Proceedings of 2015 IEEE 12th International Conference on Networking, Sensing and Control
Howard Civil Service International House, Taipei, Taiwan, April 9-11, 2015

Authentication in Flat Wireless Sensor Networks with Mobile Nodes

Mohamed Saleh
and Nagwa El-Meniawy
Pharos University in Alexndria (PUA)
Alexandria, Egypt

Essam Sourour
Alexandria University
Alexandria, Egypt

***Abstract*—Secure communication in Wireless Sensor Networks (WSNs) requires the verification of the identities of network nodes. This is to prevent a malicious intruder from injecting false data into the network. We propose an entity authentication protocol for WSNs, and show how its execution can be integrated as part of routing protocols. The integration between routing and authentication leads to two benefits. First, authentication is guided by routing; only nodes on a data path to the base station authenticate each other. Unnecessary protocol executions are therefore eliminated. Second, malicious nodes are not able to use the routing protocol to insert themselves into data paths. Our protocol assumes a flat WSN, i.e., no clustering or cluster heads. We also deal with node mobility issues by proposing a re-authentication protocol that an initially authenticated node uses when its position changes. Finally, we show how to implement the protocol using the TinyOS operating system.**

I. Introduction

Security is a major concern when a Wireless Sensor Network (WSN) is deployed in a hostile environment. Network nodes communicate via wireless channels and may be left unattended, which makes them vulnerable to attacks. Attackers may try to inject false data by adding their own nodes to the network. They may also try to eavesdrop on messages exchanged between legitimate nodes, change the content of messages or prevent them from reaching their intended destination. They may also try to mount Denial of Service (DoS) attacks in order to interrupt network services. Network designers should, therefore, determine a set of security objectives (properties), and implement security protocols that satisfy these objectives. For example, an objective is data secrecy, which means that data should not be intelligible to any unauthorized node, in order to prevent an eavesdropper from reading (understanding) content of messages.

Entity authentication [1] enables communicating entities to verify the identities of each other. In distributed systems, this is a crucial property; without it, authorization and accounting services cannot start. We will use the term "authentication" to mean entity authentication. This is not to be confused with message authentication [1] whose objective is to ensure message integrity and to identify the entity that created the message. Authentication protocols rely on cryptographic operations such as encryption and hashing. Therefore, to execute an authentication protocol, keys should be distributed (prior to deployment or via a secure channel) to network nodes. For WSNs, due to energy and storage limitations of nodes, symmetric keys are the prevalent choice [2] and several (offline) key predistribution schemes have been proposed [3]–[6]. At one extreme all nodes share the same key, which simplifies key management. However, this scheme is vulnerable since the disclosure of the key affects the whole network. At the other extreme, each pair of nodes share a distinct key; if a key is disclosed, only two nodes are compromised. But this means that, for a network of n nodes, we need $n(n-1)/2$ keys, and each node needs to store $n-1$ keys. For large n, key management becomes complicated. A compromise is to let each node, prior to deployment, store a set $\mathbb{K}$ of keys chosen from n possible keys. This is the key predistribution phase. Several methods [7], both deterministic and probabilistic, have been proposed for the choice of keys in $\mathbb{K}$. During communication, two neighboring nodes need to discover what keys they share. This is the key discovery phase and there is a possibility that no common key is found. A *key graph* [6] can then be constructed. Its vertices are nodes, and an edge is drawn between two nodes if they are neighbors that share a key. A path key is a key established between any two nodes A and B that are connected by a path $A, N_1, N_2, \ldots, N_l, B$ in the key graph. In the path key establishment phase, node A creates a key K_{AB}, encrypts it with the key shared between itself and N_1 and sends the encrypted key as a message to N_1. Now, node N_1 decrypts the message to get K_{AB}, it will then send it to N_2 after encrypting it with the key it shares with N_2. Thus, K_{AB} travels securely along the path until it reaches B.

We propose that key discovery, and key graph construction be guided by the routing protocol used by the network. So, nodes try to discover common keys at the same time that they try to discover routing paths. This can be achieved by integrating the key discovery phase, and, consequently the construction of the key graph, in the routing algorithm. We propose an authentication protocol that enables nodes to discover common keys while authenticating them to each other. We then show how to integrate our protocol with two routing algorithms, namely SPIN [8], [9] and TinyOS Beaconing [10]. As an advantage, routing is performed between authenticated entities, which avoids routing attacks such as sinkhole attacks [11]. Also, key discovery and path key establishment occur only between nodes that are on routing paths. In other words, a node executes the protocol with another node guided by the need to send (or relay) data and not with all its neighbors, hence energy is saved. Finally, when nodes are mobile, routing paths need to be updated and so does the key graph, which, in our setting, occur as a single integrated step instead of two.

