SUMP: a secure unicast messaging protocol for wireless ad hoc sensor networks

Jeff Janies,* Chin-Tser Huang, Nathan L. Johnson and Theodor Richardson

Department of Computer Science and Engineering,
University of South Carolina,
Columbia, SC, USA
E-mail: janies@cse.sc.edu
E-mail: huangct@cse.sc.edu
E-mail: johnso66@cse.sc.edu
E-mail: trichardson@towson.edu
*Corresponding author

Abstract: Most wireless ad hoc sensor networks are susceptible to routing level attacks, in which an adversary masquerades as a legitimate node to convince neighbouring nodes that it is the 'logical' next hop or is on a 'better' path for forwarding packets and arbitrarily drops the packets forwarded by neighbouring nodes. In this paper, we propose a Secure Unicast Messaging Protocol (SUMP) for wireless ad hoc sensor networks to mitigate the threat of routing level attacks. SUMP groups nodes into levels based on hop count to provide hop-by-hop group authentication using Merkle hash trees. This method allows for varied levels of security in accordance with a node's hop count from the base station and secure, directed unicast communications from the base station to individual nodes. Unlike other such protocols the parent information maintained by a node running SUMP is securely provided by the base station and thus mitigates the threat of the adversary convincing a node to forward through a non-existent path.

Keywords: sensor network; security protocol; Merkle hash tree; authenticated routing.

Reference to this paper should be made as follows: Janies, J., Huang, C-T., Johnson, N.L. and Richardson, T. (2007) 'SUMP: a secure unicast messaging protocol for wireless ad hoc sensor networks', *Int. J. Sensor Networks*, Vol. 2, Nos. 5/6, pp.358–367.

Biographical notes: Jeff Janies is a recent MS graduate in Computer Science and Engineering from the University of South Carolina. He also has a BS from Louisiana State University. His research interests include secure protocol design, MANETs and intrusion detection with a focus on scanning detection.

Chin-Tser Huang is an Assistant Professor of Computer Science and Engineering at the University of South Carolina. He received an MS and a PhD in Computer Sciences from the University of Texas at Austin and a BS in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan. His research focuses on network security, protocol design and verification and distributed systems.

Nathan L. Johnson received a BS and an ME in Computer Science and Engineering from the University of South Carolina. His research interests include network security and multiagent systems.

Theodor Richardson is an Assistant Professor of Computer Science at Towson University, Maryland. He received an ME and a PhD in Computer Science and Engineering from the University of South Carolina and a BS in Computer Science and Mathematics at Bethany College, West Virginia. His current research focuses on signal processing and information security.

1 Introduction

A typical wireless ad hoc sensor network consists of a number of sensor nodes and a base station. Each sensor node is a small, cheap device that is programmed to collect certain types of data, for example, temperature, humidity and light. The base station is an aggregation point for collected data and is viewed as the human interface into the network. The sensor nodes are typically constrained by limited battery power, small memory and low computational ability. On the other hand, the base station is assumed to have greater memory and processing power and does not have the constraints of the sensor nodes.

With their low cost and scalability, wireless ad hoc sensor networks have found a variety of applications. For instance, in the military, such networks are used in target tracking, perimeter monitoring and battlefield survey. Commercial applications of these networks include inventory control and building systems monitoring. However, the constraints on the sensor nodes, plus the open nature of communications in a wireless ad hoc sensor network, make securing networks of this type a challenging task.

A manifest vulnerability of many wireless ad hoc sensor networks is a class of attacks called routing level attacks. In a routing level attack, an adversary masquerades as a legitimate node to convince neighbouring nodes to forward their packets to it and then arbitrarily drops the packets. This class of attacks can potentially strand many well-behaved nodes and cause large holes in the sensing environment. Two examples of routing level attacks, namely black hole attacks and wormhole attacks (Karlof and Wagner, 2003), are discussed below.

A black hole attack is launched when a node (either compromised or masqueraded by an adversary) convinces neighbouring nodes that it is the 'logical' next hop for forwarding packets. The malicious node then arbitrarily drops the packets forwarded by neighbouring nodes. In the case of multihop networks, the impact of such an attack is a hole in the sensing area as nodes begin to forward packets astray from their intended destination. This attack also has the potential to strand large areas of the network that are geographically distant from the malicious node by pulling messages from their intended paths.

In a wormhole attack the adversary attempts to convince nodes to use a malicious path through legitimate means. In some protocols, for instance SPINS (Perrig et al., 2001), when a message is sent from the base station, an intermediate node will forward the message with its identification information attached to the message. A receiving node records the identification information, regards the forwarding node as its parent and proceeds to accept messages only from this parent, similar to the imprinting behavior discussed in Stajano and Anderson (1999). This provides an efficient method to prevent arbitrary rebroadcast of messages. In a wormhole attack an adversary with fast forwarding capabilities can quickly forward a message to nodes in the network. Fast forwarding is accomplished through the collusion of multiple units with fast, out-of-band communication or a strongly powered device with a larger range of communication positioned between the base station and its target nodes. Once the target nodes are convinced that the adversary node is on a 'better' path to the base station, all communications of the target nodes are attracted to the adversary node. An adversary can selectively forward information in an effort to disrupt the sensing environment.

Despite the susceptibility of wireless ad hoc sensor networks to routing level attacks, research in securing this type of network has primarily focused on the development of secure, lightweight protocols that satisfy only confidential communication and integrity of messages with little regard to the threat of routing level attacks. A popular approach is dividing the network into groups or clusters, as discussed in Cerpa and Estrin (2002), Du et al. (2003), Goodrich et al. (2002), Liu and Ning (2003a), Park and Shin (2004) and Zhu et al. (2003). These clusters are generally formed according to locality and require the addition of group and pairwise keys to be used by associated nodes. Furthermore, these approaches require a differentiation of authority where one node, known as the cluster head, has an additional responsibility: it functions as a local aggregation point for all data readings from nodes in the cluster. A node reports its readings to the cluster head and the cluster head is responsible for forwarding those readings to the base station. Thus, the amount of data transfers is reduced significantly, but communications among cluster heads are vulnerable to both black hole and wormhole attacks.

