Secure Directed Diffusion Routing Protocol for Sensor Networks using the LEAP Protocol

VijayRaman Kumar[1], Johnson Thomas[1] and Ajith Abraham[2]
[1]*Department of Computer Science, Oklahoma State University, USA*
[2]*Department of Computer Science and Engineering, Chung-Ang University, Korea*

Abstract. Sensor networks are finding multiple applications and are being increasingly deployed in the real world. These sensors are fragile with limited computational and storage resources and communicate with each other and a base station through routing protocols. Routing protocols in sensor networks seek to minimize energy consumption and do not take security into consideration, leaving the network vulnerable to malicious outside attacks. Due to the resource limitations of sensors, the standard security protocols cannot be applied. A number of key management protocols for sensors have been proposed. However, these key management protocols do not consider the routing problem. In this paper we modify an existing key management protocol called LEAP and integrate it into the directed diffusion sensor routing protocol to produce a secure routing protocol for sensor networks. We show that the proposed protocol can protect the network from malicious outside attacks. Simulation results also show that there is a slight overhead in terms of energy expenditure for the proposed protocol. The other overhead is a slightly increased packet size.

Keywords. Sensor routing protocol, key management, secure routing

1. Introduction

Recent advances in miniaturization, low-cost and low-power design have led to the development of sensor networks that are formed by hundreds or thousands of these wireless unattended sensors and actuators [1]. Usage scenarios for these devices range from real-time tracking, to monitoring of environmental conditions, to ubiquitous computing environments, to *in situ* monitoring of the health of structures or equipment [2]. The sensors can be installed at pre-selected locations and data can be collected from specific location with the help of the neighbors. Wireless sensor networks are extremely constrained in terms of memory, processor, and power. [3]. The extreme constraints of these devices make it impractical to use legacy systems [3].

In a sensor network a node communicates with other nodes that lie within the transmittable range to accomplish the given tasks. Due to the constraints of sensors, the sensor network routing protocols are much simpler than any other network routing protocols. A number of routing protocols have been proposed for sensor networks [4] [5] [6] [7]. Directed Diffusion [5] is one of the energy efficient routing protocols that have been proposed. However, the security vulnerabilities of these sensor routing protocols has been largely ignored in the literature. The directed diffusion algorithm does not use any mechanism to protect the nodes from outsider attacks. In this paper we identify the types of attacks that are possible on the directed diffusion routing algorithm. We extend the directed diffusion routing algorithm by integrating a secure key management mechanism into the routing protocol. We show that the secure directed diffusion routing protocol can provide protection against these attacks and we also investigate the overheads associated with the proposed secure diffusion routing protocol. The work reported in this paper is based on the Berkeley's MICA motes and the TinyOS sensor platform [8].

In the next section we review existing routing protocols for sensor networks. In section 3 we discuss the key management protocol that we integrate into the directed diffusion routing protocol. The attacks on directed diffusion and the proposed approach is presented in section 4. In section 5 we discuss the performance of the new protocol and analyze the security implications. Simulation results are presented in section 6 four or by conclusions.

2. Routing Protocols for Sensor Network

A number of routing protocols for sensor networks have been proposed. The TinyOS beaconing protocol constructs a breadth first spanning tree rooted at a base station [1]. The Geographic and Energy Aware Routing (GEAR) routing protocol and the Greedy Perimeter Stateless Routing (GPSR) route packets based on the geographic location of the nodes. The GPSR uses greedy forwarding at each hop, forwarding to the neighbor closest to the destination. When holes are encountered where greedy forwarding is impossible, GPSR recovers by routing around the perimeter of the void [1]. In Minimum Cost Forwarding all the nodes in the network maintain a *cost field*. The *cost field* specifies the minimum cost required to reach the base station. To forward a packet to the base station, the node checks the cost field associated with a neighbor and chooses the minimum cost route. The *cost field* can store any metric like hop-count, energy, latency or loss. A number of clustering based protocols have been proposed. LEACH (*Low-Energy Adaptive Clustering Hierarchy*) leverages clustering to efficiently disseminate queries and gather sensor readings to and from all nodes in the network [1]. LEACH organizes the nodes into clusters with one node acting as a cluster head. The nodes within a cluster send the collected data to its cluster head and the cluster heads of the entire cluster communicate to aggregate the data. The aggregated data is then forwarded to the base station. In Rumor routing [4] the query from the base station is flooded in the entire network. The data events in response to the queries are shared by the nodes. The base station does not depend on a single link to receive the events. Another proposed protocol is directed diffusion [5]. We present this protocol in detail as this is the particular protocol that is the focus of our study.

2.1. Directed Diffusion

Directed diffusion consists of several elements: interests, data messages, gradients and reinforcements [5]. The base station floods the sensor network with a query about the interested events. The ‘interest‘ query specifies the sensing task. The data messages are the events generated by a single or a group of nodes in response to the query sent by the base station. The interest queries are disseminated throughout the sensor network as an interest for named data. This dissemination sets up the “gradients” within the network to draw events. A gradient is a direction state created in each node that receives an interest [5]. The node, which generates the events, sends the events back to the base station along multiple gradient paths. The directed diffusion algorithm assumes that each node knows its location once deployed. For example, a query for locating a vehicle will specify the area to be monitored, the interval at which the sensor should respond with events and the total duration of sensing (expireAt time). The sensor that detects the wheeled vehicle might respond with the type of vehicle detected, the location, the strength of the signal, confidence in the event or detection and a event time stamp

For each active task the base station broadcasts this message periodically. The initial message for setting up the gradients and fetching the data will have a much larger interval. Intuitively, this initial message is thought of as exploratory; the base station tries to determine if there indeed are some nodes in the specified region that sense the task specified.

Each node in the network maintains an interest cache. The interest cache contains the information about the interest it received. Interest cache does not have information about the base station but the one-hop neighbor from which it received the interest. There are several fields in the interest cache. A time stamp field indicates the time stamp of the last event received. The interest cache also contains several gradient fields, up to one per neighbor. Each gradient contains the data rate field which contains the data rate requested by the corresponding neighbor, derived from the interval attribute. It also maintains a duration field derived from the time stamp and the expiresAt attributes.

