Securing Sensor Reports in Wireless Sensor Networks*

Al-Sakib Khan Pathan and Choong Seon Hong

Department of Computer Engineering, Kyung Hee University
Giheung, Yongin, Gyeonggi, 449-701 Korea
spathan@networking.khu.ac.kr, cshong@khu.ac.kr

Abstract. The sensor reports from a wireless sensor network are often used extensively in the decision making process in many systems and applications. Hence, classifying real and false sensor reports is necessary to avoid unwanted results. In this paper, we propose a scheme for securing the sensor reports in a wireless sensor network. We use one-way hash chain and pre-stored shared secret keys to provide data transmission security for the reports that travel from any source node to the base station. 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 an analysis along with the detailed description of our scheme.

1 Introduction

Wireless sensor networks (WSNs) have promised to provide a great opportunity of gathering specific types of data from specific geographic areas. WSNs can successfully operate even in unattended, hostile or hazardous areas. While this aspect of WSN has made its use very lucrative in many military and public-oriented applications [1], [2], it has also raised a lot of questions and the issue of ensuring security in such types of networks has become a major challenge. It is anticipated that, in most application domains, wireless 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 is very crucial in many applications to make sure that, the reports sent 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 severely limited memory, power and communication resources. Also, in most of the cases, the sensors do not have any knowledge of their locations in the deployment environment. Though there are many security issues to address in WSNs, in this paper, we mainly focus on ensuring security for the sensor reports, which the active and legitimate sensors send to the base station (BS). In fact, a single security scheme cannot provide all sorts of security protections in such types of wireless networks.

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

A.-M. Kermarrec, L. Bougé, and T. Priol (Eds.): Euro-Par 2007, LNCS 4641, pp. 881–890, 2007.

Here, we consider a densely deployed scenario of a wireless sensor network. Our main goal is ensuring authenticity and confidentiality of the data that reach from the source sensors to the BS and detecting falsely injected data as early as possible, so that they cannot travel a long way towards the base station, which would save the unnecessary transmissions of the intermediate sensors. Thus it could save the network from wasting its crucial energy resource. We also propose an optional key refreshment mechanism to ensure data freshness in the network which could be employed depending on the requirements or the application at hand.

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

2 Related Works

Ye et al. [3] proposed a statistical en-route filtering (SEF) scheme to detect and drop false reports during the forwarding process. In their scheme, 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. [4] proposed the interleaved hop-by-hop authentication scheme that detects false reports through interleaved authentication. Their scheme guarantees that the base station can detect a false report when no more than t nodes are compromised, where t is a security threshold. In addition, their scheme guarantees that t colluding compromised sensors can deceive at most B noncompromised nodes to forward false data they inject, where B is $O(t^2)$ in the worst case. They also proposed a variant of this scheme which guarantees $B = 0$ and which works for a small t. Motivated by [4], Lee and Cho [5] proposed an enhanced interleaved authentication scheme called the key inheritance-based filtering that prevents forwarding of false reports. In their scheme, 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 all of the mentioned schemes as we create a logical 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, Preliminaries and Threat Model

We consider a wireless sensor network with dense deployment of the sensing devices. In this network, the BS and all the sensors are loosely time synchronized, and each node knows an upper bound on the maximum synchronized error. The sensors deployed in the network have the computational, memory, communication and power resources like the current generation of sensor nodes (e.g., MICA2 motes [6]). Once the sensors are deployed over the target area, they remain relatively static in their

respective positions. Each node transmits within its transmission range isotropically (in all directions) so that each message sent is a local broadcast. 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. The base station could not be compromised in any way. We assume that, no node could be compromised by any adversary while creating the tree structure in the network (in section 4.1). Initially, each node is equally trusted by the BS. Each sensor in the network has a shared secret key with the BS which is pre-loaded into its memory. The BS keeps an index of the ids of the sensors and the corresponding shared secret keys.

To ensure authenticity of sensor reports, we use one-way hash chain. A one-way hash chain [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}, …, K_0, such that, $\forall_i : 0 \leq i < n$, $K_i = F(K_{i+1})$. 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.