Securing by groups is not the only approach. Perrig et al. (2001) propose a method based solely on membership to the network. All nodes are treated as members to a single group. There are two major components in this approach: The Secure Network Encryption Protocol (SNEP) and μTesla. SNEP provides security for unicast packets from the base station to a given node and requires loose time synchronisation between the two. μTesla provides security for broadcasts, through the use of delayed key distribution and one-way hash functions. Both components provide integrity but are also vulnerable to routing level attacks.

In this paper, we discuss the design and implementation of a novel protocol named Secure Unicast Messaging Protocol (SUMP). There are three goals in the design of SUMP: mitigating the threats of routing level attacks, ensuring that routing decisions are made by the base station and providing a level-wise grouping of nodes in contrast to the locality-wise grouping approach used in other works. Mitigating routing level attacks in the network provides survivability and reliability in hostile environments. By handling routing decisions at the base station, SUMP has the advantage of providing the current global view of the network to the base station while allowing for unicast messaging throughout the network. The level-wise grouping approach allows for varied degree of security according to the 'need' of a node. For example, we note that nodes that are closer to the base station are more likely to be mediating for other nodes that are farther from the base station. Therefore, it represents a greater risk if the nodes closer to the base station are compromised or masqueraded and greater care is needed for securing the communications of these nodes.

2 Environmental assumptions

SUMP makes the following seven environmental assumptions:

1 there are no compromised nodes in the network during the initialisation of the network

2 a collision avoidance scheme exists in the initialisation phase of the network

3 the base station cannot be compromised

4 the base station is not resource constrained

5 a node's individual key and ID information are modifiable before deployment

6 the base station is aware of all nodes in the network prior to deployment

7 nodes are fixed in location.

The first two assumptions are only restrictive for a fixed amount of time. As in Zhu et al. (2003), it is assumed that the minimum amount of time required to compromise a node is T_{min} and the amount of time required to initialise the network is T_{init}. Our approach fixes the initialisation time prior to deployment of nodes such that $T_{\text{min}} > T_{\text{init}}$. Therefore, it is assumed that there are no compromises during the initialisation of the network. Since initialisation is fixed in time, collision could cause loss of data. With a simple yet efficient collision avoidance algorithm this loss can be mitigated. The second assumption is strict and is provided for ease of discussion. In implementation it is assumed that nodes have unique delays between transmissions. This assumption, though less strict than the second assumption, aids in collision avoidance and results in negligible overhead.

The third and fourth assumptions are key assumptions. In SUMP the base station maintains global information about the network, makes routing decisions, flags compromised nodes and assesses connectivity of all nodes. The base station is assumed to be uncompromisable, since it is the human interface into the network and presumably not accessible by an adversary. However, this assumption does not limit the possibility of an adversary intercepting or modifying incoming and outgoing communications to and from the base station.

The fifth and sixth assumptions only apply in so far as the base station being able to uniquely identify the nodes that are potentially present in the network and record their key information. Location information is not provided and no knowledge of connectivity is known prior to deployment. Similar assumptions are found in Liu and Ning (2003a, 2004), Perrig et al. (2001) and Zhu et al. (2003).

The seventh assumption limits the scope of this paper to wireless ad hoc sensor networks consisting of non-mobile nodes. It is assumed that nodes will be deployed in areas that are not easily accessible and left for the duration of sensing. Therefore, routes in the network are static.

3 The secure unicast messaging protocol

Similar to other wireless ad hoc sensor network protocols, SUMP classifies network entities into two types: the base station and sensor nodes. The base station maintains network information and is responsible for making decisions regarding message propagation. The sensor nodes gather readings and maintain a limited amount of information about the network. Due to the constraints on power, memory and computational ability, sensor nodes provide only the most rudimentary of error checking. Since the base station is responsible for routing decisions and sensor nodes simply carry out these decisions, there is little that an adversary can do to affect the route of a message at the node level. The base station is assumed to be uncompromisable according to Assumption 3, therefore an adversary cannot gain routing information from the base station. Furthermore, the route specified by the base station must be followed explicitly in order for the message to be authenticated correctly by the destination, as discussed in Section 3.3.

The SUMP protocol consists of two phases of operation: *initialisation* and *messaging*. Each phase supports different primitives and local storage requirements. The purpose of the initialisation phase is to provide the base station with knowledge of individual node connectivity, density of distribution (with regards to connectivity, not locality) and establishment of paths. The messaging phase is the normal mode of operation used for data collection and dissemination.

The base station and nodes maintain information about the network that is specific to their individual view of the network structure. The base station's view of the network is global and it maintains two primary structures: the *node structure* and the *group structure*. The node structure contains a list of all paths to each node, each node's distance in hops from the base station (called *hop count*), ID and individual key. The group structure maintains information about all groups in the network in a linked list of group elements. A group element contains information about the group including a listing of all nodes' IDs, the distance from the base station to the group (called the *level*) and methods used for group membership authentication. A sensor node only maintains its own key information, a hash of the ID of the next node in its primary path to the base station (securely provided by the base station) and group membership information.

