Confidentiality in Wireless Sensor Networks

B.Veeramallu, S. Sahitya, Ch. LavanyaSusanna

Abstract: ***While much research has focused on making sensor networks that feasible and useful security has received little attention. We present a set of security protocols optimized for sensor networks: they are confidentiality and authentication, data freshness, data integrity. By using the Sensor Network Encryption Protocol we explains the basic primitives for providing confidentiality, authentication between the two nodes, data integrity and message freshness present in a wireless sensor network. That was designed as base component of Security Protocols for Sensor Networks. Here mainly two security properties are checked: authenticity and Confidentiality of similar messages components. That the first case is the communication between the networks nodes and base station in order to retrieve node Confidential information. In the second case is a key distribution protocol in a sensor network using SNEP (sensor network encryption protocol) for securing messages.***

Keywords: Sensor networks, secure communication protocols, mobile ad hoc networks, authentication of wireless communication.

I. INTRODUCTION

A challenge has been become by security in wireless sensor networks [8].Low capabilities of devices, in terms of computational power and energy consumption. It makes difficult by using traditional security protocols. The low computational power implies that special cryptographic algorithms that require less powerful processors need to be used. The combination of both problems leads us to a situation where new solutions to security protocols need to be taken. These types of new approaches take into account basically two main goals: reduce the overhead that protocol imposes to messages, and provide reasonable protection while limiting use of resources.

With these limited computation resources available on our plat-form, we cannot afford to use asymmetric cryptography and so we use symmetric cryptographic primitives to provide the security. Due to the limited program store, we construct all cryptographic primitives (i.e. encryption [10], message authentication code, hash, random number generator) out of a single block cipher for code reuse. To decrease communication overhead we exploit common state between the communicating parties.

1.2 System assumptions:

Before we discuss the security requirements and present our security infrastructure that we need to defined the system architecture and the trust for the requirements. The goal of this work is to propose the general security infrastructure. It is applicable to a variety of sensor networks [8].

Manuscript received on January 2013.

B.Veeramallu, department of computer science and engineering, KLUniversity.

S.Sahitya, department of computer science and engineering, KLUniversity.

Ch.LavanyaSusanna, department of computer science and engineering ,KLUniversity.

1.2.1. Communication architecture:

Generally, the broadcast is the fundamental communication primitive because of the sensor nodes communicate over a wireless network. The baseline protocols account for this property: on one hand they affect the trust assumptions, and on the other they minimize the energy usage.

A Smart Dust sensor network forms around one or more base stations, which the interface sensor network to the outside network. The sensor nodes are establishing a routing forest, with a base station at the root of each tree. Periodic transmission of beacons allows nodes to generate a routing topology. Each node can forward a message towards a base station, find the packets addressed to it, and handle a message broadcasts. The base station that accesses individual nodes using source routing. We assume that the base station has capabilities same as network nodes, except that it has enough battery power to surpass the lifetime of all sensor nodes, enough memory to store cryptographic keys, and that means for communicating with the outside networks.

Here an advantage with sensor networks, because most communication involves the base station and is not between two local nodes. The communication pattern within our network is of three categories:

- Node to base station communication,
  E.g.: sensor readings.
- Base station to node communication.
- E.g.: specific requests.
- Base station to all nodes, e.g., the routing beacons, queries or reprogramming of the entire network.

Our security goal is to locate these communication patterns, though we also show how to take over our baseline protocols to other communication patterns, i.e. node to node or node broadcast.

1.2.2. Trust requirements

Generally, the sensor networks may be kept in un trusted locations. While it may be possible to guarantee the integrity of the each node through dedicated secure microcontrollers (e.g., [4] or [5]), we feel that such architecture is too restrictive and does not generalize to the majority of sensor networks. Instead of this, we assume that individual sensors are un trusted. Our goal is to design the SPINS [3] key setup so a compromise of a node does not spread to other nodes. Basic wireless communication is not secure. Because it is broadcast, any adversary can eavesdrop on traffic, insert new messages, and replay old messages. So, our protocols do not place any trust assumptions on the communication infrastructure, except that the messages are delivered to the destination with non-zero probability.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Since the base station is the gateway for the nodes to communicate with the outside world, compromising the base station can leads to the entire sensor network useless. Therefore, the base stations are a necessary part of our trusted computing base upon Our trust setup reflects this and so all sensor nodes intimately trust the base station: at creation time, each node gets a master secret key X which it shares with the base station of all other keys are derived from this key, as [2].