Due to the use of wireless communications, the nodes in the network are vulnerable to various kinds of attacks. However, dealing with the attack like jamming attack and other attacks [8], [9], [13] is beyond the scope of this paper. We assume that, an adversary could try to eavesdrop on all traffic, inject false packets, and replay older packets. If in any case, a node is compromised, it could be a full compromise where all the information stored in that particular sensor are exposed to the adversary or could be a partial compromise that is, partial information is exposed.

4 Securing Sensor Reports: Our Proposed Scheme

4.1 Initialization of the One-Way Hash Chain Number in the Network

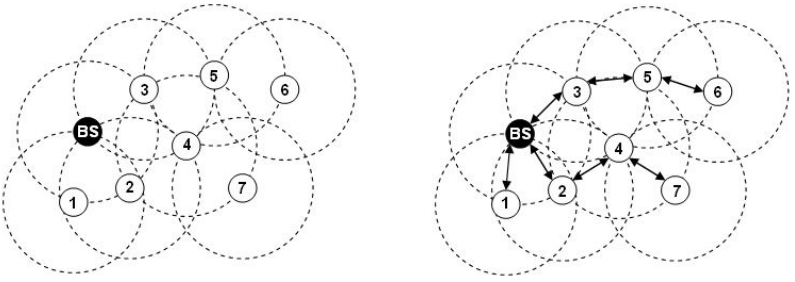
To provide authenticity of the sensor reports, all the intermediate nodes between any particular source node and the base station must be initialized with the basic one-way hash chain number. Let us suppose the initial OHC number is HS_0. To bootstrap the OHC number, the base station first generates a control packet containing HS_0 and a MAC (Message Authentication Code, 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 control packet generated by the base station B is:

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

Here, B is the id (indicates that this message is a control message sent from the base station) of the base station, *sid* indicates the sender id (required for the subsequent transmissions by the nodes in the network), *fid* is the id of the selected forwarder node. For base station, *sid* and *fid* are set to B.

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 base station as its forwarder node (in fact, the ultimate destination is the base station). Now, each of these nodes transmits the message again

within its own one-hop neighborhood (i.e., local broadcast) with its own id as *sid* and *B* as the *fid*. Any other node that has already got the control message directly from the base station (i.e., any other one-hop neighbor) ignores this packet.



(a) (b)

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

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*. When it gets 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 as neighbor information which are used for later computations. 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, that particular 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 possible 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 securely.

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 forwarder 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 node 4's turn comes and it transmits the local broadcast message using the control message, 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 and both of their *fid*s are *B* (which is in this case the base station itself). So, when the local broadcasts of node 1 or node 2 reach one another, 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 sensor report transmission phase after time slot t_{i+d}. We show the resultant network structure in Figure 1(b) after executing our algorithm on sample network in Figure 1(a).

4.2 Secure Data Transmission

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 checks the authenticity of the packet before forwarding to its immediate forwarder. Figure 2 illustrates the OHC utilization for secure data transmission.

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.

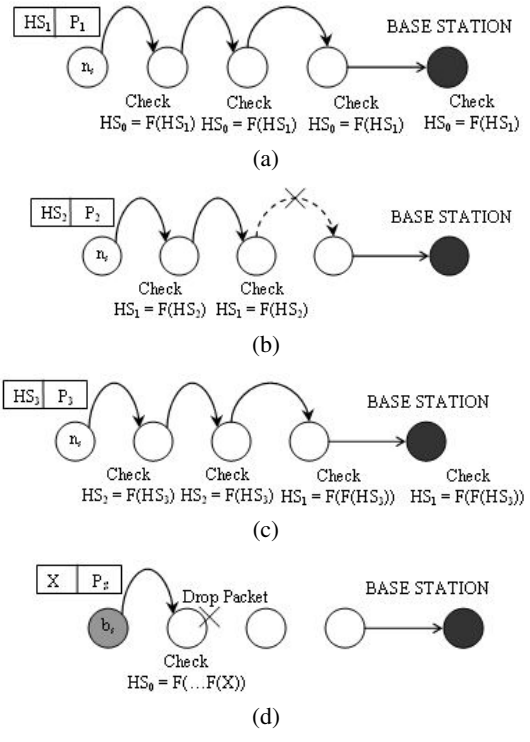


Fig. 2. (a) Authenticated packet delivery to BS (b) A packet could not reach the BS (c) OHC verification method is not hampered (d) Bogus packet detection and dropping

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:

$$B|K_s|\ MAC_{K_j}(B|K_s))$$

Where, K_j is the number in the key chain number corresponding to time slot t_j. To authenticate K_j, like the OHC initialization phase, *B* releases the key K_j in time slot t_{j+d}. On receiving this key, the nodes can verify the integrity and source authentication of K_j. 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.