The X-bow Motes™ (MICA2™, 2003) are used for a prototype implementation of SUMP. With the current generation of Motes, certain implementation parameters must be observed. These parameters do not restrict the adversary or reduce the generality in any way, but are added for ease of development and explanation. Firstly, all IDs, authentication values and hash values used are 16-bit integers. Secondly, the length of the symmetric key of a node is 32 bits. Thirdly, due to the limited packet length allowed by TinyOS (http://www.tinyos.net), SUMP supports up to 10 hops of authenticated routing and approximately 1024 nodes. Fourthly, there is a fixed time for initialisation which starts when a node receives the hello message. The value used for this time is known prior to deployment. Furthermore, the space used to store this value is not considered when referring to overall memory requirement since this space is reclaimed after initialisation. Finally, a *Message Type* byte is used to distinguish between message types: hello messages, hello reply messages, base station to node (outbound) messages, node to base station (inbound) messages, hop count change messages, root change messages, key change messages and change parent hash messages. Each of these message types represents a unique message type for operation in the current implementation. For the sake of brevity, only the first five messages are discussed in this paper.

Since it is already shown by Perrig et al. (2001) that RC5 and MD5 are appropriate for use in networks of this design, this implementation does not include an implementation of RC5 and MD5. Instead, RC5-CBC encryption (with a 32-bit block size) is assumed. MD5 is used to generate hash values and checksums. For outbound messages, a checksum is calculated from the message and the ID of the destination node. Therefore, the destination node is the only node that can verify the integrity of the message. For inbound messages, the checksum is a hash of the message's plaintext payload. Checksum values are limited to 8 bits and since the length of hash values generated by MD5 is much larger than the value used, it is assumed that only a subset of the resulting bits are used.

3.1 Initialisation

The initialisation phase in SUMP uses breadth first search similar to the method discussed by Zhu et al. (2003). However, in SUMP the base station is the only entity that initiates the search. The initialisation phase is divided into two steps: *path establishment* and *verification*. In the path establishment step, hop count and paths from the base station to a node are discovered. A node either waits to receive a hello message or forwards hello replies. In the verification step, the base station updates nodes that received an incorrect hop count due to discrepancies in the communication range of nodes. For example, node A has the ability to hear the communications from node B, but A's communication range is less than that of B's so that when A attempts to reply to B, B is out of A's range.

The base station initiates the path establishment step by issuing a hello message containing a count of 0. The structure of this hello message is shown in Figure 1. The count corresponds to the current hop count from the base station. Nodes do not respond to any communications until the hello message is received. Once a node receives the hello message the node will record the hop count into its memory, increment the hop count in the hello message and forward it. The node then replies to the base station with a message containing the hop count recorded and the ID of the node.

Figure 1 Hello message

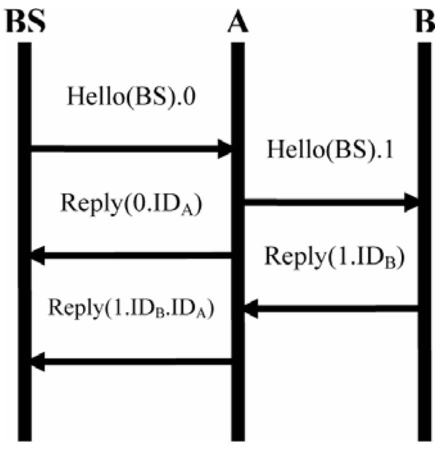
8	16	24
Msg Type	Count	checksum

Following the forwarding of the hello message, the node enters a reply forwarding state in which it will listen for other hello replies, whose structure is shown in Figure 2. When it receives a hello reply from another node, it concatenates its own ID to the end of the message and retransmits the message. A node will not respond to a hello reply message that contains its own ID in order to avoid the formation of infinite routing loops that deplete resources. A sequence diagram of path establishment is shown in Figure 3.

Figure 2 Hello reply message

8	16	24	32
Msg Type	Count	checksum	■
ID1		ID2	
ID3		ID4	
ID5		ID6	
ID7		ID8	
ID9		ID10	

Figure 3 Message sequence diagram of path establishment



When the base station receives a reply from a node it finds the path traversed by this reply based on the ID list found in the reply message. The path derived from the first reply message received from a node is stored as the primary path to the node and all paths derived from replies received after the first reply are stored as alternate paths. These alternate paths are used to reduce packet loss and enhance the survivability of the network as dynamic events occur.

After the expiration of a preset period of the path establishment step, all nodes and the base station enter the verification step, in which the base station distributes a hash of the ID of each node's next closest hop in its primary path and compares the node's recorded hop count with the length of the first path received in the path establishment step. If a discrepancy is found, the base station rectifies it by sending a hop count change request encrypted with the individual key of that node. The hop count change request includes the ID of the destination node and the hop count value determined by the base station, as seen in Figure 4. This ensures that the hop count is representative of a symmetric path between the base station and the destination node. In order to assure that these requests are delivered to their destination every node that overhears a hop count change forwards the message. If a node receives a hop count change request intended for it, the node updates its hop count accordingly.

Figure 4 Hop count change request

8	16	24
Msg Type	ID	Count

3.2 Group authentication

Once verification of appropriate hop count is completed, the base station groups nodes according to hop count. All nodes of the same hop count are members of the group with the corresponding level value. Therefore, nodes are grouped topologically by connection and not geographically by location. For example, the group representing level one is comprised of all nodes with a hop count of one. Once this grouping is completed, the base station computes the group's key information and distributes it to all nodes in the group. Hop counts of nodes can be used to generate key material and determine membership status in the network. For example, the network in Figure 5 can be represented in a general tree structure where the level of a node in the tree corresponds to its hop count as shown in Figure 6.