Finally, each node trusts itself. This assumption that seems necessary to make any forward progress. In the particular, we trust the local clock to be accurate, i.e. to have small drift. This is necessary for a authenticated broadcast protocol [2].

1.2.3. Design guidelines

With the limited computation resources available on our platform, we cannot attempt to use asymmetric cryptography and so we use symmetric cryptographic primitives to construct the SPINS [3] protocols. Due to the limited program store, we design all cryptographic primitives out of a single block cipher for code reuse. To minimize communication overhead we exploit common state between the communicating parties. Here we discuss the requirements in section2 methodology in section 3.Properties, applications in section 4 and 5.

1.3 notations:

We use the given below notations to discuss security protocols and cryptographic operations in this article: they are used in section 3.

- A,B are principals, such as communicating nodes.
- NA is a nonce generated by A (a nonce is an unpredictable bit string; it usually used to achieve freshness).
- XAB denotes the master secret (symmetric) key which is shared between A and B. No direction information is stored in this key, so we have the XAB=XBA.
- KAB and KBA denote the secret encryption keys shared between A and B. A and B derive the encryption key from the master secret key XAB based on the direction of the communication: KAB = FXAB (1) and KAB=FXAB (3), where F is a Pseudo-Random Function (PRF) [6]. We describe the details of key derivation in further detail [2] in that section 6.
- K'AB and K'BA denote the secret MAC [12] keys shared between A and B. A and B derive the encryption key from the master secret key XAB based on the direction of the communication: K'AB = FXAB (2) and K'BA=FXAB (4), where X is a pseudo-random function.
- [M]KAB is the encryption of message M with the encryption key KAB.

II. SECURITY TYPES

This section describes that the security properties required by sensor networks, and shows how they are directly applicable in a typical sensor network.

2.1. Data confidentiality

A sensor network should not leak sensor readings to its surrounding networks. In many applications (e.g., key distribution [13], [14], [15]) nodes can communicate highly sensitive data. That a standard approach for keeping sensitive data is to be secret by encrypting the data with a secret key that only intended for the receivers possess, hence achieving confidentiality. Given that observed communication patterns, we also set up secure channels between nodes and base stations and later bootstrap other secure channels as necessary.

2.2. Data authentication

Message authentication is important for many applications in sensor networks (including administrative tasks such as network reprogramming or controlling sensor node for duty cycle). Since an adversary can easily inject messages, the receiver needs to ensure that data used in any decision-making process originates from a trusted source. Informally, the data authentication allows a receiver to verify that the data really was sent by the claimed sender.

Informally, data authentication allows a receiver to verify that the data really was sent by the claimed Sender. In the two-party communication case, that data authentication can be achieved through a purely symmetric mechanism:

The sender and the receiver share a secret key to compute a message authentication code (MAC) [1] of all communicated data. When the message with a correct MAC arrives, the receiver knows that it must been sent by the sender.

This style of authentication cannot be applied to a broad-cast setting, without placing a much stronger trust assumptions on the network nodes that one sender wants to send authentic data to mutually un trusted [11] receivers are using a symmetric MAC is insecure: any one of the receivers knows the MAC key and hence, it could impersonate the sender and forge messages to other receivers. Hence, to need an asymmetric mechanism to achieve authenticated Broadcast. One of our contributions is to design authenticated broadcast from symmetric primitives only, to introduce asymmetry with delayed key disclosure and one-way function key chains.

2.3. Data integrity

In communication, data integrity ensures the receiver that the received data is not change in transit by an adversary. In SPINS [2], we achieve data integrity through authentication, that which is a stronger property.