When a node receives an interest, it checks the interest cache to see if the interest already exists. If no matching entry exists in the interest cache, then the node creates one interest entry and stores the information about the interest. This entry has a single gradient towards the neighbor from which it received the interest. The node has to distinguish the neighbors in order to send the data at the requested rate. If the entry already exists then the time stamp and expiresAt field are updated. When a gradient expires, it is removed from its interest entry. After storing the information about the interest, a node will send the interest to all its neighbors.

The gradient specifies the data rate and the direction in which to send events. In summary, interest propagation sets up state in the network to pull down the data events from the source node. The rules for interest propagation are application specific [5]. Due to the multi-path transmission of the interest, it is not possible for an adversary to prevent the interest information from reaching the nodes in the network [5].

The node which lies in the specified area, tasks its sensors to begin collecting samples. If the node finds the target then it will search its interest cache for a matching interest entry. If a matching interest is found then the node will look at the data rate parameter for all the gradients and forward the data at the rate specified. It will initially be slower for all gradients. So, all the neighbors receive a copy of the event. The source node unicasts the events, to all the neighbors for which it has a gradient. If the data rates of downstream nodes are different, then the source node interpolates the messages it is sending to the high data rate neighbor. The interpolated message is send to the low data rate neighbor.

The node which receives the events from the source, attempts to find a matching entry in its interest cache. If a match does not exist then the data message is dropped silently. If there exists a match, the received message is added to the data cache and the data message is sent to the node's neighbors [5].

The data message will eventually reach the base station. The base station reinforces one particular neighbor, and that neighbor, reinforces one of its upstream neighbors. The reinforcement continues till the message reaches the source node. The reinforcement is nothing but the same interest message with an increase in the data rate. The higher data rate events allow high quality tracking [5]. Any node in the network can send a positive reinforcement to its upstream neighbor if that node consistently sends the unseen event to it.

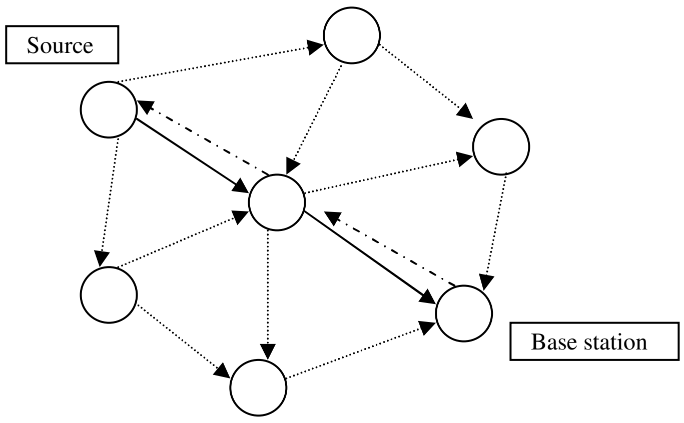

Figure 1. Positive Reinforcement

A node on the data flow path can also be negatively reinforced if that node cannot consistently supply new events to the downstream nodes. An interest message, with the initial exploratory data rate is send to the node which needs to be negatively reinforced. Like positive reinforcement, the negative reinforcement message is also forwarded to the source node. The nodes receiving this message change their data rate of the neighbor. For example, let us consider from the following figure that the data flows in the path A→C→E→G and the link A→C is congested. Now node E will receive the data message from the node D (since the other link is congested). This prompts node E to negatively reinforce node C and positively reinforce node D.

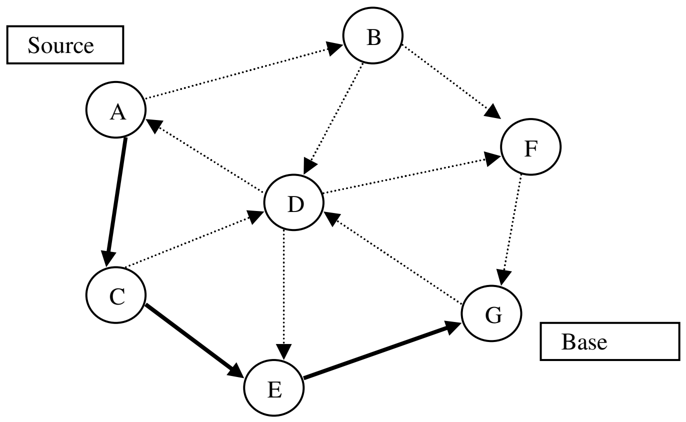

Figure 2. Alternate Path

3. Key Establishment

End-to-end mechanism is used in the conventional network for message authenticity, integrity and confidentiality. But end-to-end security mechanism is not feasible in sensor network because the communication is mainly between the one-hop neighbors. Since the sensor nodes have limited computational power, it is also not possible to use a 128 bit encryption mechanism.

In symmetric key algorithm only one key is used for both encryption and decryption. It requires a shared key between the nodes. TinySec [9], a security mechanism provided for sensor nodes in TinyOS environment uses the symmetric key algorithm. SPINS [10], another security protocol for sensor network also uses a symmetric key algorithm to provide message authentication, integrity and confidentiality. In this paper we propose the integration of LEAP [11], a security protocol for sensor networks, with directed diffusion. Unlike the above protocols, LEAP restricts the security impact of a node compromise to the immediate network neighborhood of the compromised node. Furthermore, in LEAP different types of messages exchanged between sensor nodes can have different security requirements. A single keying mechanism as in other protocols is not suitable for meeting these different security requirements. For example the announcement from the base station may have a different security requirement than a packet from a source node to the base.

3.1. Outline of LEAP

Four different keys are used in each node to provide various level of security. Each node shares a pair-wise key with all its neighbors. This key is used for the secure communication between a node and one of its neighbors. All the nodes in the sensor network share a common Global key with the base station. The base station uses this key to encrypt the interest message and all the nodes in the network uses this key to decrypt the announcements from the base station. The nodes store the interest information in their interest cache and then encrypt the message using the global key to further broadcast it. The communication cost is reduced by using this key. Each node also has a unique individual key. The individual key is used for secure communication between a node and the base station. The base station uses this key to verify the messages sent by this node and also for updating the global key of the node. LEACH also uses a cluster key. In this work, the cluster key is not used as in directed diffusion all the communications are between one-hop neighbors. In this paper we discuss how security can be provided by using three keys.

