Key Distribution and Route Selection in Wireless Sensor Networks

Nathan Lewis
Information Science Institute
University of Otago PO Box 56
Dunedin New Zealand
ndlewis@infoscience.otago.ac.nz

Noria Foukia
Information Science Institute
University of Otago PO Box 56
Dunedin New Zealand
nfoukia@infoscience.otago.ac.nz

ABSTRACT
This paper presents our approach of a dynamic symmetric key distribution for encrypting the communication between two nodes in a Wireless Sensor Network (WSN). The distribution of a shared key can be performed by any sensor node and does not always require that it is performed by the base station (BS). Each node can be selected by one of its neighbor nodes in order to distribute a pair-wise key for a communication between two nodes. The selection is based on the local computation of a trust value granted by the requesting nodes. This paper also describes a dynamic route selection mechanisms based on trust and cost, that each node performs to route data to neighbor nodes and to the BS and details the route testing mechanism that we added to the trust-based routing protocol to avoid loops.

Categories and Subject Descriptors

General Terms

Keywords
Key Distribution, Trust, Wireless Sensor Network, Route Selection.

1. INTRODUCTION
A Wireless Sensor Network (WSN) consists of spatially distributed autonomous nodes called sensors that monitor physical or environmental conditions, such as temperature or pressure at different locations [1]. They are used in a variety of applications, such as climate sensing and control in office buildings. The privacy and security issues posed by WSNs are currently crucial issues of WSN research [2]. There are different ways of compromising sensitive information in WSNs. For example, an attacker can capture traffic from individual sensor nodes, reprogram nodes, or introduce her own sensor nodes in the network to be accepted as legitimate nodes. Because of the physical characteristics of a sensor network that enables large data collection [2] and makes the sensor information available from remote access, an attacker does not require to be physically located at a sensor node, thereby aggravating the security problem. To minimize this security issue, this paper proposes an efficient dynamic distribution mechanism of pair-wise keys based on the notion of local trust. Any node in the WSN can be selected by one of its neighbor nodes in order to distribute a pair-wise key for encrypting the communication between two nodes. Establishing a trust context will ensure that only trusted nodes within the WSN can share sensed information and more important, that current communication is reliable: it allows the detection of possible intruders or weak nodes under attack. Moreover, this paper also describes a dynamic route selection mechanism triggered by changes in trust (changes may be coming from outside our protocol, such as when a malicious node is detected) that each node performs to route data to neighbor nodes (that it shares a key with) and to the BS.

The structure of the paper is as follows: Section 2 describes our shared-key distribution protocol and explains how our approach is different from existing approaches of pair-wise key distribution performed by the BS. Section 3 explains the dynamic trust-routing protocol that we propose. Section 4 explains the route testing mechanism that has been added to the routing protocol to avoid creation of loops, and finally in Section 5, we provide the conclusion.

2. THE SHARED-KEY DISTRIBUTION PROTOCOL

2.1 Neighbor-based shared-key distribution

There are many ways of defining trust [7][8]. We define the notion of two “trusted” nodes as two nodes that share a private unique symmetric key and as far as each of them are aware, both nodes have not been compromised. Trusted nodes can be immediate neighbors in the radio range of the requesting node or multi-hop nodes not in the radio range. There are already several papers about using the BS to create keys that pairs of nodes share to talk to each other [3][4][6]. In all of these works, the basic idea is that each node has a unique symmetric key with the BS and every time that a node A wants to talk to node B, it asks the BS for some information about node B. For example, the BS may return a message confirming that node B is trusted and proof that node A can show to node B so that node B knows that it can trust node A. Kerberos is one such protocol [5].

From our reading of the different works dealing with pair-wise key distributions in sensor networks, it has been observed that they all require the nodes to communicate with the BS. This means that in terms of communication cost, if each node of the sensor network has on average d radio range neighbors (d is the average degree of nodes in the sensor network) and there are N nodes in the sensor network, then the cost to set up all pair-wise keys between all neighboring nodes will be N.d communications.
between the BS and the nodes. This creates a large amount of network traffic which is not desirable in a WSN. An argument that differentiates our approach is that nodes do not have to check with the BS every time they set up a pair-wise key with a new neighbor, if they can get the key directly from one of their already trusted neighbors. Consider the case where the BS already trusts A, B and C. A and B are allocated a private unique pair-wise symmetric key (K_{AB}) by communicating with the BS, as are A and C (K_{AC}). Instead of asking the BS to provide a key when B and C want to communicate, they can be allocated their new pair-wise symmetric key by communicating with A, without the need to communicate with the BS.