978-1-4799-8069-7/15/$31.00 ©2015 IEEE

In the remainder of the paper, Sec. II discusses issues of entity authentication, and Sec. III introduces our proposed protocol. Then, Sec. IV shows how it can be integrated within routing. Section V presents an analysis of the proposed protocol, and, Sec. VI outlines the protocol implementation using TinyOS. Finally, Sec. VII concludes the paper.

II. Entity Authentication

Entity authentication is a security property that is informally defined as the ability to verify identities. It therefore involves two aspects: Identification and verification [12]. An identity, in a network setting, may be a MAC address, an IP address, a user name, etc. Verification is achieved by executing an authentication protocol during which a *claimant* proves to a *verifier* that it possesses a claimed identity. During the course of the protocol, the verifier may use the services of a trusted third party. For instance, a user logs in a network claiming to be user A. The proof is that A presents a password that no other user knows. The network access server may contact a trusted database to retrieve A's stored password in order to compare it with the one supplied by the user.[1] We consider node to node authentication, where a claimant node is successfully authenticated if it can prove (to a verifier node) that it "knows" some secret information.

A. Network Model

The network that we consider is made up from a large number (hundreds or thousands) of sensor nodes that are battery powered and have low computational and storage capacities. A base station is a node that acts as a gateway to the network, it has enough energy resources to outlast all other nodes. Moreover, the base station usually has much more computational and storage capacities than other nodes. Node to node communication is, in general, multihop, and so is the communication between a node and the base station. Nodes are mobile relative to each other and relative to the base station, which can be considered static. As time progresses, some nodes may cease to operate due to the depletion of their energy, whereas new nodes may be added to the network.

B. Authentication Protocols

Authentication protocols use cryptographic operations such as encryption and hashing in order to achieve their goal and prevent attackers from claiming identities that are not their own. Authentication can be one-way, where there is a party that acts as claimant and another as verifier. Or it can be two-way where a party is both a claimant, and a verifier of the other party's identity. Several authentication protocols have been designed for computer networks [1], some were discovered, later, to be flawed [1]. As for sensor networks, several challenges arise when we consider implementing authentication. First, computation and storage capacities are limited, which rules out the use of some algorithms [2], for instance, asymmetric cryptography which is computationally intensive and needs to store large keys. Second, a trusted server is not available, unless the base station is used for this role, provided that it is physically secured. Even in such case, this will constitute a communication bottleneck in large sensor networks. Finally, the location of nodes may change (or new nodes may be added) after initial deployment. In this case, a node may need to repeatedly authenticate itself to new nodes in its transmission range.

An authentication protocol is executed so that nodes can have "trust" in each others identities. A trusted node is allowed to use its authorized network resources. Moreover, when two nodes authenticate each other, they may agree on a *fresh* key K to be used for encryption of future communication. The key may also be used to compute a Message Authentication Codes (MACs) [1]. A MAC guarantees that a message m was created by a particular node, namely the node that knows K. It is also used to detect if m was tampered with during communication.

C. Intruder Model

The intruder is an active one, which means it not only eavesdrops on any message sent in the network, but may also participate in any protocol as an initiator or responder. It can thus mount man-in-the-middle attacks where it opens parallel sessions with multiple other parties conveying messages between them. Upon receiving a message, the intruder, may perform computations on it, then it may send it to a network node or just block it.

D. Key Distribution

In order to be able to execute an authentication protocol, secrets (keys) must be shared between nodes. The purpose of the protocol is that a node (the claimant) proves to another node (the verifier) that it knows a particular secret. As mentioned earlier, protocol steps may involve a trusted server which shares secrets with every node in the network. In this case, it is the trusted server that assures the verifier of the identity of the claimant. Key distribution is one of the trickiest issues in security systems [1]. We assume that, prior to deployment, each node i is preloaded with a set $\mathbb{K}_i$ of keys. This is the key predistribution phase. Two nodes may end up sharing at least one key, but a node, does not initially know the identities of other nodes with which it has keys in common. Some key predistribution schemes [5] may use deployment knowledge to increase the probability that two neighboring nodes share a key. However, we make no assumption about key predistribution except that each node ends up with a set of keys some of which are shared with other nodes.

III. Proposed Protocol

Our protocol is meant to enable two neighboring nodes to discover shared keys between them and, in the meanwhile, authenticate each other.

