

[27] NAI Lab Report, http://www.nai.com/nai_labs/asp_set/crypto/crypt_senseit.asp.

[28] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM 21(12), December 1978.

[29] NS2 web site, http://www.isi.edu/nsnam/ns

[30] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, "SPINS: Security Protocols for Sensor Networks", Wireless Networks Journal(WINET), 8(5):521-534, Sep 2002.

[31] B. Przydatek, D. Song and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks", To Appear in ACM SenSys'03, Los Angeles, CA, Nov, 2003.

[32] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "A Secure Routing Protocol for Ad Hoc Networks", In Proc. of 2002 IEEE International Conference on Network Protocols (ICNP). November 2002.

[33] S. Slijepcevic, V. Tsiatsis, S. Zimbeck, "On Communication Security in Wireless Ad-Hoc Sensor Networks", Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), June 2002, USA.

[34] TinyOS: http://www.cs.berkeley.edu/tos

[35] TinySec: http://www.cs.berkeley.edu/~nks/tinysec/

[36] A. Wood, J.A. Stankovic, "Denial of Service in Sensor Networks," IEEE Computer, 35(10):54-62, October 2002.

[37] L. Zhou and Z.J. Haas, "Securing Ad Hoc Networks", IEEE Network Magazine, vol. 13, no.6, November/December 1999.

[38] D. Bruschi, E. Rosti, "Secure multicast in wireless networks of mobile hosts: protocols and issue," ACM/Baltzer Mobile networks and applications, special issue on Multipoint Communication in Wireless Mobile Networks, Vol.6, No.7, December 2002.

[39] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet", ASPLOS-X conference, Oct. 2002.

[40] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao, "Target Classification and Localization in Habitat Monitoring", In Proc. of IEEE international Conference on Acoustics, Speech, and Signal Processing(ICASSP 2003), Hong Kong, China, April 2003.

[41] Y.J. Zhao, R. Govindan, and D. Estrin, "Computing Aggregates for Monitoring Wireless Sensor Networks", The First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03), Anchorage, AK, USA. May 11, 2003