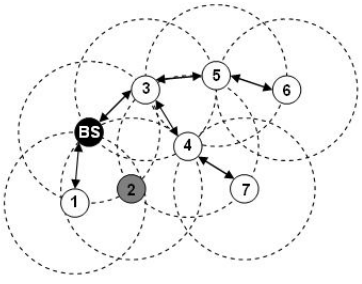


Fig. 3. Broken path recovery using upstream neighbor knowledge

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.

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 be detected by node 4. In the first phase, as it stored the id of node 3, it sends a message to node 3 informing that, node 3 would now be its forwarder. When node 4 is a source, for further packet transmission using node 3 it could use the later OHC numbers than that it used for sending the most recent packet to the BS. If there is any node upstream to node 3, through which its (node 4) packets have never been passed, takes the packet with caution and stores the current HS value as the initial HS value for node 4. Based on this HS value of node 4, the subsequent transmissions from node 4 are verified by node 3 and those upstream nodes (if any). The other node, node 7 also follows the same procedure and becomes connected again. As we are considering a highly dense deployment scenario, we think 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 (after a path is broken due to a node failure). 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 reports are discarded on the way to reach the base station. To overcome this problem, the entire OHC initialization phase could be made periodic (after certain interval, which is an application dependent parameter). The resultant structure after repairing the broken path of the sample network is shown in figure 3.

5 Analysis of Our Scheme

We analyze the security of our scheme with respect to two design goals; the ability of the base station to detect a false report and the ability of the nodes en-route to detect and filter false reports.

In our scheme, whenever the base station receives a report from any source sensor, it first checks the id of the sensor, checks the authenticity of the report by verifying the OHC number for that particular source, looks for the corresponding shared secret key and decrypts the packet. The base station could not be compromised in any way. So, it is in fact the final entity that could confirm the authenticity, confidentiality and integrity of the transmitted reports. Our security scheme is designed in a way that, any bogus report cannot reach the base station, rather would be detected and dropped by the intermediate nodes. However, if somehow a bogus packet is sent directly to the base station, it would certainly be discarded by it, for the failure of authentication check or because of false id. If in any application, the optional key refreshment mechanism is employed, once the time slot of releasing the new session key is over, the base station first tries to decrypt the incoming packets from any particular source with the X-ORed new key for that node. In case, if it produces garbage result, the BS tries with the previous shared secret key with that node (the previous key could easily be obtained again by X-ORing the most recent session key with the newly computed key for that node). This case might happen when somehow some node cannot get the new session key released by the base station.

We consider two types of attacks that should be detected by the sensors en-route to the base station:

Outsider Attack: In this case, as shown in figure 2(d) that, if an outsider node generates a packet with fake OHC number, the authentication must be failed in the very next node along the path and as a result this packet would never be forwarded. Simple verification of the OHC number prohibits the forwarding of such bogus packets and thus saves crucial energy resource of the network.

Insider Attack: If a legitimate node along any path is compromised, the attacker could grab the OHC sequence and the shared secret key. However, it should be noticed that, to use the OHC numbers successfully, the adversary should also know the last OHC number used by that particular node to send packet to the base station. Otherwise, any arbitrary use of the OHC number from that source might not be forwarded by the next intermediate node because of authentication failure. Now, in case if a node is fully compromised, that is if the adversary obtains all the required information, it actually gets the status of a legitimate node in the network. This fully compromised node could be used to generate false reports with valid authentication numbers. To prevent such type of malicious adversary, there are several factors come into play to detect the abnormal behavior of the node. In our scheme, the BS considers a report legitimate if it is reported by at least δ number of source nodes in the network, where δ is an application dependent parameter. So, the different or modified reports from a single source cannot convince the base station about any event. Also the base station notices the amount of packets generated by a particular source. These are basically the parts of an Intrusion Detection System (IDS) implemented in BS. The detailed description of the IDS is beyond the scope of this paper and will be reported in

our future works. The worst case scenario occurs if more than δ number of nodes in the network are somehow compromised. This sort of collaborative and large scale attack could be handled by periodic restructuring of the network. Finding an optimal value of the time interval for periodic restructuring is kept as our future work.

