A C-μTesla Protocol for Sensor Networks

Wen-Huei Chen
Department of Electronic Engineering, Fu Jen Catholic University
Taipei, Taiwan, R.O.C.
http://info.book678.idv.tw
mybook678@gmail.com

And Yu-Jen Chen
Department of Information Management, Chang Gung University
Taoyuan, Taiwan, R.O.C.
cyr@mail.cgu.edu.tw

A C-μTesla Protocol for Sensor Networks

Abstract

A sensor network has one or more base stations that talk to a large set of sensors, where the sensor life depends a small battery that consumes most power during communication. Before this network can be applied, many security problems must be solved, yet traditional security protocols usually need a lot of communication overhead. Recently, the μTesla protocol was proposed to address the authentication problem which ensures that no adversaries can impersonate the legal sender to control the sensors. The protocol adds a 24-bytes MAC (Message Authentication Code) to each 30-bytes message, where the MAC is based on the symmetric technique but can achieve the asymmetric property. The protocol is applied to a general case that a sensor can either compute the MAC to authenticate the message or just skip the MAC to see the message content.

In this paper, we propose a C-μTesla protocol for a specific case where a sensor wants to authenticate all messages. We reduce the overhead from 30 bytes to 4 bytes, by asking the sender to encrypt all messages without creating any MACs, so that a sensor authenticate messages through decryption. And each message adds a 4-bytes Cyclic Redundancy Check (CRC) before encryption, so that a modified encrypted message will not pass the CRC after decryption. Moreover, the C-μTesla protocol introduces a new confidentiality property that allows no adversaries to hear information of the sender through putting a fabricated sensor.

keywords. sensor network, security, protocol

1. Introduction

A *sensor network* has a large set of tiny sensors. These sensors form a *routing forest* with a *base station* at the root of each tree. Each sensor routes its sensed data to the base station which interfaces an outside network. The base station can either route commands to a sensor or broadcast them to all sensors. Some sensors can broadcast data to its nearby sensors, though it is within a limited range. In the future, such a network may be used for emergency response, energy management, medical monitoring, logistics and battlefield management. Currently, a prototype system has been implemented at UC Berkeley for heating and air-condition control. However, many feasible protocols must be proposed before the sensor network can be widely used [8].

Authentication protocols are very important in the sensor network [7]. Sensors depend on them to confirm that received messages are broadcasted from the legal sender (e.g., the base station); otherwise, an *adversary* may fabricate messages to control sensors. However, each sensor is a cheap and small-battery-powered device that has very limited computation and communication resources. For example, the sensor developed by the *SmartDust Project* of UC Berkeley has a 8-bit and 4-MHz CPU, 4.5 kbytes free memory space and 10 kbps bandwidth. As a result, each message cannot be authenticated by the *asymmetric digital signature* [7], because it needs a *communication overhead* of 50-1000 bytes per message as well as a large computation time for creating and verifying the signature [5]. Nor can it be authenticated by the symmetric method of Genario and Rohatgi [3] which reduces the computation time but keeps the communication overhead at 1000 bytes per message. In the sensor network, the life time of a sensor depends on a small battery where most power is consumed during communication. Thus, reducing the communication overhead is much more concerned than reducing the *computation time*. Rohatgi improves the method of [3] to use only 300 bytes per message [9], yet such an overhead is still unacceptable in the sensor network.

Recently, the *Tesla protocol* has been proposed for authenticating a sequence of messages with a light communication overhead [5]. The sender splits up the time into uniform intervals and prepares a key for each interval. This chain of keys (i.e., *key chain*) is constructed backwardly through a *one-way function* [2]. That is, the last key is picked randomly, and the j-th key is computed from the (j+1)-th key by a one-way function for each interval j. But the key chain is used forwardly. Initially, the sender broadcasts the first key that is digitally signed. With that signature, the receivers confirms that the first key comes from the true sender. Then, at each interval, the sender broadcasts a message with a *message authentication code* (*MAC*); this MAC is output from a *hash function* which inputs the message and the key at that interval [1]. As soon as a sensor knows that key, it decrypts that MAC to authenticate that the MAC comes from the true sender, which in turn confirm that the message itself comes from the true sender. Notice that an adversary may be monitoring the communication and uses the key to fabricate messages that control certain sensors. Thus, the key for each interval j is revealed at the j+r interval, where the r-intervals delay would make an adversary be unable to fabricate an MAC in time. This MAC is approximately 24 bytes per message of 30 bytes. However, this protocol broadcasts the first message with a digital signature that is still too expensive for the sensor network.

The Tesla protocol has been further improved into the μ*Tesla protocol* [6] by *bootstrapping* the first message through an authenticated channel, avoiding the expensive digital signature. Though the μTesla protocol is considered feasible in the sensor network, it is widely expected that its functions can be extended and its communication overhead can be further reduced so that a sensor battery can persist longer. In general, the protocol is applied to a general case that a sensor can either compute the MAC to authenticate the message or just skip the MAC to see the message content. In this paper, we propose a C-μTesla protocol for a specific case where a sensor wants to authenticate all messages , so that we can reduce the overhead per message from 30 bytes to 4 bytes.