Figure 5 A sample connected wireless ad hoc sensor network

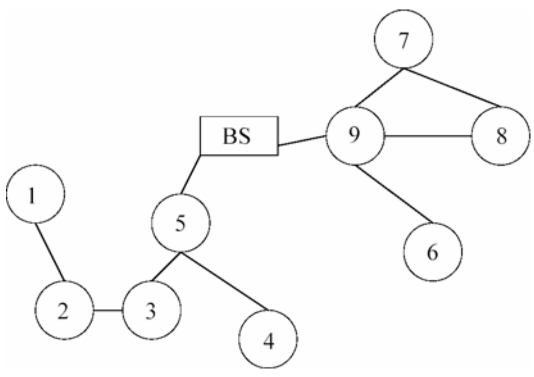
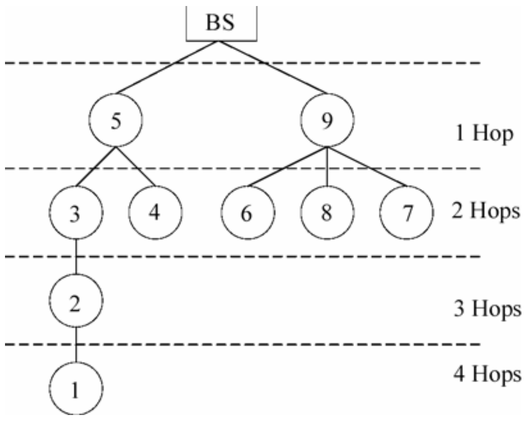


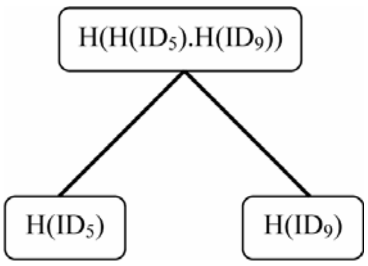
Figure 6 Tree structure for authenticated levels



This method lends itself to routing by level, in which only one node per hop forwards a received message. When a message is transmitted from the base station to a node, the primary path to the node is used to ensure that the message is not arbitrarily rebroadcast throughout the network, thus conserving network resources. However, in order to prevent malicious redirection of packets, a group authentication method is required to verify that a message is indeed from a legitimate source and intended to be routed through a member of the group.

Several group authentication methods are available for membership testing, for example the one way accumulator proposed by Benaloh and de Mare (1993) and the key chain commitment proposed by Liu and Ning (2003b), but due to the constraints of the sensor nodes many of them are not feasible. Merkle (1980) hash trees, in contrast, provide secure group membership authentication with low computational overhead and low storage requirement. A Merkle hash tree uses a secure one-way hash function to generate a binary tree in which the members of the group are represented as the leaf values. The tree is formed by concatenating the sibling values and hashing the result to form a parent element of the tree. As is seen in Figure 6, the network is divided into levels based on hop count and each level represents a group. The group that represents level 1 consists of nodes 5 and 9. Level 1's group is represented as a Merkle hash tree in Figure 7. In this representation, the hash values of nodes 5 and 9 form the leaves of the tree. The remainder of the tree is formed according to the following three rules. Firstly, the tree is a balanced binary tree. Secondly, if an element is a leaf of the tree, then its value is the hash of a node's ID. Finally, if an element is not a leaf of the tree, then its value is the result of hashing the concatenation of its two children elements' values.

Figure 7 Merkle hash tree of level 1



As a result of constructing a tree with the above rules the root value is a representative number of the entire group membership and with a node's ID can be used to authenticate a message. The base station maintains a representation of the Merkle hash tree of each level. The root value, height of the tree and node's ID are the only values stored by the node. The tree cannot be produced unless all node IDs are known. Since nodes do not store the IDs of other nodes in the network, an adversary cannot capture a node and reconstruct the membership tree from the root value and the node's ID.

3.3 Messaging

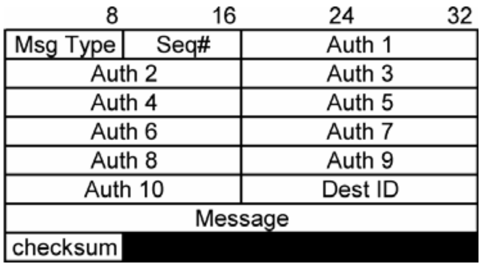
With the establishment of Merkle hash tree at every level, we are ready to start the messaging phase, which aims to securely unicast a message hop-by-hop from the source towards its destination. There are two cases to consider: outbound communication and inbound communication.

In the case of outbound communication, since the path to every node is known to the base station and the nodes are grouped according to hop count, the base station can communicate with a node by encrypting messages according to the established primary path to that node. The base station concatenates the ID of the destination node to the message and encrypts the result with the key it shares with the destination node. Then the h authentication values, where h is the height of the tree at the destination node's level, are attached to the beginning of the message as per the Merkle hash tree algorithm. These authentication values are the values needed by the node to reproduce the root value from already known information, that is, the sibling values of elements in the path from the node's hashed ID to the root of the tree. The base station then uses the next node on the primary path from the destination to the base station to encrypt the message further. For an intermediate node between the base station and the destination node the base station encrypts the entire message with the shared key of the intermediary node and concatenates the authentication values of intermediate node with the resulting message. This results in the encapsulation of the original message, *M*, in a message to the intermediary node. For example, if the path from the base station to node 2 is {5, 3, 2}, the message produced is:

$$E_{K_5}\left(\begin{array}{l}\{\text{authValues5}\}.\\ E_{K_3}\left(\{\text{authValues3}\}.E_{K_2}\left(\{\text{authValues2}\}.\text{ID}_2.M\right)\right)\end{array}\right)$$

Upon receiving a message of this structure, a node uses its key to decrypt the message and attempts to authenticate the message to determine if the message is to be forwarded by the node. If the node successfully authenticates the message using the authentication value in the message, it checks if the decrypted message begins with its ID. If so, the decrypted message is for the node and will not be retransmitted; otherwise the node forwards the decrypted message to the next hop. The structure of outbound messages in the current implementation of SUMP is shown in Figure 8.

