An Efficient Scheme for Secure Data Transmission in Wireless Sensor Networks

Al-Sakib Khan Pathan and Choong Seon Hong

Department of Computer Engineering, Kyung Hee University

spathan@networking.khu.ac.kr and cshong@khu.ac.kr

Abstract

In this paper, we propose an efficient security scheme for wireless sensor networks which provides good level of confidentiality and authenticity of the data that are transmitted from the sensors to the base station. We use one-way hash chain and pre-stored shared secret keys for ensuring the data transmission security in the network. In our scheme, we first create a tree structure in the network and then use it for authenticated and encrypted data delivery. To introduce data freshness, our scheme includes an optional key refreshment mechanism which could be applied depending upon the requirement or the application at hand. We present a brief analysis along with the detailed description of our scheme.

1. Introduction

Wireless Sensor Networks (WSN) are emerging as both an important new tier in the IT ecosystem and a rich domain of active research involving hardware and system design, networking, distributed algorithms, programming models, data management, security and social factors [1], [2], [3]. It is anticipated that in most application domains, sensor networks constitute an information source that is a mission critical system component and thus, require commensurate security protection. If an adversary can thwart the work of the network by perturbing the information produced, stopping production, or pilfering information, then the usefulness of sensor networks is drastically curtailed. Thus, it should be made sure that, the reports from the '*sensors in action*' are authentic and reach the base station (BS) without any fabrication or modification. The task of securing wireless sensor networks is however, complicated by the fact that the sensors are mass-produced anonymous devices with a severely limited energy and memory budgets, and initially with no knowledge of their locations in the deployment environment (in most of the cases).

In this paper, we basically deal with the challenge of secure data transmission in wireless sensor networks in a highly dense deployment scenario. We propose a scheme which aims at securing the data transmissions from the sensors to the base station using one-way hash chain (OHC) and pre-stored shared secret keys. Our proposed scheme provides a good level of confidentiality and authenticity of the reports sent from the source sensors to the base station.

The rest of the paper is organized as follows: Section 2 states the related works, section 3 presents our network assumptions, Section 4 describes our scheme in detail, analysis of our scheme is presented in section 5, and section 6 concludes the paper delineating the achievements from this work with future research directions.

2. Related Works

Ye et al. [4] proposed a statistical en-route filtering scheme in which a report is forwarded only if it contains the message authentication codes (MACs) generated by multiple nodes, by using keys from different partitions in a global key pool. Zhu et at. [5] proposed the interleaved hop-by-hop authentication scheme that detects false reports through interleaved authentication. Lee and Cho [6] proposed an enhanced interleaved authentication scheme called the key inheritance-based filtering that prevents forwarding of false reports. The keys of each node used in the message authentication consist of its own key and the keys inherited from its upstream nodes. Every authenticated report contains the combination of the message authentication codes generated by using the keys of the consecutive nodes in a path from the base station to a terminal node.

Our proposed scheme is different from the above mentioned schemes as we create a tree-structure in the network to use OHC for secure data transmission. The OHC ensures the authenticity of the data sent from the sensors to the base station and the confidentiality of the data is ensured with the shared secret keys of the sensors.

3. Network Assumptions and Preliminaries

This work was supported by the MIC and ITRC projects. Dr. C. S. Hong is the corresponding author.

We assume that, initially all the nodes and base station in the network have same transmission ranges and all of their clocks are synchronized. The base station (BS) has enough energy to support the network's operations for its full lifetime. Also the base station cannot be compromised by any adversary. All the sensor nodes are relatively static in their respective positions after deployment. The link between any pair of nodes in the network is bidirectional, that is, if a node n_i gets a node n_j within its transmission range (i.e. one hop), n_j also gets n_i as its one-hop neighbor. Each node transmits within its transmission range isotropically (in all directions) so that each message sent is a local broadcast. The BS contains all the shared secret keys and corresponding ids of the sensors. Each shared secret key is pre-stored in the sensor before deployment. We assume that, no node could be compromised by any adversary while creating the tree structure in the network (i.e., the first phase of our scheme which is described later in section 4.1). To ensure security for data transmission, we use pre-stored shared secret keys and one-way hash chain. Our target is to ensure confidentiality and authentication of the reports that travel from the active sensors to the base station.