The LEAP protocol uses Pseudo-Random functions to derive the keys. The base station derives one master key K and this key is loaded into all the nodes in the network before deployment. Each node is also assigned a unique id and a global key before deployment. From a master key a node can derive other keys for various security purposes. For example, a node can derive K_0 for encryption and use K_1 for authentication [11]. The individual key K_u for a node u is generated from the master key. [11].

Since the communication in sensor network is always among the neighbors, the pair-wise shared key is used more than any other key. LEAP assumes that the time required to establish all the keys in the network (T_{est}) is less then the time required (T_{min}) by the adversary to compromise one or more nodes. When a node u is deployed, it tries to discover its neighbors by broadcasting a HELLO message which contains its id and waits for each neighbor v to respond with an ACK message including the identity of node v. The ACK from every neighbor v is authenticated using the individual key K_v of node v, [11]. Node u computes its pair-wise key with v, K_{uv}. K_{uv} serves as the pair-wise key. No message is exchanged between u and v in this step.

The global key can be established before the deployment of sensors. Since all the nodes are going to share the same key with the base station, the loading of this key is done before deployment.

4. Proposed approach

We assume that the sensor network is static, i.e., sensor nodes are not mobile. The base station, acting as a controller (or key server), is assumed to have sufficient resources. We also assume that the base station is equipped with a powerful transmitter and it can send the message to any node in the network in one-hop. Every node has space for storing sufficient keys. Furthermore, we assume that the base station will not be compromised. We also assume that the sensor network is dense and there will be at least two neighbors (with enough power) for each node. The probability to break more than one pair-wise key of the same node is very minimal.

An efficient security mechanism for supporting communications in directed diffusion is proposed. The security requirements not only include authentication and confidentiality but also robustness and survivability, that is, the sensor network should be robust against various security attacks, and if an attack succeeds, its impact should be minimized.

With the introduction of the keying mechanism in directed diffusion, only the nodes which have the keys can send and receive packets in the network. The size of the keys used in LEAP is 8 bytes [11]. It has been proved that a laptop class adversary can crack the message which was encrypted using the symmetric key algorithm [12]. Furthermore, it is possible for an adversary to obtain the keys and launch the above attacks as an insider attack. Such attacks are very difficult to find. Hence, the use of symmetric key algorithm alone is not enough for providing security in sensor networks. Our scheme minimizes the effects of an attack. The incorporation of a modified version of LEAP into directed diffusion protects the network from outside attacks. The attacks are therefore primarily insider attacks. We assume the attacker has the global key for instance.

4.1. Cloning Attack

If the goal of the adversary is to receive the data messages generated by the source node then the adversary has to announce that it a base station. In figure 3, the base station (C) encrypts the interest message using the global key and broadcasts. The attacker forges the interest message with itself listed as the base station using the global key. The adversary will also change the source id in the packet to its own id and transmit the message. The neighboring nodes forward this message in the network to set up gradients along the path. The adversary is also in a position to generate a pair-wise key shared between the nodes in the data flow path to decrypt the data message sent by the source node.

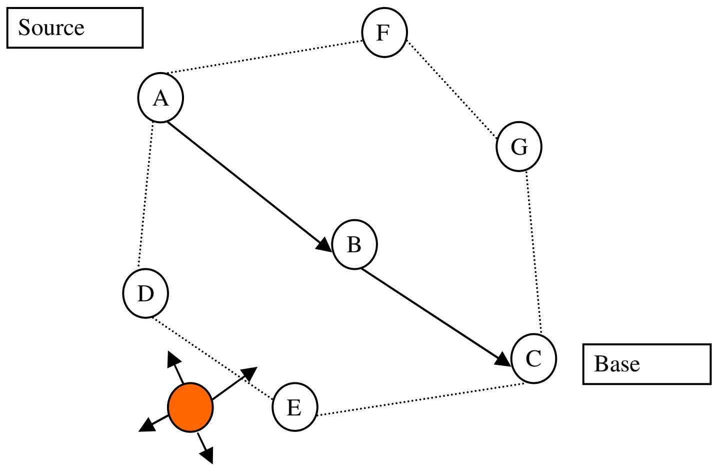

Figure 3. Cloning Attack

If each node in the network maintains the distance information of the base station, then the probability for this type of attack can be reduced. The distance of a node can be calculated using the signal strength of the message it transmitted. RSSI (Received Signal Strength Indicator) values localize the sensor network [13] [14]. Localization is a scheme by which all the nodes in the sensor network will learn about the location it is situated using one or more mobile nodes. RSSI is a signal that indicates the strength of the incoming (received) signal in a receiver. Using the signal strength, the distance of the node is found. Sensor nods such as the MICA mote already come with the hardware necessary to calculate the RSSI. The signal strength is directly proportional to the remaining battery power. We assume that the base station is equipped with long lasting power, so the signal strength never reduces. The nodes can find out the distance of the base station with reasonable accuracy.

During the initial set-up time, the base station should broadcast the "Hello" packets powerful enough to reach all the nodes in the network. Each node receives this message and stores the RSSI value of the base station. Each node has to calculate the RSSI value of any future message received from the base station and compares it with the existing value. It becomes very difficult for a node to pose as the base. Due to noise, the nodes in the network should accept the packets from a base if the RSSI is within a MAX and MIN value.

4.2. Flow Suppression

Flow suppression is denial of service attack. The easiest way to suppress a flow is to spoof negative reinforcements. The adversary simply sends the negative reinforcement to the node which is delivering data at a high rate. The adversary has to find out the unique id of the downstream node and the interest information to launch this attack.