We ask the sender to encrypt all messages without creating any MACs, so that a sensor authenticate the message by decrypting the encrypted message and see its content. Each message adds a 4-bytes Cyclic Redundancy Check(CRC) [4] before encryption, so that a fabricated message will not pass the CRC check after decryption. Moreover, the new protocol does not allow an adversary to put a fabricated sensor to hear any information from the sender while the earlier protocol allows. For a possible application, consider that the Environment Bureau installs a set of sensors in a factory to monitor the pollution. At a certain time, the bureau may remotely control the base station to send commands to these sensors, which then return the sensed information. Certainly, the bureau does not want the adversary (who owns that factory) to know that commands and put off the pollution then.

In Section 2, we review the μTesla Protocol. In Section 3, we propose the C-μTesla protocol for the first confidentiality problem. In Section 4, our conclusions are presented.

2. The μTesla Protocol

Consider an example sensor network which has a sender (S) and four receivers (R1, R2, R3 and R4.) The sender is trusted by all receivers, those which may not trust each other, and each receiver trusts itself. The sender wants to sequentially broadcast four messages m_1, m_2, m_3 and m_4 to the receivers. Thus, it prepares a key chain [k_0, k_1, k_2, k_3, k_4]. It picks k_4 randomly and computes $k_3 = f(k_4)$, $k_2 = f(k_3)$, $k_1 = f(k_2)$, and $k_0= f(k_1)$ where f is a one-way function. Receivers R1, R2, R3 and R4 share distinct Secret keys X1, X2, X3, X4 with the receiver respectively.

The μTesla protocol has two phases. In the first (i.e., bootstrap) phase, the sender will send the first key to each receiver personally, through the encryption of the Secret key shared between them. Consider the example of receiver R2 . R2 first sends a Nonce to the sender. Then, the sender returns "k_0, MAC(k_0||Nonce, X2)" to R2, where "||" connects k0 and Nonce, and X2 is the Secret key between the receiver and the sender. R2 confirms that the first key k_0 comes from the true sender, because it is encrypted by the Secret key X2 which is known only to R2 and the sender. The Nonce is used to defend against an adversary from resending an earlier message.

Formally, a receiver Rj may bootstrap at any interval i (thus the first key is called k_i) and needs some related information [6] (namely, T) to join the broadcasting group, thus the protocol steps are described as follows:

Rj--> S: Nonce

S --> Rj: k_i ||T , MAC(k_i || T || Nonce, Xj)

In the second (i.e., broadcast) phase, the sender will broadcast authenticated messages to all receivers. The sender splits the time into six intervals and broadcasts the following messages:

Interval 1:	m_1, MAC(m_1, k_1)	Interval 2:	m_2, MAC(m_2, k_2)
Interval 3:	m_3, MAC(m_3, k_3), k_1	Interval 4:	m_4, MAC(m_4, k_4), k_2
Interval 5:	k_3	Interval 6:	k_4

At interval j (where j = 1, 2, 3, 4), the sender broadcasts m_j, MAC(m_j, k_j) to all receivers, where m_j is the message and MAC(m_j, k_j) is a message authentication code of m_j based on key k_j. At interval j (where j =3, 4, 5, 6), the sender broadcast key k_{j-2} to all receivers. Notice that each key is revealed two intervals later, where this delay is assumed to make an adversary be unable to fabricate the MAC in time.

The messages at each interval promises the authentication and integrity properties. At intervals 1 and 2, each receivers store the received message because they have no keys to verify it. At interval 3, each receiver will see key k_1. The receiver has already known that k_0 is used by the sender in the bootstrap phase. It then verifies that $k_0 = f(k_1)$ by the one-way function f. This one-way function allows a receiver to verify that k_1 follow k_0 in the key chain, but it does not allow any adversary to compute k_1 from k_0. If $k_0 = f(k_1)$, MAC(m_1, k_1) will be authenticated to come from the true sender because k_0 comes from the true sender. The authentication property has thus been established. Each sensor also checks that MAC(m_1, k_1) is output from a hash function which inputs m_1 and k_1. If m_1 is altered either by the adversary or error during transmission, such a check will fail. The integrity property has thus been established. By going through intervals 4, 5 and 6, all messages will be authenticated.

Formally, a sender may reveal the key after d intervals, thus the protocol step for interval j is described as follows:

S -> R: m_j, MAC(m_j, k_j), k_{j-d}

but the item "k_{j-d}" is missed in the first d intervals and items "m_j, MAC(m_j, k_j)" are missed in the last d intervals.

3. The C-μTesla Protocol

In Section 2, we have briefly described the μTesla Protocol which contains the bootstrap and broadcast phases. In this section, we will modify the protocol in both phases so that it can ensure that no adversaries can hear information from the sender by putting a fabricated receiver. And we will show that it saves the communication overhead per message from 30 bytes to 4 bytes.