Figure 8 Messaging phase outbound message



In the current implementation of SUMP, the header is of fixed length, which means there are always 10 auth values regardless of the number of actual authentications needed. Therefore, unneeded or previously processed authentication values are zeroed out. Note that as long as the number of auth values does not exceed the upper bound, header with variable lengths is entirely feasible with just a field of header length added and this choice of fixed-length header is only for the sake of implementation convenience. An individual node only uses as many non-zero *auth values* as necessary to authenticate a message for that hop. After finishing authentication, the node zeros out the used auth values. For instance, if a given level has a Merkle hash tree of height 2, then only the first two non-zero auth values are used in authentication. If the message is authenticated, the node decrypts the remaining auth values, message field and checksum and forwards the resulting packet to the next hop. The sequence number is added to a node's transmitted list only if the message is authenticated. This mitigates premature processing of a message which can result in a breakdown of communication.

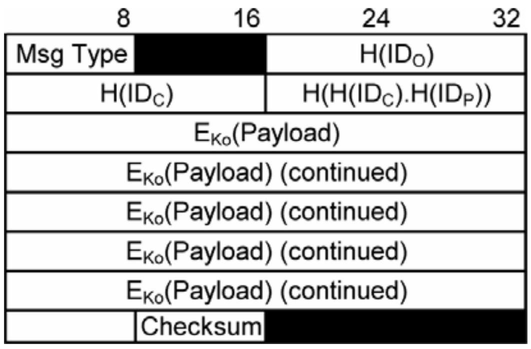
In the case of inbound communication, a message flooding approach could be used, in which every node that overhears a message originating from a node encrypts the payload and retransmits. Since the base station knows the path from every node, it can easily decrypt the message by applying the key of all nodes on the path from the node to the base station in reverse order. Thus, the chain of keys used to decrypt the message provides a method of message authentication.

However, although the message flooding approach provides the benefit of increased likelihood of delivery based on its avoidance of routing-level attacks, it is an inefficient method for inbound message delivery, as it requires as many as $n - 1$ messages for a network consisting of *n* nodes in the worst case. Alternately, one can conceive a directed unicast inbound messaging scheme that requires knowledge of surrounding nodes. To implement directed unicast messaging from a node to the base station, the node is required to store a routing table which consists of the ID of the parent node to which inbound messages should be forwarded and the ID of child nodes from which inbound messages will be received. However, this scheme has the following drawbacks. Firstly, this scheme is subject to the routing-level attacks because of the requirement of storing parent information in a node. Secondly, with the network topology undetermined until deployment time the size of the routing table is variable. Keeping a routing table of variable size is expensive and will squeeze the already limited node memory. Furthermore, if a node keeping such a routing table is compromised, the adversary gains explicit knowledge of some portion of the network topology.

To avoid the problems of the above inbound message delivery scheme, we extend the mechanisms in place for outbound messaging to allow secure unicast inbound messaging. To accomplish this, the Merkle hash tree approach can be locally employed to provide directed authentication of inbound messaging. By examining the primary paths established during initialisation the base station chooses an appropriate parent for each node in the network. In principle, the parent node *P* for a given node *C* is the next node on *C*'s primary path to the base station. Once the base station has chosen the parent node *P* for node *C*, it securely distributes a hash of the parent's ID, $H(\text{ID}_P)$, to node *C*. Note that this hash value does not reveal the ID of parent node *P* to node *C*, because according to the property of a hash function, it is extremely hard to derive ID_P from $H(\text{ID}_P)$. Therefore this scheme is not subject to routing-level attacks.

Each node uses this information to securely direct inbound communication towards the base station hop-by-hop. To transmit an inbound message, node *C* first encrypts the message with symmetric key that it shares with the base station. Then *C* supplies the hash of its own ID $H(\text{ID}_C)$, the hash of the ID of the originating node $H(\text{ID}_O)$ (in this case $H(\text{ID}_C) = H(\text{ID}_O)$ because *C* is the originating node) and a hash of the concatenation of this value with the hash of its parent's ID, $H(H(\text{ID}_C)\ .\ H(ID_P))$. Note the value $H(H(\text{ID}_C)\ .\ H(\text{ID}_P))$ is representative of a Merkle hash tree root vale for a group of only two members. The structure of the inbound message is shown in Figure 9.

Figure 9 Messaging phase inbound message


8
16
24
32
Msg Type
H(ID$_O$)
H(ID$_C$)
H(H(ID$_C$).H(ID$_P$))
E$_{Ko}$(Payload)
E$_{Ko}$(Payload) (continued)
E$_{Ko}$(Payload) (continued)
E$_{Ko}$(Payload) (continued)
E$_{Ko}$(Payload) (continued)
Checksum

When node P receives an inbound message, it first attempts to authenticate the message by computing the concatenation of a hash of its own ID with the hash value $H(\mathrm{ID}_C)$ provided in the message. If this value is equivalent to the value $H(H(\mathrm{ID}_C) \,.\, H(\mathrm{ID}_P))$, node P replaces the values $H(\mathrm{ID}_C)$ and $H(H(\mathrm{ID}_C) \,.\, H(\mathrm{ID}_P))$ with its own ID hash $H(\mathrm{ID}_P)$ and $H(H(\mathrm{ID}_P) \,.\, H(\mathrm{ID}_{PP}))$ (where ID_{PP} is the ID of the parent of P) and forwards the message.

When the base station receives a packet, it can determine the originating node by utilising the $H(\mathrm{ID}_O)$ value contained in the message. Then the base station decrypts the payload of the message with the symmetric key that it shares with the originating node.