The way it works is as follows: Node B wants to communicate with node C. B sends a message to all of its trusted immediate (in the radio range of the requesting node) neighbors asking them if they trust node C. If a trusted neighbor A replies that it does, B then asks A to allocate them a new pair-wise symmetric key K_{BC}, as described above with the direct communication with the BS. If no neighbors reply, B has no option but to ask the BS (perhaps through multiple hops) to allocate the key.

Initially, each node needs to establish a pair-wise key with one other neighbor via the BS. Later, a pair of nodes can establish a pair-wise key via any trusted neighbor that they have in common. Moreover, considering the physical distribution of nodes, most nodes are far away from the BS. Rather than having the majority of nodes using multi-hop communications to set up their first local trust relationship and then all further trust relationships are established with single-hop communications (immediate neighbor). In the previous scheme, N.d communications with the BS were required to establish pair-wise keys between all immediate neighbors. Theoretically, in our new scheme, the cost of communications should be reduced by the factor d, as each node only has to communicate with the BS only once.

2.1.2 Calculating the initial trust
Suppose that we have 3 sensor nodes A, B and C. Node A is the allocator of the key between B and C and has trust values TN_{A,B}=0 and TN_{A,C}=0. We need to calculate the initial trust between B and C. B has a trust value TN_{B,A}. C has a trust value TN_{C,A}. B can calculate the new trust value TN_{B,C} using TN_{A,B} and TN_{A,C} using some local function. In our approach we simply multiply the values together, i.e. TN_{B,C} = TN_{B,A} * TN_{A,C}. Multiply works well for limiting cases, resulting in TN_{B,C}=0 if either TN_{B,A}=0 or TN_{A,C}=0 and resulting in TN_{B,C}=1 if TN_{B,A}=TN_{A,C}=1 (0.0 represents no trust and 1.0 means full trust). It also results in sensible initial trust values with other values for TN_{B,A} and TN_{A,C}.

2.1.3 Protocol description
At the beginning of the shared-key distribution protocol, each node sends a Hello message. Nodes that hear a Hello message record the address of the sending node in their Neighbor table. A BS that receives a Hello message will immediately send a Key message to that node. This Key message informs the node that the BS is within radio range. Once any node has a route to a BS, it begins to look through its list of neighbors for any neighbor that it does not yet have a shared pair-wise key. In a standard Kerberos-based [5] system, the node would then ask the BS for a pair-wise key that it can use to securely communicate with that neighbor (the “target”). Here instead, the requester node will send out an Allocator Request message. If any immediate neighbor shares pair-wise keys with both the requester and the target then that neighbor (the “allocator”) can reply with an Allocator Reply message. If no neighbor replies to the Allocator Request message before a timer has expired then the requester has no choice other than to send a Key Request to its BS (note that each node must have at least one key allocated to it via a BS before it can begin requesting keys with other neighbors). However if an Allocator Reply message is received then the requester can send a Key Request message directly to allocator. When a node or BS receives a Key Request message, it generates a new random key and sends it to the requester in a Key Reply message. The requester receives and records the new pair-wise key and creates a Key message that it sends to the target, including a copy of the token that was in the Key Reply message. The Key message is received by the target and it can use the token to verify that the new pair-wise key came from the trusted allocator.

2.2 Simulation, results and analysis
The neighbor-based shared-key distribution has been simulated using the OMNet++ simulator [9] in C++. To measure the efficiency of the algorithm, simulations were run to record the total number of packets sent by all nodes. These include Hello messages, Allocator Request and Reply messages (of which there are none when the BS performs all key allocations), Key Request and Key Reply messages (counted once for each hop from the source to the destination) and Key messages.