Our scheme ensures replay protection as the OHC numbers are checked as authentic only with later values. If a previous packet is captured by an adversary with the id of a legitimate node, and is sent again later, it would simply be discarded by the intermediate nodes. If an adversary uses a valid OHC number with invalid id, it would be detected by the BS and eventually the adversary would be exposed.

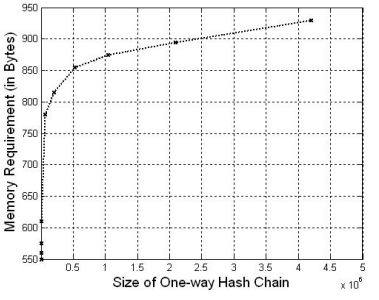


Fig. 4. Memory requirement for One-way Hash Chain generation

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. Recently, some efficient OHC generation algorithms for resource-constrained platforms have been proposed [10], [11], [12]. Among these algorithms, the fractal graph traversal algorithm [10] 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.

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 [10]. 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.

As the re-construction of the network structure and OHC initialization is periodic, new sensors could be added later in the network and they could actively participate in the network after a new tree is created. This feature ensures scalability of our scheme.

6 Conclusions and Future Works

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 and develop an Intrusion Detection System (IDS) to provide supplementary security supports in the network. Also we are currently 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, because of the page limitations, we have shortened some of the parts in this paper.

References

1. Akyildiz, I.F., Kasimoglu, I.H.: Wireless Sensor and Actor Networks: Research Challenges. Ad Hoc Networks 2(4), 351–367 (2004)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: A Survey. Computer Networks 38, 393–422 (2002)
3. Ye, F., Luo, H., Lu, S., Zhang, L.: Statistical En-Route Filtering of Injected False Data in Sensor Networks. IEEE Journal on Sel. Areas in Comm. 23(4), 839–850 (2005)
4. Zhu, S., Setia, S., Jajodia, S., Ning, P.: An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. In: Proceedings of S&P, pp. 259–271 (2004)
5. Lee, H.Y., Cho, T.H.: Key Inheritance-Based False Data Filtering Scheme in Wireless Sensor Networks. In: Madria, S.K., Claypool, K.T., Kannan, R., Uppuluri, P., Gore, M.M. (eds.) ICDCIT 2006. LNCS, vol. 4317, pp. 116–127. Springer, Heidelberg (2006)
6. Xbow Sensor Networks, available at: http://www.xbow.com/
7. Lamport, L.: Constructing digital signatures from one-way function. In technical report SRI-CSL-98, SRI International (October 1979)
8. Pathan, A.-S.K., Lee, H.-W., Hong, C.S.: Security in Wireless Sensor Networks: Issues and Challenges. In: Proc. of the 8th IEEE ICACT 2006, vol. II, pp. 1043–1048. IEEE Computer Society Press, Los Alamitos (2006)
9. Wood, A.D., Stankovic, J.A., Son, S.H.: JAM: a jammed-area mapping service for sensor networks. In: 24th IEEE RTSS, pp. 286–297. IEEE Computer Society Press, Los Alamitos (2003)
10. Coppersmith, D., Jakobsson, M.: Almost Optimal Hash Sequence Traversal. In: 6th International Financial Cryptography, Bermuda (March 2002)
11. Jakobsson, M.: Fractal hash sequence representation and traversal. In: 2002 IEEE International Symposium on Information Theory, Switzerland, IEEE Computer Society Press, Los Alamitos (2002)
12. Sella, Y.: On the computation-storage trade-offs of hash chain traversal. In: 7th International Financial Cryptography Conference, Guadeloupe (2003)
13. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. Elsevier's Ad Hoc Network Journal, 293–315 (September 2003)