3.4 *Dynamic node events*

Dynamic node events, such as node joins and node deaths, are open problems. A reseeding or addition of more nodes, represents an interesting challenge to SUMP. It is assumed that the communication channel is free of malicious activity only during the initialisation phase and therefore the initialisation method discussed previously is not effective for adding new nodes to an established network after the expiration of the initialisation phase. Although an initialisation key to encrypt all initialisation messages would alleviate this concern, the secure distribution of such a key is resource consuming. On the other hand, node death gives rise to several questions. How is node death detected? What happens to nodes that are unable to receive communications from the base station due to the death of a node? Since a node's death may alter the hop count of one or more nodes, does the base station have to regroup all nodes that had paths containing the dead node? These questions, among others, present an interesting challenge to this protocol. Since the base station maintains a list of alternate paths to a node, communication can be reestablished easily, but the group membership may no longer be representative of the network. The problems with node joins and node deaths can be mitigated by dynamic reseeding discussed in Section 3.5 on scalability.

SUMP is able to handle scheduled sleep of nodes. There are two possibilities: either the base station sends secure unicast messages to instruct individual nodes to enter sleep mode or the base station has knowledge about the preset sleeping schedule of nodes. Since the base station has knowledge about all the available paths to each node, it can arrange to get around sleeping nodes or wake them up.

3.5 *Scalability*

For the purpose of providing scalability, SUMP can be extended to allow each node to operate on multiple network hierarchies. Instead of using all 10 auth value fields, 1 field could be spared to provide space for a hierarchy identification code. Hierarchy identification requires only that a node can identify whether it has a router entry for that hierarchy. If it has an entry, it will process the message just as in the current protocol. If it does not have an entry, it will ignore the message. Using this schema of multiple hierarchies, nodes can be dynamically reseeded as necessary simply by adding another entry in the routing table. This also allows extensibility by virtue of the fact that each hierarchy will have its own 'base station'; this may not necessarily be the true aggregation point but a more powerful node that is sparsely distributed with the standard nodes. Each of these more powerful nodes will form its own hierarchy over the standard nodes and use the same hierarchy schema (which can be similarly extended) to communicate with the true aggregation point, namely the base station of the whole network. In the seeding process, the messages must be updated to include the hierarchy identification code and the nodes must be allowed to respond to multiple seeding messages to alert each base station of its proximity.

4 Evaluation

In this section, we evaluate the initialisation and messaging phase of SUMP. Their evaluation is done separately, since an evaluation of both mechanisms at once proves to be cumbersome and lacks overall benefit to the reader. We first employ a multiagent simulation to evaluate the initialisation phase and then use a comparative analysis between SNEP and SUMP to evaluate the messaging phase.

4.1 *Initialisation evaluation*

As discussed in Section 3.1, the purpose of path establishment step in the initialisation phase of SUMP is to discover all possible paths from each node to the base station, including the 'best' path. There have been works on finding the shortest path with low flooding cost, (e.g. Allard et al., 2003; Cheng and Heinzelman, 2003). However, the 'best' path in our context is not necessarily the physical shortest path, because the physical shortest path may not be usable due to node failure or power limit. Rather, recall from Section 3.1 that the first reply message received from a node is stored as the primary path to the node, which represents the fastest and reliably usable path. To demonstrate the effectiveness of this path establishment scheme in achieving the above purpose, a NetLogo (see http://ccl.northwestern.edu/netlogo/) simulation is employed. NetLogo is a tool for multiagent modelling. In the simulation the base station and all nodes are represented as parallel agents with a particular communication range. We want to point out that the NetLogo simulation has its deficiencies. Firstly, NetLogo views each node as an agent and gets to them in no particular order when accomplishing the task. In a real environment the order a message would be received would be based on proximity since the message would arrive at the closest nodes first and they would begin processing the message. Secondly, the initial setup of a dense network will take a long time under NetLogo since all nodes must share the same pool of resources and do not have their own dedicated processor and memory to accomplish the task. However, this simulation is sufficient to verify whether all possible paths between the base station and the nodes are discovered in the initialisation phase.

In this simulation, a broadcast method is used so that the initial 'Hello' message is propagated to each node. Each node, in turn, responds by transmitting a 'Hello Reply'. In a simulated network composed of the base station and n nodes, $(n+1)$ 'Hello' messages (transmitted once by the base station and each node) and at least n 'Hello Reply' messages (transmitted at least once by each node) exist in parallel, but since each node that overhears a 'Hello Reply' will rebroadcast a 'Hello Reply' the number of replies can grow tremendously. Figure 12 is a perfect example of this rapid growth. With each node being near, it shows how the redundancy is built into this scheme; on the other hand, it also portrays how quickly the message count can increase in a dense node deployment.

We conducted three test runs to evaluate the performance of the path establishment scheme under different topologies. The first test run is a simple network with a large number of nodes and two cycles, as shown in Figure 10. This topology provides several paths to many of the nodes and a choice for the base station. Note that despite the cycles no infinite loops will be formed since a node will not retransmit a message that already contains its own ID in the path list. The second test run is similar to the first, but with fewer nodes and a slightly different topology, as shown in Figure 11. The third test run shows that with a dense network where many nodes are connected directly to each other, as shown in Figure 12. In this network there exist many possible paths via which a small number of nodes are used to propagate a message to a given destination. In all the three test runs, it is obvious that our path establishment scheme is able to make the resulting network as connected as possible and report the path information to the base station. Thus the effectiveness of our path establishment scheme is demonstrated.

4.2 Messaging evaluation

For analysing the messaging phase, SUMP is compared to SNEP (the unicast protocol presented by Perrig et al. (2001)). SNEP is known for its efficiency and security in wireless ad hoc sensor networks. The reason of this exclusive comparison is twofold. Firstly, SNEP is well documented in its weaknesses and strengths. The authors provide extensive analysis of its potential and overhead in this specific network environment. Secondly, SNEP and SUMP are designed exclusively for unicast messaging and therefore, the comparisons made are fair and accurate to the protocols' design.