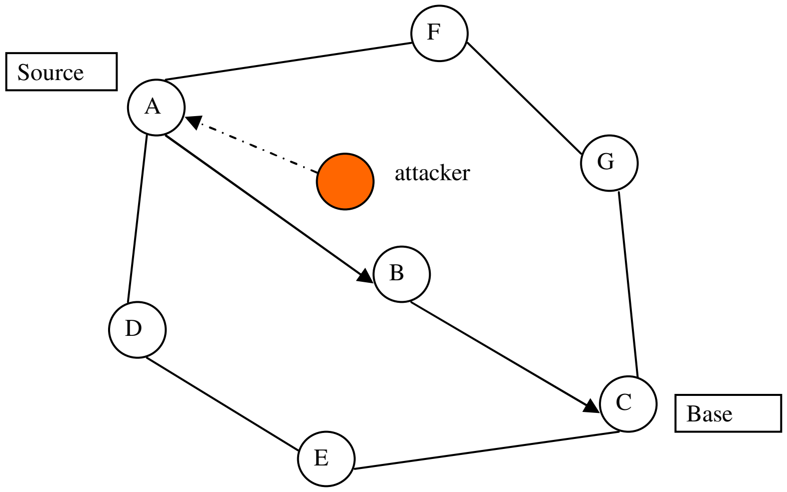

Figure 4. Negative Reinforcement

In figure 4 the data flow path is A→B→C. If the path between A and B or B and C is congested for a time, then the paths A→D→E or A→F→G will provide the required data to the base station at a faster rate. If the nodes E or G supplies the new events consistently to the base station then the base station will positively reinforce E or G. After positively reinforcing one of E or G, the base station also negatively reinforces B. This will cause the node B to negatively reinforce the source (A) thereby suppressing the data flow.

If the adversary breaks the pair-wise key shared between the nodes A and B then, it can simply send a negative reinforcement to node A posing as node B. Node B receives this information and changes the data rate value in the interest cache. This is a simple instance of denial of service attack. For negative reinforcement information to be valid and processed further, the node which received the negative reinforcement should also receive the information about negative reinforcement from at least one more neighbor. For example, if B sends a negative reinforcement to A and if it has to be valid, then the adversary has to compromise at least one other neighbor of B and obtain its pair-wise key. The attacker will be new node that is not recognized. LEAP allows new nodes to join the network only on the basis of a new master key [11].Since we have assumed that it will be difficult for the adversary to break more than one pair-wise shared key of a single neighbor, the negative reinforcement attack can be prevented. If node A did not get the confirmation of negative reinforcement from one other neighbor then it decides that the negative reinforcement information is not authentic and sends the information to the base station for key revocation.

4.3. Path Influence

The adversary can attract all the traffic through itself by announcing the availability of more energy and/or the availability of a high quality link to reach the base station (by sending a powerful signal). All the nodes start sending the packets to this adversary. After receiving the packets, the adversary can choose to forward packets selectively or change the packet information and forward it. The adversary can also launch this attack by spoofing positive and negative reinforcement.

Let us consider the node A as the source and node C as the base station from In figure 5 if the adversary breaks the pair-wise key shared by nodes E and B, it might try to attract all the nodes in the network by sending a fake announcement to B .The message can be regarding the availability of high quality path to base station and/or high remaining battery power. On receiving the information, B further forwards to all its neighbors. The nodes will forward everything to the attacker.

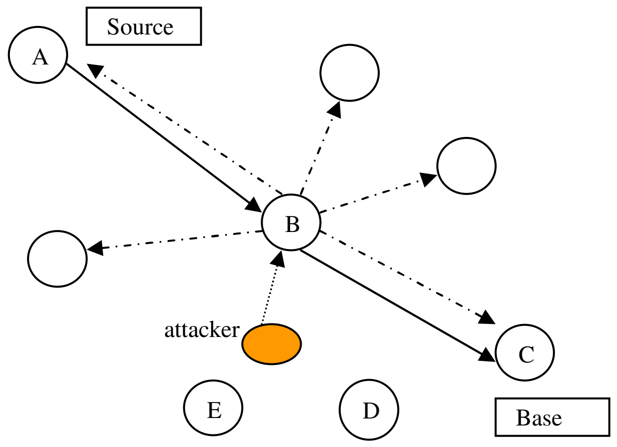

Figure 5. Path Influence

LEAP assumes that the adversary takes T_{min} time to break a pair-wise key of a node and it is larger than the time it takes for all the sensor nodes to establish the pair-wise keys. During this initial setup period each sensor node will construct a hash table with the node id as the unique key. The table will also contain transmission rate and RSSI value for all its neighbors. The hash table can be searched in O(1) time. The nodes store the data during the pair-wise key setup time. Each node appends its remaining battery power with every packet so that the neighbor nodes can construct the table. Note that because of our assumption that the adversary takes at least T_{min} time to break one or more keys, the initial packets from the nodes will be authentic.

If the adversary announces the availability of high quality link to a neighbor, the receiving node will find out by comparing the table value with the received signal strength value. The RSSI value is expected to decrease with the decrease in remaining battery power. By comparing the value of signal strength from a node and the value of RSSI exist in the table, a node can differentiate an authentic and fake message with a high probability. The node can then send the information to the base station for key revocation.

The remaining power of each node is inversely proportional to the data rate. However, a neighboring node may be involved in multiple communications. The message from a node will therefore also include its total transmission rate. The node in the data flow path can therefore calculate the remaining battery power and determine whether the node is an attacker. Note that this calculation is necessary only when a neighbor sends an announcement regarding its battery power and transmission rate.

However, knowing the battery power of the immediate neighbor alone is not enough. An adversary might try to send an announcement using other nodes id. For example, in figure 10 assume the adversary knows the pair-wise key between E and B. The attacker might send a message about its power availability to node B faking its identity as E. In the message content, the adversary might announce that node E received the information from node D, that is, node D is a desirable node to route packets through. The proposed approach will not allow node B to determine whether the message received is a fake or authentic.

If each node has the knowledge of the remaining battery power of all other nodes in the network, then the comparisons can be made for the farther nodes too. If nodes in the network share the information regarding the battery power and RSSI of their neighbors then this type of attack can be prevented. While forwarding the data message each node has to attach the remaining battery power value of all its neighbors in the packet. On receiving this message, each node stores the information. Note that no additional transmission and receiving is necessary in this case. The overhead is the increased packet size and the need for extra memory space in each node to store the information about two hop neighbor nodes information.