A. Design Criteria

We considered the following criteria when designing the protocol:

- The WSN has a flat structure, so, no nodes assume special roles, except the base station, which acts as gateway.
- The base station should not be involved in node to node authentication steps.

[1] This is an oversimplified example since passwords are not sent in clear text over untrusted channels.

TABLE I. NOTATION USED IN PROTOCOL DESCRIPTIONS.

Message	Meaning
$A, B, C, \ldots$	Node names
$K_{A1}, K_{A2}, \ldots$	Keys stored in node A
K_{AB}	Key shared between nodes A and B
N_A	Nonce generated by node A
$h(m)$	Cryptographic hash value of message m
$\text{MAC}_K(m)$	MAC value computed over message m using key K
$m_1 \| m_2$	Concatenation of m_1 and m_2
$\{m\}_K$	m encrypted by K
$i, j, k, \ldots$	Natural numbers
$A \rightarrow B : m$	A sends to B message m

- Nodes are mobile, and, when a node's neighbors change, re-authentication may be needed.
- The protocol should not reveal information about the keys stored in nodes, e.g., keys indices.
- The protocol should not prevent in-network processing of data at intermediate nodes.

B. Protocol Description

Table I shows the notation that we use in protocol descriptions. Assume Node A stores n keys and would like to run a mutual authentication protocol with node B. Figure 1 illustrates the protocol steps.

In Step 1, node A sends its identity A and a challenge (nonce N_A) to node B so that authentication can begin. It also sends MACs of $A|N_A$ computed using each key that A possesses. The MACs are meant to discover which of these keys is also possessed by node B. When B receives the message sent by A, it computes $\text{MAC}_{K_{Bj}}(A|N_A)$, for each key K_{Bj}. Each of the MACs that B computed is compared with those sent by A. If a match is found, B notes the order i of A's MAC that satisfies the equality $\text{MAC}_{K_{Ai}}(A|N_A) = \text{MAC}_{K_{Bj}}(A|N_A)$. In step 2, B sends i along with $(B|A|i|N_A|N_B)$ encrypted by K_{Ai}. When A receives this message it knows which key is shared between itself and B. It uses this key to decrypt the message and, in Step 3, returns N_B back to B. Now both A and B are authenticated to each other as we will show in Sec. V.

After completion of the protocol, each node may compute a *session key* K_{AB} that is used later to encrypt messages or compute MACs. One possible way of computing K_AB is to use the hash value $h(A|B|N_A|N_B)$. Then, K_{AB} is saved in each node's storage along with the identity of the other node. Now assume node A has moved and is no longer B's neighbor, it would like to re-authenticate itself in its new location. It can execute the authentication algorithm or it may use its shared key K_{AB} with B to authenticate itself to a node C, provided that C shares a key with B also. Re-authentication proceeds shown in Fig. 2. We note that re-authentication is simpler than authentication and that B is used here as a trusted authority.

IV. INTEGRATION WITH ROUTING PROTOCOLS

We show how the authentication protocol can be integrated with the TinyOS Beaconing [10] and the SPIN [8], [9] routing protocols.

A. TinyOS Beaconing

In TinyOS beaconing, a breadth-first spanning tree of the network is constructed. The base station broadcasts a beacon message that contains its ID and the number of hops set to 0. Any node that receives this beacon, marks the base station as its parent in the tree. It also rebroadcasts the beacon with its own ID and the number of hops incremented by 1. Any node hearing a beacon for the first time, stores the node ID obtained from the beacon as its parent node. It also stores the number of hops of its parent. When a node receives a beacon with the number of hops less than the one of its parent, it updates its routing information by changing the ID of its parent to the one it just received in the beacon. It also rebroadcasts the beacon with its own ID and the number of hops incremented by 1.

In order to be able to integrate our protocol within TinyOS beaconing, we identify three types of messages:

- `BC`: The original beacon, modified to carry the message in step 1 of our authentication protocol.
- `BC-RES`: This a response to a received `BC`. The response is sent by a node to its parent (whose ID is in the `BC` message) only if the node finds that it has at least one key in common with the parent. The `BC-RES` message carries the message from step 2 of our authentication protocol.
- `BC-ACK`: When a node receives a `BC-RES` from a child node, it sends a `BC-ACK` message to the child node. The `BC-ACK` message carries the message from step 3 of our authentication protocol.

By the end of the modified beaconing protocol execution, the routing tree, rooted at the base station, will contain authenticated data paths. No external node can insert itself in a routing path (unless if it knows one of the keys). Also, when the path is set up messages that are sent along any path can be encrypted.

