A Novel Application for Secure Transmission of Data on Sensor Networks

S. Bala Subrahmanyam[1], Y. Ramesh Kumar[2]

[1]*M.Tech from JNTU Kakinada, Avanthi Institute of Engineering & Tech., Cherukupally, A.P. India.*

[2]*Assoc.Professor, Department of CSE, Avanthi Institute of Engineering & Tech., Cherukupally, A.P. India.*

Email: [1]bss.mahanti@gmail.com, [2]javaramesh143@gmail.com

***Abstract*- The many sensor networks like mobile sensor networks or networks that are deployed to monitor difficult areas are deployed in an unplanned fashion. So, any sensor in such a network can end up being adjacent to any other sensor in the network. To secure the communications between every pair of adjacent sensors in such a network, each sensor x in the network needs to store n -1 symmetric keys that sensor x shares with all the other sensors, where n is the number of sensors in the network. This storage requirement of the keying protocol is rather severe, especially when n is large and the available storage in each sensor is modest. Earlier efforts to redesign this keying protocol and reduce the number of keys to be stored in each sensor. In this mechanism, the security issues are takes place. It provides weak security. In this paper, we present a fully secure keying protocol where each sensor needs to store (n+1)/2 keys, which is much less than the n-1 keys that need to be stored in each sensor in the original keying protocol. We also show that in any fully secure keying protocol, each sensor needs to store at least (n -1)/2 keys. We can use encryption techniques at time of key transmission and data transmission.**

INTRODUCTION

Many wireless sensor networks are deployed in arbitrary and unplanned fashion. Examples of such networks are networks of mobile sensors and networks that are deployed in a hurry to monitor evolving crisis situations or continuously changing battle fields. In any such network, any deployed sensor can end up being adjacent to any other deployed sensor. Thus, each pair of sensors, say sensors x and y, in the network need to share a symmetric key, denoted Kx,y, that can be used to secure the communication between sensors x and y if these two sensors happen to be deployed adjacent to one another. In particular, if sensors x and y become adjacent to one another, then these two sensors can use their shared symmetric key Kx,y to authenticate one another (i.e. defend against impersonation) and to encrypt and decrypt their exchanged data messages (i.e. defend against eavesdropping).

It follows from this discussion that each sensor x in such a network is required to store n - 1 symmetric keys, where n is the total number of sensors in the network and each stored key is shared between sensor x and a different sensor in the network. This requirement that each sensor in the network stores n-1 symmetric keys, where n is the number of sensors in the network, is rather severe especially when n is large and the available storage to store keys in every sensor is modest.

There are two main keying protocols that were proposed in the past to reduce the number of stored keys in each sensor in the network. We refer to these two protocols as the probabilistic keying protocol and the grid keying protocol. In the probabilistic keying protocol, each sensor in the network stores a small number of keys that are selected at random from a large set of keys. When two adjacent sensors need to exchange data messages, the two sensors identify which keys they have in common then use a combination of their common keys as a symmetric key to encrypt and decrypt their exchanged data messages. Clearly, this protocol can probabilistically defend against eavesdropping.

Unfortunately, the probabilistic keying protocol suffers from the following problem. The stored keys in any sensor x are independent of the identity of sensor x and so these keys cannot be used to authenticate sensor x to any other sensor in the network. In other word, the probabilistic protocol cannot defend against impersonation.

In the grid keying protocol, each sensor is assigned an identifier which is the coordinates of a distinct node in a two-dimensional grid. Also each symmetric key is assigned an identifier which is the coordinates of a distinct node in two-dimensional grid. Then a sensor x stores a symmetric key K iff the identifiers of x and K satisfy certain given relation. When two adjacent sensors need to exchange data messages, the two sensors identify which keys they have in common then use a combination of their common keys as a symmetric key to encrypt and decrypt their exchanged data messages.