4.3.1. Calculation of Remaining Energy

The ADC7 of the Mica mote gives the battery voltage [15]. For our simulation we used the two components provided by nesC language to calculate the remaining power, computeRates(..) and PowerMonQuery. Using the data sheets provided for the MICA mote, power required for all the operations can be calculated from [16].

4.4. Selective Forwarding

In selective forwarding the attacker, after receiving the data messages from the upstream neighbor does not forward all the messages. The adversary can modify the data message or inject its own data message to the downstream nodes. In the figure below, let us consider that the data flows from A to B to C. If an attacker breaks the pair-wise key shared between B and A, then the adversary can modify the data message and then send it to B. B further forwards the data message to the base station.

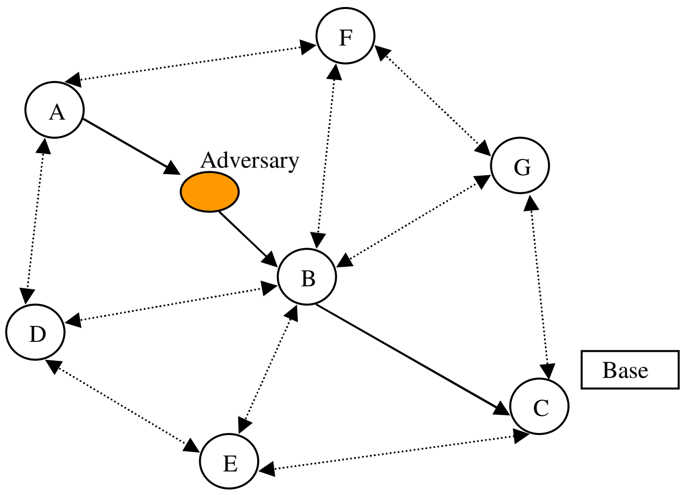

Figure 6. Selective Forwarding

In figure 6 the dashed lines represent the low data rate paths. The data rate of path A→B→C (say 100 events /second) is different from the low data rate paths that receive data at 1 event/second. Let us assume that the adversary breaks the pair-wise shared key used between A and B. Now the adversary gets the access to the data messages from the source node. It can modify the data messages and send to B or selectively forward it to B. After receiving the modified packet the node B forwards the packet to the base station. Node B also sends the data messages to neighbors D, E, F, and G at the original data rate. The source node A also sends the data messages to its neighbors other than B (F and D). The data messages received from the source are stored in the data cache of each node. Node D will therefore have data messages from source A and node B in its data cache. By implementing a simple check mechanism to compare the two data messages, we can find out the selective forwarding attack. Note that the same comparison will be performed in all the neighbors of the node B.

The data message comparison is application dependent. For example in a chemical leak monitoring application, the base station might spread interest information about the possible gas leak at specific location. The base station expects either "YES" or "NO" as the message from the source node. Positive reinforcement will take place on the paths with a 'YES' reply. The low data rate paths may receive the approximation of the 100 events, not the 100th event. If the adversary modifies the "YES" data message from the source to a "NO" and forwards it, then the nodes can find out the modification. Different techniques are used in other applications to find out whether the data is same or not. A confidence rate can be used for matching two events. The source node compares the stored wave form of an event with the sensed event and finds out the confidence value. In the data cache each node maintains the last seen event (data message) from the neighbor. A data comparison method can check the data message received from a node and the data message present in the data cache, allowing a node to determine if the data message is fake. The data comparison technique is application dependent. The basic idea is that there will be a relation between the two events of the same interest.

4.5. Node Inclusion and Exclusion

4.5.1. Node Addition

If a new node is to be added in the network then it should be loaded with a new global key, the master key K_i and $K_j(u)$ individual keys where j is $1< j <=m$. The procedure for calculating the individual key of the node is as discussed above. The base station can calculate the individual keys from the master key. The base station unicasts a "Hello" packet. The added node receives this message and calculates the signal strength value for the base station and stores it (so that it can authenticate any future message from base station).

The new node added in the network will broadcast its "Hello" packet with its id specified in the packet. The node which receives this packet replies by broadcasting "Hello" packets. Let us consider that the nodes u_1 to u_N are present in the network and node v is the new node. Now node v has to set up pair-wise key with all its neighbors' u_1 to u_M where $M <=n$. Since the nodes u_1 to u_M already have the individual key of v, each can calculate the pair-wise shared key with node v. The procedure is same as the initial pair-wise key set-up of LEAP. The nodes u_1 to u_M add node v in their neighbor list.

After addition, the added node needs to know the interest information to build the gradients in the network. This problem can be solved if the added node's neighbors pass the interest information to the new node. This will cause the added node v to receive the data messages from its neighbors. Note that the node v also builds the table for storing the RSSI and the battery power remaining of its neighbors from the message it received from them while setting up the pair-wise keys.

4.5.2. Node Leaving

If there is node failure in the network, directed diffusion uses negative reinforcement to truncate a path if a node does not respond within a specified amount of time. In our approach, each node will maintain the remaining battery power information of all its neighbors. When the battery power of a node reaches a threshold level, then the neighbors of the node removes the id of the node from their list. This will cause the base station to select another path to get the data from the source node.

4.5.2.1. Negative Reinforcement

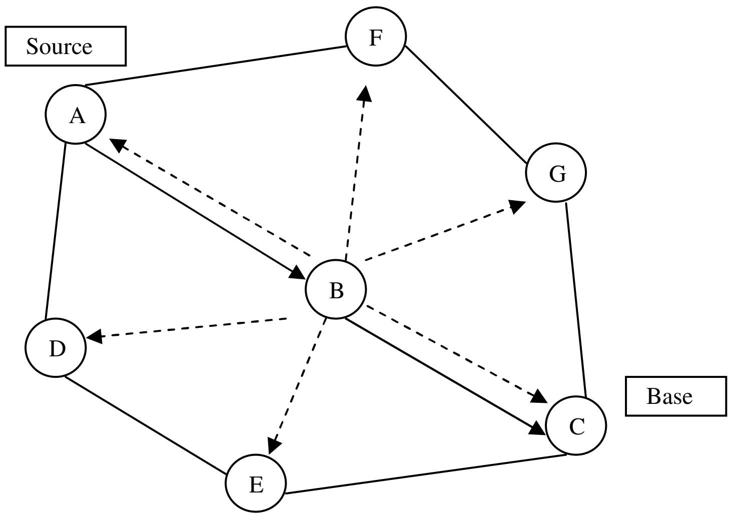

Figure 7. Negative Reinforcement

If link AB is congested, B will send a negative reinforcement to A and the negative reinforcement information to all its neighbors. Even if one or more nodes did not have enough power to forward the information to the source node, the availability of multiple paths ensures that at least one other neighbor gives the information to the source.