We compared the neighbor-based shared-key distribution to distribution of pair-wise keys performed only by the BS. In our simulations, we had one BS in the center of a square region which contained randomly scattered nodes. The size of the area was increased in proportion to the number of nodes to keep a consistent average node degree. The experiments were repeated with average node degrees of 10 and 15. Each simulation ended when all nodes had acquired pair-wise keys with every one of their radio-range neighbors. The results are the averages of each of the 100 iterations (Figure 1). Standard deviations were in the range 5.9 to 8.5 for simulations with 100 nodes and between 3.1 and 5.8 for simulations with 200 to 500 nodes.

![Figure 1: Number of transmissions/node required for all nodes to share a key with their immediate neighbors.](image-url)
The objective of using the neighbor-based shared-key distribution scheme was to reduce the number of transmissions required while forwarding additional Key Request and Key Reply messages to and from the BS. The cost of this is the number of transmissions required to find out if an immediate neighbor would be able to perform the key allocation, rather than having to ask a BS. We found that when there are a small number of nodes, the overhead of Allocator Request and Allocator Reply packets exceeded the savings from reduced Key Request and Key Reply forwarding. Nodes that are only a few hops away from the BS send route transmissions with the proposed scheme than with the simple BS distribution scheme. But the proposed scheme scales well with increasing network size. As the number of nodes increases, the proportion of nodes that are far away from the BS increases. Beyond a certain network size, our method may be able to give a substantial performance increase which is appropriate for many application of WSNs. The threshold and savings depend on the average node degree. From these results, we can see that the lower the degree, the smaller the network size needs to be before we begin to see the benefits of using our algorithm; around 125 nodes when d=10 and around 300 nodes when d=15). Also, the rate of increase in number of transmissions per node slows down as the number of nodes increases when using our algorithm.

3. DYNAMIC TRUST-BASED ROUTING PROTOCOL

3.1 Routing behavior

In a simple WSN, data is routed from the nodes to the BS and global maintenance messages are flooded from a BS to the nodes. When a node sends a request for information to the BS a route must be maintained to allow the reply to be sent back. In more complex situations a node may wish to send a message to a specific node, perhaps for data aggregation.

We expect our algorithm to exhibit the following properties:

- If a route will not succeed (there is some node on the route that has an attributed trust value of 0.0) then nodes will not use that route (trust value is between 0.0 and 1.0, 0.0 represents no trust and 1.0 means full trust).
- The algorithm should avoid loops when a node is selecting a new route.
- The algorithm should minimize the overhead of route and trust information that nodes are transmitting to maintain trusted routes to the BS.

3.2 Computing the trust and cost of a route

Node A use Formulas (1) and (2) to computes the trust $TR_{A \rightarrow B}$ and cost $CR_{A \rightarrow B}$ of a route through node B to BS (although the destination may also be any node other than the BS) by combining the trust $TN_{A \rightarrow B}$ (trust that node A grants to node B) and $CN_{A \rightarrow B}$ (cost to transmit directly to B) that it has for node B with the trust $TR_{B \rightarrow BS}$ and cost $CR_{B \rightarrow BS}$ that node B has broadcast (see Figure 2).

$$TR_{A \rightarrow B} = \min(TN_{A \rightarrow B}, TR_{B \rightarrow BS})$$

$$CR_{A \rightarrow B} = CN_{A \rightarrow B} + CR_{B \rightarrow BS}$$

3.2.1 Choosing between two routes

A node chooses a route according to the cost of a route as well as its trust of the route. Cost can have any value $\geq 0$. For simplicity, the cost for transmission from one node to an immediate neighbor is 1 in our simulations, but any metric may be used, such as latency or the radio power required to reach the neighbor. A node wishes to minimize the cost while maximizing the trust of the route it chooses to use.

![Figure 2. Trust and cost parameters to compute the trust of routes.](image)

Suppose that node A is given the choice between two routes (to the BS or any other node). For simplicity we call these routes $R_B$ and $R_C$ with trust values $TR_B$ and $TR_C$ and costs $CR_B$ and $CR_C$ respectively; there are five cases to consider:

- In the extreme case of $TR_B = TR_C$ and $CR_B = CR_C$ then the previously selected route should be maintained.
- If $TR_B = TR_C$ then choose the cheaper route.
- If $CR_B = CR_C$ then choose the more trusted route.
- If one route is more trusted and cheaper than the other route then choose that route.
- If $TR_B > TR_C$ and $CR_B > CR_C$ then the node must have some way of deciding if it is more important to find a more reliable route or a cheaper one. The choice should depend on both the cost and the trust.