The grid keying protocol has two advantages (over the probabilistic protocol). First, this protocol can defend against impersonation (unlike the probabilistic protocol) and can defend against eavesdropping (like the probabilistic protocol). Second, each sensor in this protocol needs to store only O(log n) symmetric keys, where n is the number of sensors in the network. Unfortunately, it turns out that the grid keying protocol is vulnerable to collusion. Specifically, a small gang of adversarial sensors in the network can pool their stored keys together and use the pooled keys to decrypt all the exchanged data messages in the sensor network. This situation raises the following important questions:

Is it possible to design a keying protocol, where each sensor stores less than n-1 symmetric keys and yet the protocol is deterministically secure against impersonation, eavesdropping, and collusion?

In this paper, we investigate a sensor network whose topology is not planned in advance, prior to the deployment of the network. Thus, when the network is deployed, any sensor can end up being adjacent to any other sensor in the network. (There are many occasions when a sensor network needs to be deployed before its topology can be planned in great detail. For example, when a wildfire breaks out unexpectedly, a sensor network that monitors the fire may need to be deployed in a hurry, before the network topology can be planned accurately. A second example, when a sensor network is deployed in a battlefield whose perimeter is continuously changing, the topology of the network cannot be determined fully until the time when the network is to be deployed.

As a third example, if the deployed sensor network is mobile, then a detailed plan of the initial topology may be of little value.) In this network, when a sensor x is deployed, it first attempts to identify the identity of each sensor adjacent to x, then starts to exchange data with each of those adjacent sensors.

Any sensor z in this network can be an "adversary", and can attempt to disrupt the communication between any two legitimate sensors, say sensors x and y, by launching the following two attacks:

1) Impersonation Attack: Sensor z notices that it is adjacent to sensor x while sensor y is not. Thus, sensor z attempts to convince sensor x that it (z) is in fact sensor y. If sensor z succeeds, then sensor x may start to exchange data messages with sensor z, thinking that it is communicating with sensor y.

2) Eavesdropping Attack: Sensor z notices that it is adjacent to both sensors x and y, and that sensors x and y are adjacent to one another. Thus, when sensors x and y start to exchange data messages, sensor z can copy each exchanged data message between x and y.

If the network has n sensors, then each sensor in the network needs to store (n-1) symmetric keys before the network is deployed. If n is large, then the storage requirement, just to store the required shared keys, is relatively large, especially since the size of storage in each sensor is typically small.

To solve this problem, we present the following two results in this paper:

1) Efficiency: There is a keying protocol, where each sensor shares a distinct symmetric key with every other sensor in the network, and yet each sensor needs to store exactly (n+1)/2 symmetric keys, before the network is deployed.

2) Optimality: In every keying protocol, where each sensor shares a distinct symmetric key with every other sensor in the network, each sensor needs to store at least (n-1)/2 symmetric keys, before the network is deployed.

METHODOLOGY

A MUTUAL AUTHENTICATION PROTOCOL:

Before the sensors are deployed in a network, each sensor x is supplied with the following items:

1) One distinct identifier ix in the range 0…n-1

2) One universal key ux

3) (n-1)/2 symmetric keys Kx,y = H(ix|uy) each of which is shared between sensor x and another sensor y, where ix is below iy After every sensor is supplied with these items, the sensors are deployed in random locations in the network.

Now if two sensors x and y happen to become adjacent to one another, then these two sensors need to execute a mutual authentication protocol so that sensor x proves to sensor y that it is indeed sensor x and sensor y proves to sensor x that it is indeed sensor y.

The mutual authentication protocol consists of the following six steps.

Step 1: Sensor x selects a random nonce nx and sends a hello message that is received by sensor y.
x → y : hello(ix, nx)
Step 2: Sensor y selects a random nonce ny and sends a hello message that is received by sensor x.
x ← y : hello(iy, ny)
Step 3: Sensor x determines whether ix is below iy. Then it either fetches Kx;y from its memory or computes it. Finally, sensor x sends a verify message to sensor y.
x → y : verify(ix, iy, H(ix|iy|ny|Kx,y))
Step 4: Sensor y determines whether iy is below ix. Then it either fetches Kx,y from its memory or computes it. Finally, sensor y sends a verify message to sensor x.
x ← y : verify(iy, ix, H(iy|ix|nx|Kx,y))
Step 5: Sensor x computes H(iy|ix|nx|Kx,y) and compares it with the received H(iy|ix|nx|Kx,y). If they are equal, then x concludes that the sensor claiming to be sensor y is indeed sensor y. Otherwise, no conclusion can be reached.
Step 6: Sensor y computes H(ix|iy|ny|Kx,y) and compares it with the received H(ix|iy|ny|Kx,y). If they are equal, then y concludes that the sensor claiming to be sensor x is indeed sensor x. Otherwise, no conclusion can be reached.