4.5.2.2. Selective Forwarding

The availability of multiple nodes to check the data from the source node ensures that the nodes in the data flow path sends the original data received from the source node to the base station.

4.5.2.3. Path Influence and Cloning Attack

Thus the solutions for these two attacks are not affected by the loss of one or more nodes.

5. Performance and Security Analysis

5.1. Storage Requirement

In this section we determine the overheads in implementing the LEAP algorithm in directed diffusion routing. In our scheme, each node has to store an individual key, a group key and *d* pair-wise keys, where d is the number of one-hop neighbors. The number of bytes required depends on the density of the sensor networks. If transmission radius r of a node is *r* and assuming nodes are evenly distributed with a density *d* , the total number of keys a node has to store is $d\pi r^2+2$. Each key is 8 bytes in length [1]. Thus the total bytes required for a node will be, $8d\pi r^2+16$ bytes. Although memory is a very scarce resource for the current generation of sensor nodes (4 KB SRAM in a Berkeley Mica Mote), for a reasonable degree *d*, storage is not an issue in our scheme.

The nodes in the network need to store the RSSI value and the remaining battery value of all its one and two-hop neighbors. The RSSI and the remaining battery power take a byte each to store. So, the total bytes required for a node becomes, (d+2)*8 + (D + d)*2 bytes. Here D is the 2nd hop neighbor and d is the one-hop neighbor. Since there is no cluster key in our scheme, considerable amount of storage space in each node is saved when compared to LEAP (d per node. However, the introduction of multiple Master keys increases the storage requirement. Depending upon the number of node additions, the memory required to store the keys increases.

5.2. Computational Cost

The pair-wise key and individual key set-up is done after the deployment. Each node has to calculate on average d pair-wise keys and one individual key. If there are N number of nodes in the sensor network then the number of key calculations will be $N(d_p +1)$, where the subscript *p* stands for the cost to calculate a pair-wise key

The number of encryptions and decryptions is an important factor in determining the efficiency of our scheme. The base station encrypts the interest message using the global key and broadcast the message. The nodes after receiving the interest message, decrypts and stores the interest in the interest cache. This is followed by encrypting the message using the global key and broadcasting further. Each node will broadcast the interest message and neighbors decrypt the message. If the number of nodes that get the interest message from the base is M and L is the average number of nodes reachable from other nodes, the total number of encryptions and decryptions will be $1_e + Md_e + M_d + L_d$. A subscript *e* stands for the cost to encrypt and a subscript *d* stands for the cost to decrypt

Besides the pair-wise key and individual key, each node also has to calculate the individual key of all its neighbors for verification. Given that this cost is represented by the subscript *i*, the number of calculations is Nd_i.

The computation cost to establish the keys in the network will be less for our proposed version when we compare with LEAP. Nodes don't have to calculate the cluster keys in our case. Our case requires some additional comparison for keeping the network secure. All the packets from the base station will be compared with the stored value of battery power remaining and RSSI value. This cost goes up when the number of interest messages go up.

5.3. Communication Cost

Since the interests are broadcasted, each node has to transmit only once (15 mA [17], [16]). However, each node will receive the same interest from all of its neighbors (d*12 mA). The same applies for the data messages. Hence with the number of interests the communication cost increases. For I interests, the transmission and receiving cost for broadcasting will be $I(N_t + d_r)$. Here the subscript *t* is the cost for transmitting and r is the cost for receiving

For receiving and transmitting data messages, the communication cost will depend on the number of paths reinforced positively and the data rate requested by the downstream nodes. Let d be the number of neighbors of a node which is in the non data flow path and m the number of nodes in the non data flow path. Let D be the number of nodes involved in the data flow path (including the base). Let the maximum data rate imposed by the base station be *mdr*. The power required will be,

$$((D\text{-}2)*mdr*15\text{ mA} + (D\text{-}1)*mdr)*12\text{ mA} + m_i*15\text{ mA} + md*12\text{ mA}$$

The communication cost for sending and receiving queries or data message is same in our scheme as in directed diffusion. Only when a node wants to send a negative reinforcement, it has to multicast the information about sending the negative reinforcement to all other neighbors. However, a negative reinforcement message is not expected often in the network, so the overhead occurred by a few negative reinforcement can be compromised for increased security.

5.4. Packet Size

The following is the packet used by the TinySec [9], the keying mechanism proposed for sensor networks. The packet format of TinyOS is also given for comparison purposes. For our purpose we will use the TinySec packet format with additional information like GPS coordinates and remaining battery power.

Table 1. TinyOS packet format

Dest	AM	Len	Grp	Data	CRC
(2)	(1)	(1)	(1)	(0-29)	(2)

Table 2. TinySec packet format

Dest	AM	Len	Src	Ctr	Data	MAC
(2)	(1)	(1)	(2)	(2)	(0-29)	(4)

The nodes in our network have to include the remaining battery power. Hence the packet format will be:

Table 3. Packet with power information.

Dest	AM	Len	Src	Ctr	Power	Data	MAC
(2)	(1)	(1)	(2)	(2)	(1)	(0-29)	(4)

The TinyOS packet format in table 1 does not have the source node information; this leaves the entire network vulnerable to the outsider attack. Any node can inject a packet with little effort. To detect transmission errors, TinyOS senders compute a 16-bit cycle redundancy check (CRC) over the packet. The receiver recomputes the CRC during reception and verifies it with the received CRC field. If they are equal, the receiver accepts the packet and rejects it otherwise. However, this CRC does not provide any security from attacks. Since we have included the MAC field, CRC is not needed. Active message types are similar to port numbers in TCP/IP. The AM type specifies the appropriate handler function, to extract and interpret the message on the receiver. The TinyOS packet format contains a group field to prevent different sensor networks from interfering with each other. It can be thought of as a kind of weak access control mechanism for non-malicious environments. The Ctr field is used for specifying the packet numbers. Our implementation needs one byte more than the TinySec packet format.