According to Equation (3), we propose to take the trust of two routes and compare the cost of trying and failing with one route vs. trying and failing with the other. Let $E_B$ be the Expenditure of choosing $R_B$. Then $E_B$ is calculated as:

$$E_B = CR_B + (1-TR_B)CR_B$$

We assume that $(1-TR_B)$ is proportional to the probability that using route $R_B$ fails. Similarly,

$$E_C = CR_C + (1-TR_C)CR_B$$

Comparing $E_B$ to $E_C$, the route with the smaller Expenditure is chosen by A.

We can test the formula and see that we get the same result as earlier:

- If $TR_B = TR_C$ then the cheapest will always have a smaller Expenditure.
- If $CR_B = CR_C$ then the most trusted will always have a smaller Expenditure.
- If one route is more trusted and cheaper than the other route then the Expenditure will always be smaller for this route.
3.2.2 Choosing between more than two routes

We can compare all "pairs" of available routes using the method described above. There is a potential problem: what if node A is given the choice between three routes $R_5$, $R_c$, and $R_9$ where $E_5 < E_c < E_9$ and $E_5 < E_9$? In this situation there is no overall best route. We have tested this empirically and found that for all possible values of $TR_{bc}$, $TR_c$, $TR$ of $CR_c$, and $CR_9$ there is always one route that has a smaller Expenditure when compared with the other two routes, except in trivial cases such as when two or more routes have the same TR and CR or when $CR_{bc} = CR_c = CR_9 = 0$.

In order to compare all the routes to find the best one, a node does not need to compare all pairs. It only needs to compare two routes and discard the route with the higher Expenditure until it has eliminated all but one route, which is its best choice.

4. ROUTE TESTING TO AVOID LOOPS

In our simulations, when sending route update information it would be impractical to verify that all the nodes that can use the information have successfully received it. For this reason, nodes may have an inconsistent picture of the state of the network.

When the cost of using a route changes, the affected node chooses a new best route and broadcasts to all immediate neighbors the new cost and trust information that it is using. It is possible that a neighbor fails to receive the Route Update message. Consider the situation in which node A must choose a new route to the BS because its current route has failed. It may decide that the route through node B is the best choice, even though the route through B has a higher cost than the previous route that node A was using. Node A then broadcasts that it has made a new route to the BS (other nodes may route through node A) with the new trust and cost values. Node B may fail to receive the Route Update message. In this case, if node B is later forced to choose a new route to the BS, it may choose to send data through node A, thinking that the route through node A costs less than the actual cost that node A most recently broadcast. This creates a loop where data sent to the BS through (or by) nodes A or B will loop between the two nodes.

To prevent loops caused by inconsistent routing information, we had nodes send a Route Test message along a new route before using it. When a node sends a Route Test message to the destination of a route there are three possible outcomes:

1) The Route Test message is forwarded all the way to the destination of the route, at which time a Route Confirm message is returned by the destination node. When a Route Confirm message arrives at the initial node, the node knows that the route is reliable and void of loops. If a Route Confirm message is received then the node can start using that route and transmit a Route Update message to all neighbors.

2) The Route Test message is lost, in which case the node will time-out waiting for the reply, label the route as unreliable and chooses another route from those it has available.

3) The Route Test message ends up returning (possibly via multiple forwardings) to the initial node. This can only occur if the route contained a loop, in which case the node labels the route a loop (excluded) and chooses another route from those it has available.

For example, in Figure 3 node A has two available routes to the BS. If it tests the route through node C then the Route Test message will end up back at node A, at which point it knows that the route through node C will produce a loop, whereas if node A tests the route through node B then the BS will return a Route Confirm message.

![Figure 3. Testing Loops](image)

5. CONCLUSION

In this paper, we presented a key distribution method for encrypting the communication between two nodes in a Wireless Sensor Network (WSN). This method is based on trust to decide which neighbor node allocates keys.

This paper also described in more detail, a dynamic route selection mechanisms based on trust and cost, that each node performs to route data to neighbor nodes and to the BS and details the route testing mechanism that we added to the trust-based routing protocol to avoid loops.

6. REFERENCES