A DATA EXCHANGE PROTOCOL

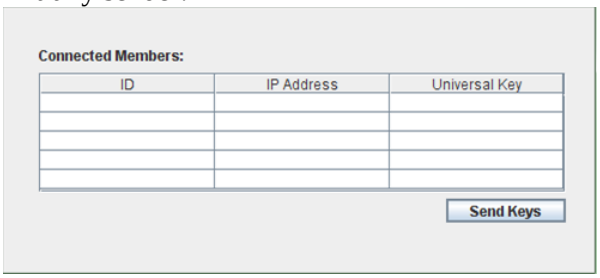
After two adjacent sensors x and y have authenticated one another using the mutual authentication protocol described in the previous section, sensors x and y can now start exchanging data messages according to the following data exchange protocol. (Recall that nx and ny are the two things that were selected at random by sensors x and y, respectively, in the mutual authentication protocol.)

Step 1: Sensor x concatenates the nonce ny with the text of the data message to be sent, encrypts the concatenation using the symmetric key Kx,y, and sends the result in a data message to sensor y.
x → y : data(ix, iy, Kx,y (ny|text))
Step 2: Sensor y concatenates the nonce nx with the text of the data message to be sent, encrypts the concatenation using the symmetric key Kx,y, and sends the result in a data message to sensor x.
x ← y : data(iy, ix, Kx,y (nx|text))

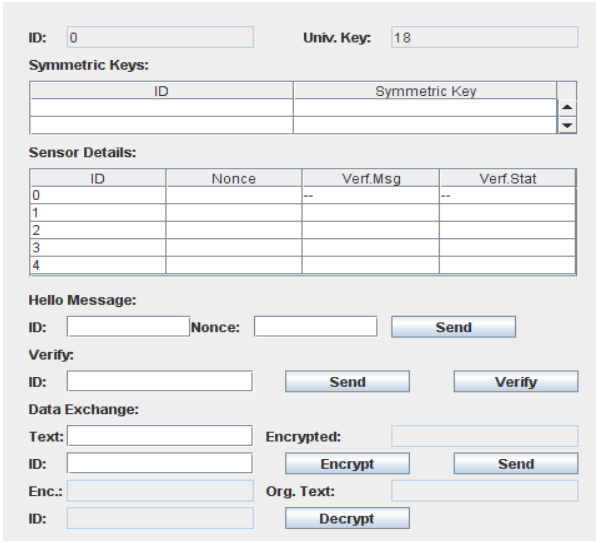
Sensors x and y can repeat Steps 1 and 2 any number of times to exchange data between themselves.

The above two algorithms are implemented for the following data as shown below:

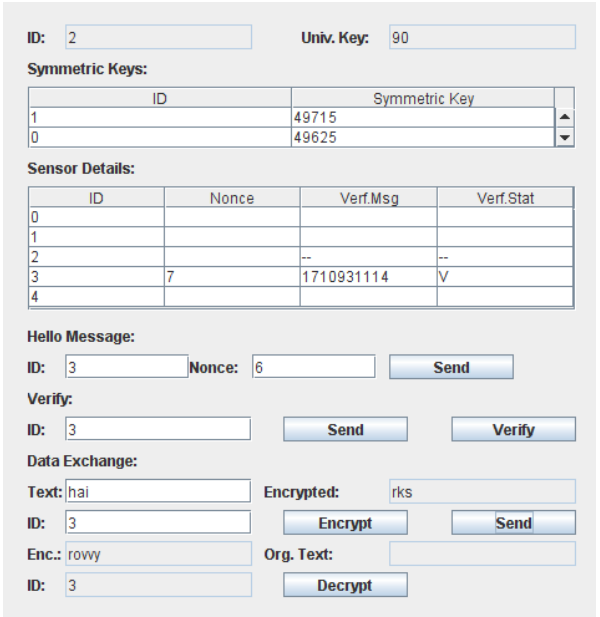
Initially sender:



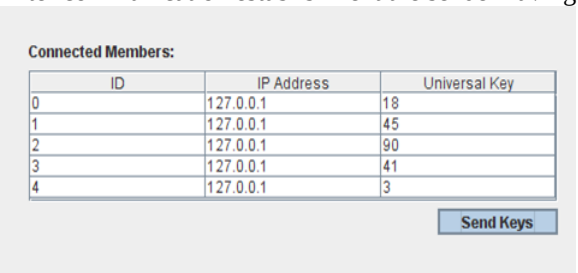
Receiver:



Transmission process as follows:
Receiver info:



After communication establishment the sender having



Typically, each sensor in a sensor network with n sensors needs to store n - 1 shared symmetric keys to communicate securely with each other. Thus, the number of shared symmetric keys stored in the sensor network is n(n-1). However, the optimal number of shared symmetric keys for secure communication, theoretically, is $\binom{n}{2} = n(n-1)/2$.

Although there have been many approaches that attempt to reduce the number of shared symmetric keys, they lead to a loss of security: they are all vulnerable to collusion.

In this paper, we show the best keying protocol for sensor networks, that needs to store only (n + 1)/2 shared symmetric keys to each sensor. The number of shared symmetric keys stored in a sensor network with n sensors is n(n + 1)/2, which is close to the optimal number of shared symmetric keys for any key distribution scheme that is not vulnerable to collusion.

It may be noted that in addition to the low number of keys stored, and the ability to resist collusion between sensors, our keying protocol has two further advantages.

Firstly, it is uniform: we store the same number of keys in each sensor. Secondly, it is computationally cheap, and thus suitable for a low-power computer such as a sensor: when two sensors are adjacent to each other, the computation of a shared symmetric key requires only hashing, which is a cheap computation and can be done fast. As our protocol has many desirable properties, such as efficiency, uniformity and security, we call this protocol the best keying protocol for sensor networks.

REFERENCE

[1] The Best Keying Protocol for Sensor Networks by Taehwan Choi,H. B. Acharya,Mohamed G. Gouda

[2] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS), 2002, pp. 299–308.

CONCLUSION

[3] S. Sana and M. Matsumoto, "Proceedings of a wireless sensor network protocol for disaster management," in Information, Decision and Control (IDC), 2007, pp. 209 –213.

[4] S. Hynes and N. C. Rowe, "A multi-agent simulation for assessing massive sensor deployment," Journal of Battlefield Technology, vol. 7, pp. 23–36, 2004.

[5] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in Proceedings of the 9th ACM conference on Computer and communications security (CCS), 2002, pp. 41–47.

[6] L. Gong and D. J. Wheeler, "A matrix key-distribution scheme," Journal of Cryptology, vol. 2, pp. 51–59, January 1990.

[7] S. S. Kulkarni, M. G. Gouda, and A. Arora, "Secret instantiation in ad-hoc networks," Special Issue of Elsevier Journal of Computer Communications on Dependable Wireless Sensor Networks, vol. 29, pp. 200–215, 2005.

[8] A. Aiyer, L. Alvisi, and M. Gouda, "Key grids: A protocol family for assigning symmetric keys," in Proceedings of IEEE International Conference on Network Protocols (ICNP), 2006, pp. 178–186.

[9] E. S. Elmallah, M. G. Gouda, and S. S. Kulkarni, "Logarithmic keying," ACM Transactions on Autonomic Systems, vol. 3, pp. 18:1–18:18, December 2008.

AUTHORS BIOGRAPHY

S. Bala Subrahmanyam, Studying M.Tech in Computer Science Engineering. Avanthi Institute of Engineering & Tech., Cherukupalli, Vizianagaram, A.P. India.

Y. Ramesh Kumar is working as Assoc. Professor, in CSE Department, Avanthi Institute of Engineering & Tech., Cherukupalli, Vizianagaram, A.P. India. He has received his M.Tech from Andhra University, Visakhapatnam.