The μTesla protocol cannot defend against this attack in both phases. As shown in Section 2, the sender sends "k_i || T ", "MAC(k_i || X || Nonce)" to each receiver in the bootstrap phase. At each interval of the broadcast phase, the sender broadcasts "m_j, MAC(m_j, k_j), k_{j-d}" to all receivers. Thus, an adversary may put a fabricated receiver (or an instrument) to receive the wireless signal and then know k_0, m_j and k_{j-d}. As mentioned in Section 1, knowing both information may corrupt the whole system.

We first modify the the bootstrap phase. Formally, a receiver R_j may bootstrap at any interval i (thus the first key is k_i) and needs some related information [6] (namely, T) to join the group. The receiver also needs k_0 for decryption. Thus the new protocol steps are described as follows:

Rj--> S: Nonce

S --> Rj: f(k_0 ||k_i ||T||Nonce||CRC(k_0 ||k_i ||T||Nonce), X_j) where f is a symmetric encryption function and X_j is the Secret key between R_j and the sender.

That is, we encrypt the whole message by a Secret key X_j, so that the adversary cannot see the message content. The receiver encrypts the ciphertext by the Secret key X_j and confirms that it is sent from the true sender; no ones other than itself and the sender knows that key. The authentication property has thus been achieved. The message content includes key k_i, k_0 , related information T and the CRC of the former three items. The CRC is used to ensure that the message encrypted content will not be altered during the transmission. The integrity property has thus been achieved.

Second, we modify the protocol steps of the broadcast phase. Consider that a sender may reveal the key d intervals later, the protocol step for interval j is described as follows:

S -> R: $f(m_j||CRC(m_j), k_j)$, $f(k_{j-d}, k_0)$

where the item $f(k_{j-d}, k_0)$ is missed in the first d intervals and the item $f(m_j||cheksum(m_j), k_j)$ is missed in the last d intervals.

Consider the first item $f(m_j||CRC(m_j), k_j)$. We concatenate message m_j with its CRC to preserve the integrity property of the message content. Then, we encrypt the whole message by key k_j at interval j. Notice that k_j is chained from key k_i which has already been authenticated, we can authenticate that the message comes from the true sender. This encryption also ensures that the adversary could not install a fabricated receiver to see the message content, because he does not know k_j. Consider the second item $f(k_{j-d}, k_0)$. We encrypt the key by key k_0 of the key chain; this ensures that the key will not be revealed to any receiver outside the current broadcasting group.

The example of the earlier section for the broadcast phase is modified into:

Interval 1: $f(m_1||CRC(m_1), k_1)$	Interval 2: $f(m_2||CRC(m_2), k_2)$
Interval 3: $f(m_3||CRC(m_3), k_3)$, $f(k_1, k_0)$	Interval 4: $f(m_4||CRC(m_4), k_4)$, $f(k_2, k_0)$
Interval 5: $f(k_3, k_0)$	Interval 6: $f(k_4, k_0)$

Each message is around 30 bytes, and we use a CRC of 4 bytes (yet its size is changable depending upon how severe the environment is.) Thus, the new message is 34 bytes, and the encrypted message remains the same length. We have reduced the communication overhead to 4 bytes per message.

4. Conclusions

In this paper, we have modified the μTesla protocol for overcoming a new confidentiality problem and an authentication problem using an overhead of 4 bytes per message, in contrast to the earlier one which uses 24 bytes per message that overcomes only the authentication problem.

Currently, our mechanisms are used only for the base station broadcasting, but we are exploring their use for sensor broadcasting [10].

References

[1] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *IEEE INFOCOM*, 1999, March 1999.

[2] M. Jakobsson, "Fractal hash sequence representation and traversal." *IEEE Symposium on Information Theory*, July 2002, pp. 437-444.

[3] R. Gennaro and P. Rohatgi, "How to sign digital streams, in Advances in Cryptology - Crypto' 97," *Lecture Notes in Computer Science*, Vol. 1294, 1997, pp. 180-197.

[4] S. Lin and D. J. Costello, *Error Control Coding: Foundementals and Applications*, Prentice Hall, 2nd Edition, 2004.

[5] A. Perrig, R. Canetti, J. D. Tygar and D. Song, "The Tesla Broadcast Authentication Protocol," *RSA Crypto Bytes*, Summer, 2002.

[6] A. Perrig, R. Szeczyk, J. D. Tygar, V. Wen and D. E. Culler, "SPINS: Security Protocols for Sensor networks," *Wireless Networks*, Vol. 8, 2002, pp. 521-534.

[7] C. P. Pfleeger, *Security in Computing, 2nd Edition*, Prentice-Hall Inc., New Jersey, U.S.A., 1997.

[8] K. S. J, Pister, J. M. Kahn and B. E. Boser, "Smart dust: wireless networks of millimeter-scale sensor nodes," *http://robotics.eecs.berkeley.edu/~pister/SmartDust/,* 1999.

[9] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," *ACM Conference on Computer and Communication Security*, Vol. 21, No.2, 1978, pp. 120-126.

[10] W. H. Chen and Y. R. Chen, "A bootstrapping scheme for inter-sensor authentication within sensor networks," IEEE Communication Letters, Vol. 9, No. 10, 2005.