Figure 10 Test run 1

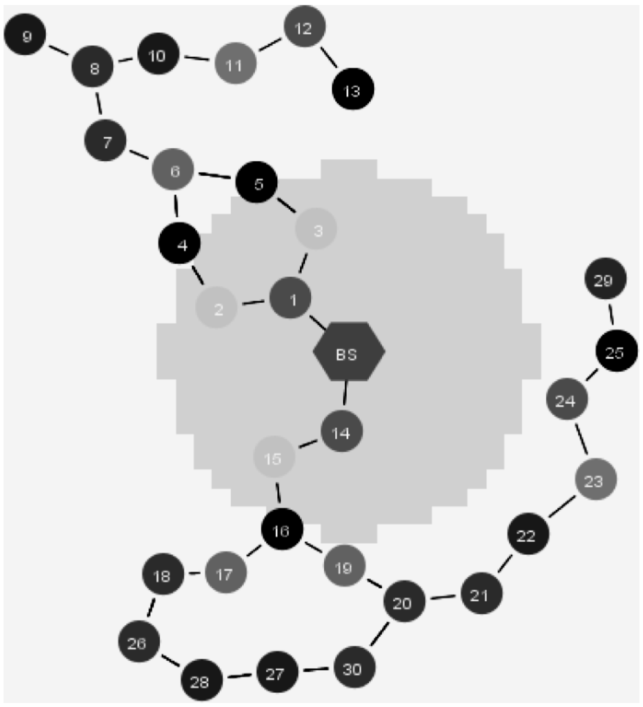


Figure 11 Test run 2

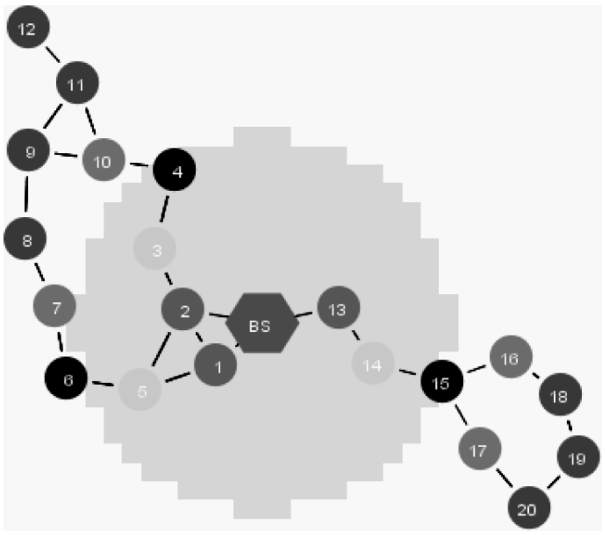
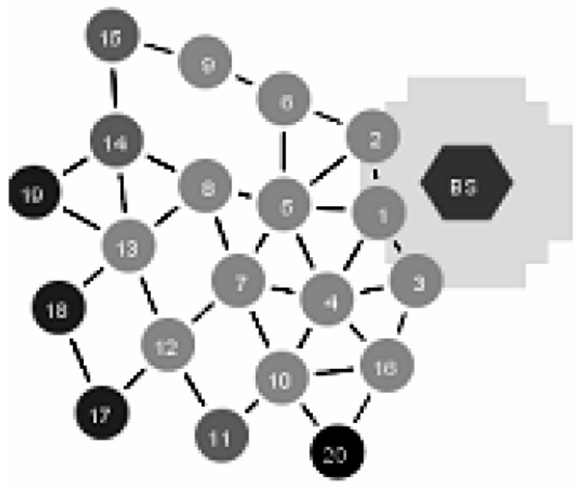


Figure 12 Test run 3



4.2.1 Storage requirements

The storage requirements of SUMP are comparable to SNEP. As previously mentioned, the ID and Merkle hash tree root values are 16 bits in the current implementation and the symmetric key shared with the base station is 32 bits. The tree height is stored as a 16-bit integer, the hash of the node's parent's ID is a 16-bit integer and the hop count is an 8-bit integer, resulting in a total storage requirement of 104 bits. Although SNEP requires nodes to store only a symmetric key and counter, it also requires synchronisation between the two communicating parties and knowledge of the parent. Therefore, if it is assumed that the counter is stored in a 16-bit integer and the key size is equivalent, SNEP requires only 48 bits of storage. In contrast, SUMP requires 56 more bits (7 bytes) of storage than SNEP, but SUMP has the strength in that it does not require synchronisation.

4.2.2 Communication overhead

Communication overhead is comprised of two main components: size and frequency. For the purposes of this analysis, size only refers to the additional bytes added to a message to provide security. Obviously SUMP incurs much overhead in outbound communications due to the addition of the authentication values, which alone require 20 bytes. In comparison, although SNEP requires only 8 bytes of overhead, it does not provide any directional information for outbound communications. Due to the utilisation of directional information SUMP performs better then SNEP with regards to frequency of outbound communication. In this context frequency refers to the number of times that a packet is transmitted beyond what is required for the message to be received by the destination node. In the case of outbound communication, SUMP has little additional overhead with regard to frequency since the message is directed. SNEP requires all nodes to rebroadcast every message in order to ensure that a message is received. However, in SUMP the deliverability of the message is dependent on proper forwarding by all intermediate nodes. Therefore, breakdowns in communication require reestablishment of paths.

In the case of inbound communications, both perform equally well with only a limited amount of directional information provided by the transmitting node. In the case of SUMP this directional information is the hash of the ID of the next hop, the hash of the source node and the hash of both the previous two hash values concatenated. In the case of SNEP this is done with some other unique identification information.

4.2.3 Resistance to routing level attacks

SNEP requires nodes to gather their own parent information and thus is vulnerable to routing level attacks. In contrast, SUMP does not share this vulnerability. Since the information about a node's parent is provided by the base station in a secure fashion, these forms of attacks are limited in impact. In the case of the black hole attack no disruption is caused in either direction. For outbound communications, if a path is compromised the base station can use an alternate path. For inbound communications, the only way that an adversary can fool a node into believing that it is the next hop is by breaking the symmetric key shared by base station and a node and by discovering the authentication chain for the same node. Therefore, SUMP mitigates these routing threats.