2.4. Data freshness

Sensor networks pass on measurements over time, so it is not sufficient to guarantee confidentiality and authentication; we also must note that each message is fresh. Informally, data freshness means that the data is the latest, and it ensures that no adversary replayed old messages.

We found two types of freshness: weak freshness, which gives a partial message ordering, but takes no delay information, strong freshness, which provides a total order on a request-response pair, that allows for delay estimation. Weak freshness is used in sensor measurements, while strong freshness is used in time synchronization within the networks.

III. METHODOLOGY

3.1 Existing system:

An autonomous system of Wireless Sensor networks are made up of collaborative mobile nodes. Wireless Sensor networks can be cleverly setup without relying on any pre-existing infrastructure.

Implementing that a public key user authentication is a challenging issue in Wireless Sensor networks because of its salient nature of the network. The user authentication is of 3steps. In the first step Registration of user that is successful login will be happened, in the second step of user smart card will be validated with symmetric key and in third phase using the key the authentication of the user will be done. But here there is no confidentiality of the data in this system.

3.2 Proposed system:

Data confidentiality is one of the most basic security primitives and it is used in almost every security protocol. A simple form of the confidentiality can be achieved through encryption, but the pure encryption is not sufficient and Another important security property is semantic security, which it ensures that an eavesdropper has no information about the plaintext that even if it sees multiple encryptions of the same plaintext. For example, even if an attacker has encryption of a 0 bit and an encryption of 1 bit, it will not help it distinguish whether a new encryption is an encryption of 0 or 1.A basic technique to achieve this is randomization: Before encrypting the message with a chaining encryption function (i.e. DES-CBC), the sender precedes the message with a random bit string.

This prevents from the attacker inferring the plaintext of encrypted messages if he knows plaintext. Cipher text pairs encrypted with the same key that is used in encryption. Sending the randomized data over a wireless channel, however, more energy is to be needed. So we design another cryptographic mechanism. It achieves semantic security with no additional transmission overhead. We use two counters shared by the parties, for the block cipher in counter mode (CTR) [2]. In traditional approach we send the counter along with each message. But since we are using sensors in the communicating parties share only one counter and increment it after each block, the sender can save energy by sending the message without the counter. At end of this section we explain a counter exchange protocol, which the communicating parties use to synchronize (or resynchronize) their counter values. We use a message authentication code (MAC) for achieving two-party authentication and data integrity,

A good security design is not to reuse the similar cryptographic key for different cryptographic primitives; this prevents potential interaction between the primitives that might introduce a weakness. Hence we derive independent keys for our encryption and MAC [12] operations.

3.2.1 Counter exchange protocol:

To achieve small SNEP messages, we assume that two communicating Parties A and B know each other with counter values CA and CB and so the counter does not need to be added to each encrypted message. However, if any messages are lost then the shared counter state can become inconsistent. Now present protocols to synchronize the counter state. To bootstrap the counter [2] values that initially, we use the following protocol:

A --> B : CA,
B --> A : CB, MAC (K'BA CA || CB)
A --> B : CB, MAC (K'AB, CA || CB)

Note that the counter values are not secret, so there is no need of using encryption method. However, this protocol requires strong freshness, so both parties A and B use their counters as a nonce (assuming that the protocol never runs twice with the same counter values, hence incrementing counters if necessary) and here the MAC does not need to include the names of A or B.

Since the MAC keys K'AB and K'BA implicitly bind the message to the parties, and that ensure the direction of the message. If party A realizes that the counter CB of party B is not synchronized any more, A can request the current counter of B using a nonce NA to ensure strong freshness of the reply:

A --> B : NA
B --> A : CB, MAC (K'BA NA || C B)

To prevent a potential denial-of-service (DoS) attack, where an attacker keeps sending false messages to lure the nodes into performing counter synchronization and the nodes can switch to sending the counter with each encrypted message they send. To detect such a denial-of-service [2] attack, there is an another approach is, to attach another short MAC to the message that does not depend on the counter.

IV. PROPERTIES

4.1 Semantic security: Since the counter value is incremented after each message [7] that means the same message is encrypted differently at each time. The counter value is sufficiently long enough, So never repeat within the lifetime of the node.