One-way hash chain (OHC) is basically used to provide authenticity of data in the network. An OHC [7] is a sequence of numbers generated by one-way function F that has the property that for a given x, it is easy to compute $y = F(x)$. However, given F and y, it is computationally infeasible to determine x, such that, $x = F^{-1}(y)$. An one-way hash chain (OHC) is a sequence of numbers $K_n, K_{n-1}, \ldots, K_0$, such that, $\forall_i : 0 \le i < n$, $K_i = F(K_{i+1})$. To generate an OHC, first a random number K_r is selected as the seed, and then F is applied successively on K_r to generate other numbers in the sequence. In the next section, we describe in detail how the shared secret keys are used with OHC in our scheme to provide data transmission security.

4. Our OHC-based Security Scheme

We describe our security scheme with two interrelated phases. The first phase is used for initializing the one-way hash chain number in the network. We create a secure path from the base station to the source nodes (any sensor in the network). Along the path, the OHC plays the major role to provide authenticity of the reported data. In the second phase, we use the OHC numbers to detect and filter out false data injected by the potential adversaries.

4.1. Initializing OHC Number in the Network

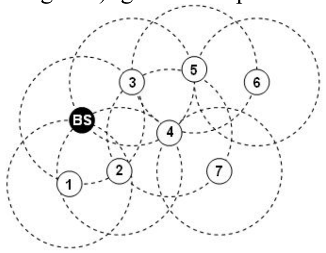
According to our assumption, all the sensors and the base station have shared secret keys that are pre-stored before deployment of the network. So, when the sensors are deployed in the target area randomly, each sensor contains a shared secret key with the base station. To provide data transmission authenticity, all the intermediate nodes between any source and the base station must be initialized with the basic one-way hash chain number. Let us suppose the initial OHC number is HS_0. To bootstrap the OHC number, the base station first generates a control packet containing HS_0 and a MAC (Message Authentication Code, noted here as MAC_{K_i}) for the control packet using a key K_i, where K_i is the number in the key chain number corresponding to time slot t_i. The format of the initial control packet generated by the base station (or sink) B is:

$$bcm:\ B|HS_0|MAC_{K_i}(B|HS_0)$$

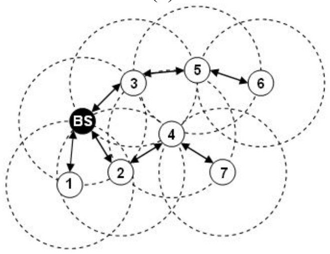
Here, B is the id (indicates that this message is a control message sent from the base station) of the base station. The initialization message is first received by the one-hop neighbors of the base station. Receiving the message, each node in the one-hop neighborhood stores the value of HS_0 and sets the BS as its forwarder node (as the ultimate destination is the base station). Now, each of these nodes transmits the message within its own one-hop neighborhood (i.e., local broadcast) including its own id. The format of this message is:

$$ncm:\ sid|fid|B|HS_0|MAC_{K_i}(B|HS_0)$$

where, *sid* is its own id and *fid* is the id of its selected forwarder node. As stated earlier, for the one-hop neighbors of the base station, this *fid* is set to *B*. These nodes basically do not alter the main control message rather, they just add two more parameters with it to pass it down the whole network. Any other node that has already got the control message from the base station (i.e., any other one-hop neighbor) ignores this packet.