4.3 Weaknesses

We note that SUMP has two identifiable weaknesses. Firstly, outbound messages are limited to 32 bits of data and thus the number of messages is potentially increased. This results in greater consumption of power to transmit the same amount of information. Secondly, SUMP limits the sizes of groups. Due to the fixed message length, only a limited number of authentication values can be included in a message.

The limit on the size of outbound messages does not hinder this method as greatly as would be expected. Due to the limited amount of information stored on a node, individual updates require very little space. However, the overhead of the security implementation does deplete resources. This depletion of resources is mitigated by the fact that unicast messages follow specific paths and are completely ignored by nodes that are unrelated to the communication.

This implementation of SUMP does not allow for all possible network configurations. It can be seen that with the mote's limited packet size and 16-bit authentication values, only ten authentication values can be stored in each message. Additionally the payload for these messages is limited to 32 bits. With the introduction of the message type byte in the message and the limited storage requirements this size is sufficient for maintenance operations such as hop count updates, root updates, key updates and parent hash updates.

5 Concluding remarks

In this paper, we introduce SUMP, a SUMP for wireless ad hoc sensor networks. Security and efficiency are the two emphases in the design of SUMP. Through the use of Merkle hash trees, SUMP provides security for these networks with survivability. As shown in our experiments, SUMP is applicable even on very constrained devices such as the X-bow Mote.

The strengths of SUMP are manifest in the following three regards. Firstly, it is demonstrated that SUMP is not susceptible to black hole and wormhole attacks that would otherwise allow an adversary to disrupt communication in the network. Secondly, it is shown that very little storage is required by SUMP for sensor nodes to securely route outbound messages. Finally, communication overhead is alleviated by avoiding arbitrary rebroadcast of messages.

Future expansions of this work include the introduction of a secure broadcast mechanism that capitalises on the global knowledge maintained by the base station, the development of node-to-node communication methods and the extension of a sleep state to save power. It is desirable to develop a lightweight secure broadcast protocol in addition to the proposed SUMP and we believe that the global knowledge maintained by the base station in SUMP can be exploited to reduce the computational cost. Moreover, the group establishment could allow for subnetwork routing to route messages among nodes in the same group.

Acknowledgements

This work is supported in part by an AFRL/DARPA grant (FA8750-04-2-0260) and an NSF grant (CNS-0551650). The authors thank Dr. John Zachary and our colleagues at NIST for their support.

References

Allard, G., Jacquet, P. and Viennot, L. (2003) 'Ad hoc routing protocols with multipoint relaying', *Proceedings of ALGOTEL 2003*, Banyuls, France, May.

Benaloh, J. and de Mare, M. (1993) 'One way accumulators: a decentralized alternative to digital signatures', *Proceedings of Advances in Cryptology (EUROCRYPT'93)*, Vol. 765, *Lecture Notes in Computer Science*, Springer.

Cerpa, A. and Estrin, D. (2002) 'ASCENT: adaptive self-configuring sensor networks topologies', *Proceedings of INFOCOM 2002*, New York, NY, USA, June.

Cheng, Z. and Heinzelman, W. (2003) 'Flooding strategy for target discovery in wireless networks', *Proceedings of Sixth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'03)*, San Diego, California, September.

Du, W., Deng, J., Han, Y. and Varshney, P. (2003) 'A pairwise key pre-distribution scheme for wireless sensor networks', *Proceedings of the Tenth ACM Conference on Computer and Communication Security (CCS 03)*, Washington, DC.

Goodrich, M., Tamassia, R. and Hasic, J. (2002) 'An efficient dynamic and distributed cryptographic accumulator', *Proceedings of the Information Security Conference (ISC '02)*, Vol. 2433, *Lecture Notes in Computer Science*, Springer.

Karlof, C. and Wagner, D. (2003) 'Secure routing in wireless sensor networks: attacks and countermeasures', *Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications*, May.

Liu, D. and Ning, P. (2003a) 'Establishing pairwise keys in distributed sensor networks', *Proceedings of the Tenth ACM Conference on Computer and Communication Security (CCS 03)*, Washington, DC.

Liu, D. and Ning, P. (2003b) 'Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks', *Proceedings of Tenth Annual Network and Distributed System Security Symposium (NDSS 03)*, San Diego, CA.

Liu, D. and Ning, P. (2004) 'Multi-level μTesla: a broadcast authentication system for distributed sensor networks', *ACM Transactions in Embedded Computing Systems (TECS)*, Vol. 3, No. 4, pp.800–836, November.

Merkle, R. (1980) 'Protocols for public key cryptosystems', *Proceedings of the IEEE Symposium on Security and Privacy.*

MICA2™ Specifications Data Sheet (2003) 'Document 6020-0042-04', Available at: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-04_B_MICA2.pdf Rev. B, May.

Park, T. and Shin, K. (2004) 'LiSP: a lightweight security protocol for wireless sensor networks', *Proceedings of the ACM Transactions on Embedded Computing Systems*, Vol. 3, No. 3, August.

Perrig, A., Szewczyk, R., Wen, V., Culler, D. and Tygar, J. (2001) 'SPINS: security protocols for sensor networks', *Proceedings of Seventh Annual ACM International Conference of Mobile Computing and Neworks (MOBICOM 2001)*, Rome, Italy, July.

Stajano, F. and Anderson, R. (1999) 'The resurrecting duckling: security issues for ad-hoc wireless networks', *AT&T Software Symposium.*

Zhu, S., Setia, S. and Jajodia, S. (2003) 'LEAP: efficient security mechanisms for large-scale distributed sensor networks', *Proceedings of the Tenth ACM Conference on Computer and Communication Security (CCS 03)*, Washington, DC.