5.5. Security Analysis

Although our algorithm requires more memory and energy, it defends the sensor nodes against potential directed diffusion routing attacks. With the introduction of LEAP, the outsider attack has been completely eliminated. The LEAP algorithm prevents the sinkhole and wormhole attacks. The proposed solution will provide more security to the network even if some of the nodes in the network are compromised.

6. Simulations

A simulatios program in C++ code was written to simulate sensor networks of sizes 10, 30 and 50. The original directed diffusion algorithm does not use any keying mechanism therefore the memory required in each node is minimal. In our algorithm each node has to store one individual key, one group key and d number of pair-wise keys, where d is the is the number of neighbors of a node. The following graph was plotted for the storage requirement for the sensor network of size 10, 30 and 50. The nodes were placed at random location and the simulation was run for 30 times each for 10, 30 and 50 nodes. From figure 8, we can infer that the memory require to store the keys increases with the increase in the number of nodes. Each node also has to store the RSSI value and remaining battery power of one-hop and two-hop neighbors.

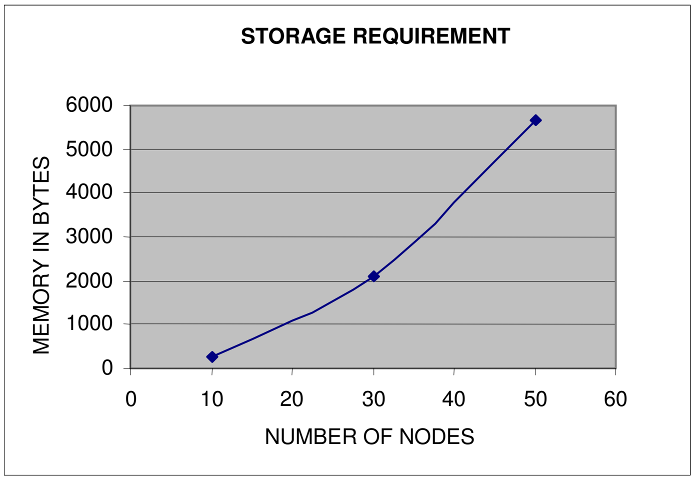

Figure 8. Memory Requirements

In our proposed algorithm each node has to find and store the ids of all the neighbor nodes before communicating with them. The initial set-up of the sensor network is different in our algorithm than the setup in the original directed diffusion. In directed diffusion, in order to establish the pair-wise key with all its neighbors, each node has to send a “hello” packet. Our approach includes an extra communication overhead in the form of a “hello” packet sent by the base station to all the nodes in the network so that the nodes in the network can calculate and store the base station’s RSSI value. The following graph shows the communication overhead caused by our algorithm and the original communication cost of the directed diffusion algorithm.

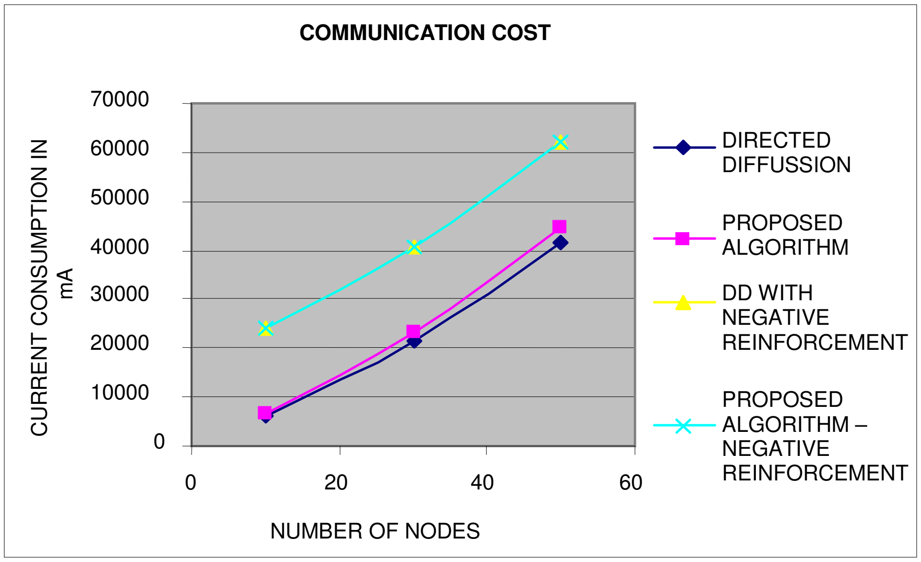

Figure 9. Power Consumption

From figure 9 we can see that the difference between directed diffusion and our proposed algorithm increases with the increase in the network size. The above graph also shows the overhead caused by our proposed solution for the negative reinforcement. Even if there is an authentic negative reinforcement, there will be communication and computation overheads in our proposed algorithm. If a node sends a negative reinforcement to an upstream neighbor, then it has to send the negative reinforcement information to all its neighbors, which is not done in the directed diffusion algorithm. As we can see from the above graph, the increase in the network size and the density causes more transmission and reception in the network, in other words, the overhead increases with the network size.

Our solutions to all the four type of attacks require some computational power. Our proposed solution to the selective forwarding algorithm causes more overhead, since all the packets forwarded in the high data flow path have to be compared by all other nodes. The solutions to other three types of attacks require the nodes to compare only when the need arises whereas in case of selective forwarding, the nodes have to compare all the time the interest is active. Besides the comparison overheads, the other computational energy required is for the encryption and decryption of the messages transmitted in the network.

When the nodes in the high data flow path sends the data to the base station, the low data flow path nodes have to compare the data forwarded by the nodes in the high data flow nodes. The computation overhead for implementing selective forwarding depends upon the number of nodes in the low data flow path. The following graph shows that the power consumption increases with the number of nodes.

