Contents lists available at SciVerse ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Fast track article

The best keying protocol for sensor networks

Taehwan Choi [a,*], H.B. Acharya [a], Mohamed G. Gouda [a,b]

[a] Computer Science Department, The University of Texas at Austin, Austin, TX 78712, USA
[b] The National Science Foundation, 4201 Wilson Boulevard, Arlington, VA 22230, USA

ARTICLE INFO

Article history:
Available online 19 June 2012

Keywords:
Sensor network
Keying protocol
Collusion-proof
Secure communication
Lower bound
Optimal keying
Uniform keying

ABSTRACT

Many sensor networks (especially networks of mobile sensors or networks that are deployed to monitor crisis situations) are deployed in an arbitrary and unplanned fashion. Thus, 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 an upper bound on 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 have produced protocols that are vulnerable to impersonation, eavesdropping, and collusion attacks. 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.

1. Introduction

Many wireless sensor networks are deployed in arbitrary and unplanned fashion. Examples of such networks are networks of mobile sensors [1] and networks that are deployed in a hurry to monitor evolving crisis situations [2] or continuously changing battle fields [3].

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 by $K_{x,y}$, 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 $K_{x,y}$ 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* [4], 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

* Corresponding author. Tel.: +1 512 466 8300.
E-mail addresses: ctlight@cs.utexas.edu (T. Choi), acharya@cs.utexas.edu (H.B. Acharya), mgouda@nsf.gov (M.G. Gouda).

1574-1192/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2012.06.007

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* [5–8], 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 an upper bound on 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 show that the answer to this question is yes. In particular, we present a new keying protocol where each sensor stores only $(n + 1)/2$ symmetric keys, and yet the protocol is deterministically secure against impersonation, eavesdropping, and collusion. We also show that this new protocol is near optimal by showing that each sensor, in any keying protocol that is deterministically secure against impersonation, eavesdropping, and collusion, needs to store at least $(n - 1)/2$ symmetric keys.

This paper is a revised and expanded version of our paper that was published in D-SPAN workshop [9].

2. Sensor networks and adversaries

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.

To defend against these two types of attacks, sensors x and y need to share a symmetric key, denoted by $K_{x,y}$ or $K_{y,x}$. The shared key $K_{x,y}$ needs to be known only to both sensors x and y, and not to any other sensor in the network, before these two sensors are deployed in the network. In Sections 4 and 5, we show how sensors x and y can use their shared key $K_{x,y}$ to defend against these two types of attacks.

3. Keying protocols for sensor networks

A *keying protocol* for a sensor network is a scheme for assigning a unique symmetric key $K_{x,y}$ to each pair of distinct sensors x and y in the network. Each symmetric key $K_{x,y}$, that is assigned by the keying protocol, becomes known only to sensors x and y (and not to any other sensor in the network) before the network is deployed and before the adjacent sensors in the deployed network start to communicate with one another.