4.2 Data authentication: If the MAC verifies correctly, a receiver knows that the message is send from the claimed sender.

4.3 Replay protection: The counter value in the MAC prevents replay of old messages. If the counter were not present in a MAC, an adversary could easily replay messages.

4.4 Weak freshness: If the message is verifies correctly, the receiver knows that a message must have been sent after the previous message it received correctly. This leads a message ordering and yields weak freshness. Low communication overhead. The counter state is kept at each end point and does not need to be sent in each message.

V. APPLICATIONS

SNEP provides a number of advantages.

- SNEP has low communication overhead; it only adds 8 bytes per message. like many cryptographic protocols it uses a counter, but here we avoid transmitting the
- Counter value by keeping state at the both end points.
- Third, SNEP achieves semantic security, a strong security property which prevents eavesdroppers from inferring the message content from the encrypted message (see discussion below).
- At last, the same simple and efficient protocol also gives us data authentication, replay protection, and weak message freshness.

VI. CONCLUSION

We designed security subsystem for an extremely limited sensor network platform.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Here we have identified and implemented useful security protocols for sensor networks: authenticated and confidential communication, and authenticated broadcast. We have implemented applications including an authenticated routing scheme and a secure node-to-node key agreement protocol. Most of our design is universal and it is applicable to other networks of low-end devices. Our primitives only depend on fast symmetric cryptography, and apply to different device configurations.

Energy spent on sending or receiving messages is to be compared with our limited platform energy spent for security is negligible. It is possible to encrypt and authenticate all sensor readings. The communication costs are also very small. Data authentication, freshness, and confidentiality properties use 6 bytes out of 30 byte packets. So, it is feasible to guarantee these properties on a per packet basis.

REFERENCES

1. T. Sarika and Shaik Shah Nawaz," Multi-Factor User Authentication in Wireless Sensor Networks "(International Journal of Computer Science and Tele-communications, September 2011).
2. For sensor networks Adrian perrig, Robert Szewczyk, J.D.Tygar, Victorwen and David E. Culler "Security Protocols for sensor networks [ACM] Wireless Networks, 8:5, September 2002, pp. 521-534).
3. Llanos Tobarra, Diego Cazorla and Fernando Cuartero." Formal Analysis of Sensor Network Encryption Protocol".
4. Atmel, Secure Microcontrollers for Smart Cards, http://www.atmel.com/atmel/acrobat/1065s.pdf.
5. Dallas, iButton:A Java powered cryptographic iButton, http://www.ibutton.com /ibuttons/java. html.
6. O. Gold Reich, S. Gold wasser and S. Micali, How to construct random functions, Journal of the ACM 33(4) (1986) 792.807.
7. S. Basagni, K. Herrin, E.Rosti and D. Bruschi, Secure Pebblenets, in: ACM International Symposium on Mobile Ad Hoc Networking and Computing (2001).
8. P. Bergstrom, K. Driscoll and J. Kimball, Making home automation Communications secure, IEEE Computer (2001) .
9. M. Bellare, A. Desai, E. Jokipii and P. Rogaway, A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation, in: Symposium on Foundations of Computer Science (1997).
10. R.L. Rivest, the RC5 encryption algorithm, in: Workshop on Fast Software Encryption (1995) pp. 86.96.
11. N.Modadugu, D.Boneh and M. Kim, Generating RSA keys on a handheld using an untrusted server, RSA 2000.
12. G. Yuval, Reinventing the Travois: MAC in 30 ROM bytes, in: Workshop on Speed Software Encryption (1997).
13. S. Zhu, S. Setia and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks." in ACM Conference on Computer and Communications Security, S. Jajodia, V. Atluri ACM, 2003, pp. 62–72.
14. H. Chan, A. Perrig, and D. X. Song, "Random key pre-distribution schemes for sensor networks". in IEEE Symposium on Security and Privacy. IEEE Computer Society, 2003.
15. L.Eschenauer and V.Gligor, "A key-management scheme for distributed sensor networks." In ACM on Computer and Communications Security, V. Atluri, Ed. ACM, 2002.