B. SPIN

SPIN has three types of messages: `ADV`, `REQ`, and `DATA`. A source node advertises (`ADV`) what data it has to send, it describes the data using attribute-value pairs. A node that hears this, responds with a request (`REQ`) if it does not already have the data, or forwards the `ADV` message in case it has the data. Finally the data source sends the data (`DATA`). Each of the SPIN steps is modified to carry messages from a step in the authentication protocol, in order. So that the first `DATA` message sent will complete Step 3 of the authentication protocol and nodes will then be ready to start exchanging sensor data that is encrypted with the session key.

V. ANALYSIS AND RELATED WORK

A. Analysis

The protocols are analyzed from both the security and performance point of views. The authentication protocol is secure against replay attacks due to the use of nonces. It is a modified version of the one in [13], which is provably secure provided that the nonce generation algorithm satisfies pseudo-random properties. The modification is the computation of multiple MACs instead of just one MAC as in the case of the original protocol.

$$\begin{array}{lll} \text{Step 1.} & A \rightarrow B: & A|N_A|\,\text{MAC}_{K_{A1}}(A|N_A)\,|\ldots|\,\text{MAC}_{K_{An}}(A|N_A) \\ \text{Step 2.} & B \rightarrow A: & i|\,\{B|A|i|N_A|N_B\}_{K_{Ai}} \\ \text{Step 3.} & A \rightarrow B: & \{A|N_B\}_{K_{Ai}} \end{array}$$

Fig. 1. Protocol description.

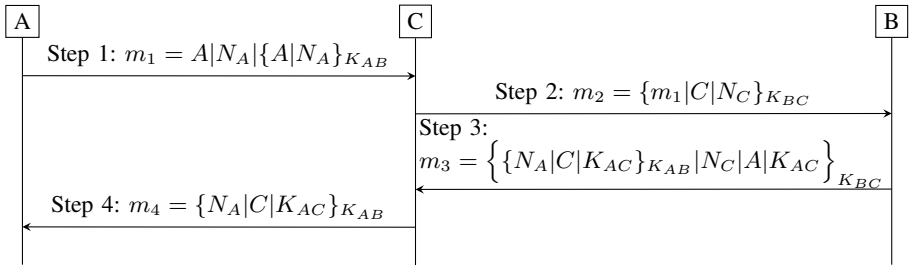

Fig. 2. Re-authentication Procedure.

The re-authentication protocol is immune against replay attacks due to the use of nonces. Also, when C receives K_{AC} from B it is confident that it is freshly generated by B (due to N_C) and that it is intended for communication with A, since A's identity is in the message encrypted by K_{BC}. A similar argument applies to A.

The bottleneck in the execution of the authentication protocol is the computation of I MACs by A, where I is the number of keys predistributed to A. The same applies for B which computes J MACs, where J is the number of keys stored in B. We note that if B does not have any common keys with A, it aborts the protocol, but it would have already computed J MACs and performed $I \times J$ value comparisons. The re-authentication protocol is much lighter and has the minimum number of steps when A and B are out of range with each other, but both are in range with C.

B. Related Work

Key predistribution mechanisms are many, an excellent survey is given in [7]. Usually papers discuss authentication and routing as two separate steps since authentication is normally executed on top of routing. In [14], protocols are given for message authentication and authenticated broadcast used for secure routing, however this involves message authentication and not entity authentication, also nodes share keys only with the base station.

In [15] and [16], the base station acts as a key distribution center and protocols are given for authentication. One work [16] relies on probabilistic key sharing among the nodes of a random graph and uses a simple shared-key discovery protocol for key distribution, revocation and node re-keying. When a pair of nodes N_1 and N_m do no directly share a key, a path $\langle N_1, N_2, \ldots, N_m\rangle$ is established, where for any two consecutive nodes N_i, N_{i+1} on the path, there exists at least one shared key. If such a path exists, nodes can communicate securely along the path using a *path key*, which represents the combination of encryption-decryption operations at each hop in the path. However, this work does not consider flat WSNs; cluster heads are sometimes used as "sub-base stations". Also, routing is not considered when the authors discuss authentication or path key establishment.

In the work closest to ours [17], authors consider node mobility and re-authentication in the context of hierarchical sensor networks where cluster heads are static. The authors introduce a protocol with multiple phases:

- Phase 0: The common neighbor discovery.
- Phase 1: Setting up neighbor sink relationship.
- Phase 2: Neighbor group authentication key share.
- Phase 3: initial node authentications.
- Phase 4 Node re-authentication with lower overhead than initial authentication.