(a)



(b)

Figure 1. An example scenario of OHC initialization phase (a) Network in the initial state (b) After initialization phase is over. Here, we have three paths, $B \rightarrow 3 \rightarrow 5 \rightarrow 6$, $B \rightarrow 2 \rightarrow 4 \rightarrow 7$ and $B \rightarrow 1$ along which the one-way hash chain number is initialized. The dashed circles indicate the transmission ranges of the nodes

Now, any node that has not received the message earlier (i.e., two hops away from the base station) receives it and stores the initial OHC number, HS_0. It then sets the id of the sender node as its forwarder node and again locally broadcasts the control message with its own id as *sid*. In case, it has got the control packet from two or more sender nodes, it picks up the message which it receives first and discards all other messages. However, this node stores the ids of the other senders. This knowledge is necessary to repair a broken path, which we will discuss later in this paper. When this node does the local broadcast with the modified *sid* and *fid*, the previous sender node eventually knows that it has been selected by its downstream node as a forwarder (as each link is bidirectional). So, the upstream node (in this case, the one-hop neighbor of the base station) sets itself as a forwarder for this node. This process continues and eventually a tree-structure is created in the network where, each node has a forwarder node on the way to reach to the base station and a downstream node that can send data to it destined to the base station. To authenticate HS_0, *B* releases the key K_i in time slot t_{i+d}. On receiving this key, an intermediate node can verify the integrity and source authentication of HS_0. It is to be noted that, *bcm* won't bring any attack against the network even if the nodes on the other side of the network don't receive K_i at t_{i+d}. Since, the messages that are *MAC*ed by K_i are supposed to be sent out at time slot t, an adversary cannot launch any attacks with K_i when it gets K_i at t_{i+d}. Thus, along each path the initial OHC number is initialized.

Let us illustrate the initialization phase with an example. Figure 1 shows the example scenario. At the very beginning, the base station BS transmits the control message *bcm* with initial OHC number and MAC. Nodes 1, 2 and 3 in this case get the initial control message. All of these nodes set base station *B* as their immediate upstream forwarders and set the respective *fid*s. Node 4 is within the transmission ranges of both 2 and 3. So, when node 4 gets the message from two different sender nodes, it has to pick up one as the forwarder node. Say, node 4 has chosen 2 as its forwarder. When it transmits the local broadcast message using the control message *ncm*, node 2 knows that node 4 is its downstream node and sets itself as the forwarder of node 4. Now, when node 3 does the local broadcast, node 5 also gets the message and it could set node 3 as its own forwarder. When node 5 gets the message again from node 4 with node 4 as the *sid*, it simply ignores the message as it has already chosen its forwarder. Also, it could be noticed from the example that, node 1 and node 2 are the one-hop neighbors of the base station and they both get the control message from the same source (which is in this case the base station itself). So, when the local broadcasts of node 1 or node 2 reach each other, as previously stated they simply ignore the messages. This process continues until all the nodes in the network are included in the OHC initialization tree. All the nodes get the initial value of OHC number and the network becomes ready for data transmission phase after time slot t_{i+d}. We show the output network structure in Figure 1 (b) after executing our algorithm on the sample network shown in Figure 1(a).

4.2. Secure Data Transmission

After execution of the first phase, the whole network is structured as a tree where the root node is the base station and there are other non-leaf and leaf nodes. In this section, we describe how the sensors transmit their sensed reports securely to the base station.

To send the data securely to the sink, each source node n_s maintains a unique one-way hash chain, *HS:* $<HS_n, HS_{n-1}, \ldots, HS_1, HS_0>$. When a source node, n_s sends a report to the sink using the path created in the sink-rooted tree (for example, a path is $n_s \rightarrow \ldots \rightarrow n_{m-1} \rightarrow n_m \rightarrow B$), it encrypts the packet with its shared secret key with the sink (or base station), includes its own id and an OHC sequence number from *HS* in the packet. It attaches HS_1 for the first packet, HS_2 for the second packet, and so on. To validate an OHC number, each intermediate node $n_1,\ldots,n_m$ maintains a verifier I_S for each source node, n_s. Initially, I_s for a particular source node is set to HS_0. When n_s sends the *i*th packet, it includes HS_i with the packet. When any intermediate node n_k receives this packet, it verifies, if $I_s = F(HS_i)$. If so, n_k validates the packet, it forwards it to the next intermediate node, and sets I_s to HS_i. In general, n_k can choose to apply the verification test iteratively up to a fixed number *w* times, checking at each step whether, $I_s = F(F\ldots(F(HS_i)))$. If the packet is not validated after the verification process has been performed *w* times, n_k simply drops the packet. By performing the verification process *w* times, up to a sequence of *w* packet losses can be tolerated, where the value of *w* depends on the average packet loss rate of the network. Note that, an intermediate node need not to decrypt the packet rather it can optionally check the authenticity of the packet before forwarding to its immediate forwarder. Figure 2 illustrates the OHC utilization for secure data transmission from any source node to the base station.