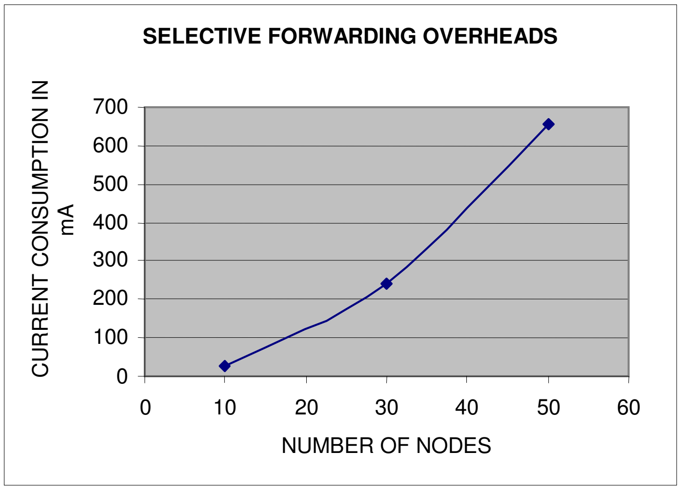

Figure 10. Selective forwarding overhead

When the base station or any node in the network sends a negative reinforcement, the upstream node which receives this negative reinforcement, processes only after comparing the same information from all its neighbors. The node which received the negative reinforcement information will forward the negative reinforcement to the upstream node only after receiving the negative reinforcement information from at least from two neighbors. The computation overhead caused by this type of attack is minimal and moreover this overhead occurs only when there is a negative reinforcement passed by some nodes in the network. The negative reinforcement is not a frequent occurence in a sensor network.

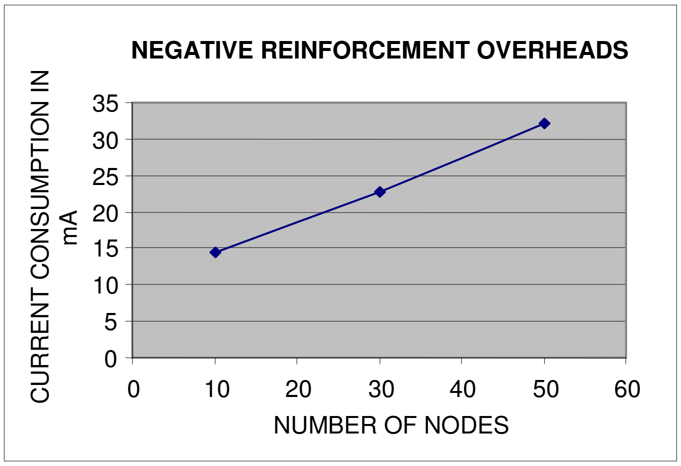

Figure 11. Negative Reinforcement overhead

7. Conclusions

In this paper, we have proposed a secure routing mechanism for sensor networks based on directed diffusion. Our simulation results show that the storage space required increases with the increase in the number of nodes in the network. When compared to the LEAP algorithm, our algorithm requires less memory space since our algorithm uses only three types of keys. In the case of negative reinforcement, our proposed algorithm differs little from directed diffusion. The density of the network determines the storage space required for each node; if there are more neighbors for a node then the node has to store more keys, thus increasing the memory requirement. Each packet contains the information about remaining battery power information and the RSSI value of a node. Therefore although there are overheads associated with our approach, including larger packet size, they are minimal while providing security. Our proposed algorithm can be used for applications which require message authentication and message confidentiality. We have assumed that the network is static; the proposed approach needs to be improved to handle mobile nodes. In directed diffusion a node transmits the interest information even to the node which originally sent it. This work can be extended by analyzing the duplicate interest problem and providing a solution. The packet size is also an important factor in determining the efficiency of our algorithm. Further work can be done to reduce the size of the packets.

References

[1] Karlof, C. and D. Wagner (2003). "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures." *AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols* Volume: 1**,** Issue 1, Page: 293--315.

[2] Asada, G., T. Dong, et al. (1998). "Wireless integrated network sensors: Low power systems on a chip." *Proc. 24th IEEE European Solid-State Circuits Conference* Page: 9-18.

[3] Hill, J., R. Szewczyk, et al. (2000). "System Architecture Directions for Networked Sensors." Proceedings of the ninth international conference on Architectural support for programming languages and operating systems Page: 93-104.

[4] D. Braginsky and D. Estrin, "Rumor Routing Algorithm for Sensor Networks," in the Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA, October 2002.

[5] Intanagonwiwat, C., R. Govindan, et al. (2003). "Directed Diffusion for Wireless Sensor Networking." Proceedings of the 1st ACM international workshop on wireless sensor networks and applications, Page: 2-16.

[6] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in the Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01), Rome, Italy, July 2001.

[7] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless sensor networks," in the Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00), Boston, MA, August 2000.

[8] TinyOS, http://sourceforge.net/projects/tinyos/ (last accessed March 31, 2006)

[9] Karlof, C., N. Sastry, et al. (2004). "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks." Proceedings of the 2nd ACM International workshop on wireless sensor networks and applications, Page: 22-29.

[10] Perrig, A., R. Szewczyk, et al. (2001). "SPINS: Security Protocols for Sensor Networks." Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001, Page: 189-199.

[11] Zhu, S., S. Setia, et al. (2003). "LEAP: efficient security mechanisms for large-scale distributed sensor networks." Proceedings of the 2nd ACM International workshop on wireless sensor networks and applications, Page: 62-72.

[12] Zhang, T., S. Pande, et al. (2002). "Tamper Resistance a Cautionary Note." Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems Volume 6, Issue 3, Page: 209-219.

[13] Hightower, J., R. Want, et al. (2000). "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength." Proceedings of Sixth Annual International Conference on Mobile Computing and Network, Page: 1-13.

[14] Pethe, A. G., G. Krishnakumar, et al. (2004). "Rank Based Localization Protocol for Sensor Networks." Proceedings of 8th Annual International Conference on Mobile Computing and Network, Page: 1-18.

[15] Crossbow Technology www.xbow.com

[16] Welsh, V.S. (2004), "Simulating the power consumption of large-scale sensor network applications". Proceedings of the 2nd international conference on Embedded networked sensor systems, Page: 188-200.

[17] TinyOS Documentation http://tinyos.net/tinyos-1.x/doc/ (last accessed March 31, 2006)