Our work is different in that we do not consider the presence of cluster heads; our network has a flat structure. Also, we propose the integration of authentication within routing which is a novel idea.

VI. Implementation

TinyOS [18] is an operating system for WSN. Its applications, to be installed on sensor nodes, are written in a programming language called Network embedded systems C (NesC) [18]. A NesC application is composed of two types of components: Modules and configurations. Modules contain executable code in form of event-handlers and commands. A collection of event-handlers and commands is called an interface. An event handler is called whenever its event occurs, e.g., the reception of a message. A command is executed whenever it is called by a *user* of this command, e.g., sending a message. Events and commands cross the boundaries of modules, e.g., code in a module may call a command implemented in another module. For this to occur, however, modules need to be explicitly interconnected in a configuration. A configuration explicitly specifies which modules are interconnected, so that linking (of a function name to a function implementation) is totally known in compile time.

TinySec [19] is a library of security services for TinyOS. It provides a secure link layer protocol built on top of TinyOS link layer Active Messages (AMs) [18]. Also, it contains a library of crytptographic algorithms for encryption and message authentication. Our implementation uses TinySec's

encryption and MAC algorithms to implement our protocol. We also use the link layer implementation of TinyOS for single hop transmission of messages. We deal with the following issues:

- Storage of keys on a sensor node. For this, we designed the interface `KeyStore`, whose implementation saves keys as local data and can be queried to return keys to be used for encryption algorithms on the node.
- Storage of route information on sensor nodes. For this, we designed the interface `NodeKey` whose implementation saves a list of neighboring nodes along with the keys shared with these nodes.

To test the protocol, simulation is under way using the TOSSIM [20] simulator.

VII. Conclusion

We presented a mutual authentication protocol and showed how its execution can be integrated with some routing protocols. Our protocol achieves multiple goals. First, it enables two nodes to discover shared keys between them and in the meanwhile authenticate each other. Second, the protocol's execution is guided by routing needs, i.e., a node A executes the protocol only if it is a source node or a relay node of data that is sent to a sink node B, e.g., the base station. Node A does not then necessarily authenticate itself to all its neighbors and it can thus save energy. Third, when network topology changes due to node mobility, both routing paths and the key graph may be updated in an integrated step. Simulations using software environments or actual nodes need to be carried out in order to evaluate the performance of the protocol in practical conditions.

References

[1] D. R. Stinson, *Cryptography: Theory and practice*. Chapman and Hall/CRC; 3 edition, 2005.

[2] D. Boyle and T. Newe, "Securing wireless sensor networks: Security architectures," *JNW*, vol. 3, no. 1, pp. 65–77, 2008.

[3] J. Ibriq and I. Mahgoub, "A hierarchical key establishment scheme forwireless sensor networks," in *Proceedings of AINA*. IEEE Computer Society, 2007, pp. 210–219.

[4] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of SIGSAC: 10th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2003.

[5] A. K. Das, "A key establishment scheme for mobile wireless sensor networks using post-deployment knowledge," *CoRR*, vol. abs/1108.1302, 2011.

[6] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of SIGSAC: 9th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2002.

[7] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," Rensselaer Polytechnic Institute, Tech. Rep., 2005.

[8] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom-99)*. N.Y.: ACM Press, Aug. 15–20 1999, pp. 174–185.

[9] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 169–185, 2002.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proceedings of the International Conference on Architectural Support for Progrmming Languages and Operating Systems (ASPLOS)*, 2000.

[11] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[12] T. Y. C. Woo and S. S. Lam, "Authentication for distributed systems," *Internet Besieged: Countering Cyberspace Scofflaws*, 1998.

[13] Bellare and Rogaway, "Entity authentication and key distribution," in *CRYPTO: Proceedings of Crypto*, 1993.

[14] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Proceedings of Wireless Networks*, 2001, pp. 189–199.

[15] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 500–528, 2006.

[16] Y. Qiu, J. Zhou, J. Baek, and J. Lopez, "Authentication and key establishment in dynamic wireless sensor networks," *Sensors*, vol. 10, no. 4, pp. 3718–3731, 2010.

[17] K. Han and T. Shon, "Sensor authentication in dynamic wireless sensor network environments," *International Journal of RFID Security and Cryptography*, vol. 1, no. 1/2, 2012.

[18] P. Levis and D. Gay, *TinyOS Programming*. Cambridge University Press, 2009.

[19] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link-layer security architecture for wireless sensor networks," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[20] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the first International Conference on Embedded Networked Sensor Systems (SenSys'03)*, 2003.