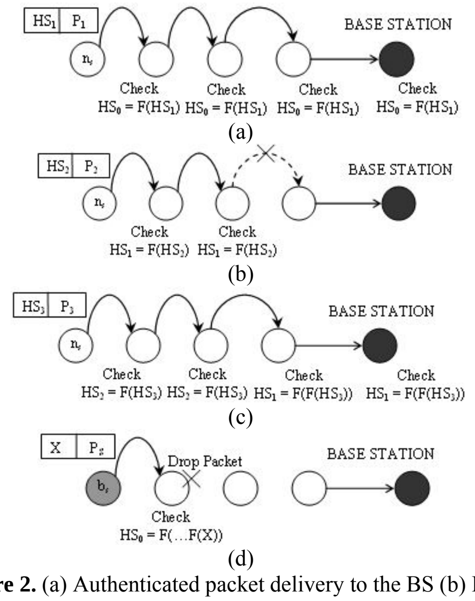


Figure 2. (a) Authenticated packet delivery to the BS (b) Packet could not reach BS (c) But it cannot affect the OHC verification (d) Bogus packet detection and drop by an intermediate node

In Figure 2(a), the source node n_s sends the first packet to the base station with the OHC value HS_1. The content of the packet is encrypted with the secret key that it shares with the base station. Getting the packet, the base station performs the authenticity check by verifying the hash chain number and gets the report by decrypting it with the shared key for that particular source node. Figure 2(b) shows a scenario where the packet P_2 could not reach the base station for some reason. In spite of that, the OHC verification is not hampered as for the next packet, the third intermediate node performs the hash verification twice (Figure 2(c)). Here, at the very first attempt it cannot get the value of HS_1 in the verification process but in the second iteration, it verifies it as a valid packet from the source n_s. In fact, in this case, the intermediate node can perform the hash number verification *w* times where, *w* is an application dependent parameter. In Figure 2(d) an adversary tries to send a bogus packet with a false hash chain number and it is detected in the next upstream node. Eventually such bogus packet fails to pass the authentication check and is dropped in the very next hop. This feature saves energy of the network as the falsely injected packets cannot travel through the network for more than one hop.

4.3. Optional Key Refreshment of the Sensors

To provide data freshness and to increase the level of security, our scheme has an optional key refreshment mechanism. In this case, the base station periodically broadcasts a new session key to the sensors in the network. The format for this message is: $B|K_s|\ MAC_{K_s}(B|K_s))$, where, K_s is the number in the key chain number corresponding to time slot t_s. To authenticate K_s like the OHC initialization phase, *B* releases the key K_s in time slot t_{s+d}. On receiving this key, the nodes can verify the integrity and source authentication of K_s. Then each node gets the new key by performing an X-OR (exclusive OR) operation with its old key. This method could also be utilized for refreshing the keys of a specific number of nodes. In that case, the base station could simply send the K_s to the specific node by encrypting it with the previous shared secret key. Upon receiving the new key, the node can perform the X-OR operation and could use the newly derived key for subsequent data transmissions.

Changing encryption keys time-to-time has an advantage as it guarantees data freshness in the network. Moreover, it helps to maintain confidentiality of the transmitted data by preventing the use of the same secret key at all the times.

4.4. Repairing a Broken Path and OHC Re-initialization

If in any case, any node between the source node and the base station fails, it could make one or more paths useless. Eventually, in such a case all the downstream nodes along that particular path get disconnected from the base station. To repair such a broken path, we use the stored upstream knowledge of the sensors. We know that, in the first phase each downstream node stores the ids of the one-hop upstream senders of the control message. So, this knowledge could be used for repairing the path quickly.

Let us illustrate it with an example. Say, in Figure 1(b), node 2 is somehow damaged or failed to continue. So, the nodes 4 and 7 get disconnected from the base station. This failure could first be detected by the one-hop neighbors of node 2 in the tree i.e., node 4 and node 1. In the first phase, as node 4 got a message from node 3 which tried to become its forwarder, node 4 could use that knowledge to repair the path. So, node 4 first does a local broadcast of an error message that it has lost its previous forwarder and sets node 3 as its forwarder. Accordingly, node 3 gets a forwarding status for node 4. If there were more senders who had sent control messages to node 4 at the time of tree construction, node 4 would have chosen any one of the nodes. We know that in the first phase, each node stores the information about its neighbors who try to become its forwarder. If node 4 is required to send any packet as a source node, it could simply send it using the OHC number in the sequence, HS_{k+1} which is next to its last used OHC number, HS_k. For node 3, node 4 is a new source, so it could save its HS value in I_4. The subsequent transmissions from node 4 are verified by node 3 based on this initial knowledge. There is another stranded node in our example, node 7. In this scenario it does not lose node 4 as forwarder, so it faces no problem. But, if it had been affected in any way, it would have followed the same procedure to repair the broken path. The structure of the new path after broken path recovery is shown in Figure 3.

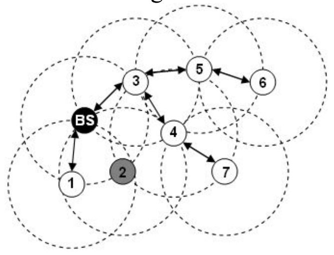


Figure 3. Repairing a broken path

As we are considering a highly dense deployment scenario, we believe that, in most of the cases, a node might initially get two or more upstream senders who would try to be its forwarder. This procedure works fine as long as no more than *w* packets are lost on the way, from any source node. If within the time of repairing the path, more than *w* packets are lost from a particular source, the OHC chain along that path breaks down. In fact, this is the worst case where all the downstream nodes along the path become invalid to the base station and their sent data are discarded on the way to reach the base station. To overcome this problem, the entire OHC initialization phase (the first phase of our protocol) could be made periodic (after certain interval, which is an application dependent parameter). Determining the best possible time interval for re-initialization of the first phase is kept as our future work.

5. Analysis of Our Scheme and Discussions

The method of generating and storing a long OHC in a sensor node is not straight forward. Naive algorithms require either too much memory to store every OHC number, or too much time to compute the next OHC number. None of these algorithms are practical on resource-constrained sensor nodes. Recently, some efficient OHC generation algorithms for resource-constrained platforms have been proposed [8], [9], [10].

Among these algorithms, the fractal graph traversal algorithm [8] could perform well on the traditional sensor nodes. This algorithm stores only some of the intermediate numbers, called pebbles, of an OHC, and uses them to compute other numbers. If the size of an OHC is *n* (there are total *n* numbers in this OHC), the algorithm performs approximately $\frac{1}{2}\log_2 n$ one-way function operations to compute the next OHC number, and requires a little more than $\log_2 n$ units of memory to save pebbles.

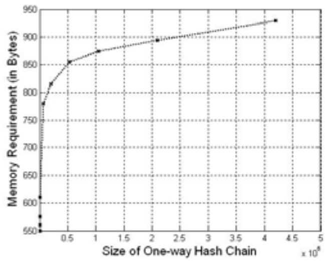


Figure 4. Memory requirement for One-way Hash Chain generation

The length of an OHC that is needed for a source node is also an important factor. The typical length is between 2^{11} to 2^{22}. If the length of an OHC is 2^{22} and a node uses one OHC number per second, it will take more than a month to exhaust all numbers from this chain. Figure 4 shows the storage requirements for storing pebbles for different lengths of an OHC. This includes a skipjack based one-way function and OHC generation based on [8]. We see that a node needs about 930 bytes to maintain an OHC of length 2^{22}. This includes 256 bytes lookup table for skipjack, which can be shared with other applications. Other than this, each node has to store only a few ids of the upstream sender nodes. Overall, the memory requirement could be well met with today's sensor nodes.

In our scheme we mainly ensure the data authenticity and confidentiality. It is evident from our scheme description that, any falsely injected data could not travel more than even a single hop. Any adversary trying to inject false reports must generate valid OHC number to forward the data towards the base station. If any shared key of any node is exposed to any adversary, it makes no sense to it as it needs valid OHC number to forward the packet. Similarly, getting the OHC only is not useful for sending bogus reports. In the worst case, if a node is compromised fully, the node could be used for sending bogus reports. However, we assume that, there is an intrusion detection system in the network which runs side-by-side to detect the abnormal behavior of a legitimate node. The details of the intrusion detection system is out of the scope of this paper. For energy efficient transmission of the control messages a scheme like [11] could be combined with our scheme which we have kept as our future work.

6. Conclusions and Future Works

In this paper, we have proposed an efficient security scheme for providing data transmission security in wireless sensor networks. Our scheme takes advantage of one-way hash key chain numbers and pre-stored shared secret keys. There is an optional key refreshing mechanism which could be used to increase the level of security and to ensure data freshness. In this paper, we have focused basically on providing security during data transmission from the source sensor nodes to the base station. However, there is a lot of scope for further research in this area. As our future works, we will investigate the energy-efficiency of our scheme in detail. Also we are working on finding out an optimal value of interval for re-initiating the first phase of the scheme so that, the maximum lifetime of the network could be ensured along with data transmission security. Finally, it should be mentioned that, for the lack of space, we have shortened some of the parts in this paper.

7. References

[1] D. E. Culler, and W. Hong, "Wireless Sensor Networks", *Communication of the ACM*, Vol. 47, No. 6, pp. 30-33, June 2004.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey", *Computer Networks*, 38, pp. 393-422, 2002.

[3] S. Dai, X. Jing, and L. Li, "Research and analysis on routing protocols for wireless sensor networks", *Proc. International Conference on Communications, Circuits and Systems*, Volume 1, pp. 407-411, 27-30 May, 2005.

[4] F. Ye, H. Luo, S. Lu, "Statistical En-Route Filtering of Injected False Data in Sensor Networks", *IEEE J. Sel. Area Comm.*, 23(4), pp. 839-850, 2005.

[5] S. Zhu, S. Setia, S. Jajodia, P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks", In *Proc. of S&P*, pp. 259-271, 2004.

[6] H. Y. Lee, and T. H. Cho, "Key Inheritance-Based False Data Filtering Scheme in Wireless Sensor Networks", *LNCS 4317*, Spr.-Ver., pp. 116-127, 2006.

[7] L. Lamport, *Constructing digital signatures from one-way function*. In technical report SRI-CSL-98, SRI International, October 1979.

[8] D. Coppersmith and M. Jakobsson, "Almost Optimal Hash Sequence Traversal", *In 6th International Financial Cryptography 2002*, Bermuda, March 2002.

[9] M. Jakobsson, "Fractal hash sequence representation and traversal", *In 2002 IEEE International Symposium on Information Theory*, Switzerland, July 2002.

[10] Y. Sella, "On the computation-storage trade-offs of hash chain traversal", *In 7th International Financial Cryptography Conference*, Guadeloupe, January 2003.

[11] Mamun-Or-Rashid, M. M. Alam, and C. S. Hong, "Wasteful Energy Conserving Maximum Lifetime Routing for Wireless Sensor Network", *ICOIN 2006*, pp.15-19, Sendai, Japan, January 2006.