Cooperative Public Key Authentication Protocol in Wireless Sensor Network

DaeHun Nyang and Abedelaziz Mohaisen*

Information Security Research Laboratory *of* InHa University
Inofmration Technology & Telecommunications Graduate School
253 YongHyun-dong, Nam-gu, Incheon 402-751, Korea
nyang@inha.ac.kr, asm@seclab.inha.ac.kr
http://seclab.inha.ac.kr

Abstract. Recent measurements for Public Key Cryptography (PKC) protocols on 8-bit wireless sensor nodes showed optimistic results. It has been shown that Elliptic Curve Cryptography (ECC) is quite applicable to WSN. Still, PKC is much expensive in terms of computation and memory compared by the Symmetric Key Cryptography (SKC). In addition, in PKC, each public key needs to be authenticated before it's used. We believe that sooner or later, PKC will be widely deployed in WSN. Therefore, we present a cooperative distributed public key authentication scheme that does not require any cryptographic overhead. In our scheme, each node is let to store a few number of hashed keys for other nodes. When a public key authentication is required, nodes who store this key help in authenticating it in a distributed and cooperative way. We consider the constrained resources of the sensor node. Additionally, we extend our scheme to fit with small range of authentication error.

Keywords: Public Key Authentication, Cooperative Protocol, Voting.

1 Introduction

Wireless Sensor Network (WSN) is a resulting successful mergence of different technologies. Advancements in different fields including microelectronics, semiconductors, networking, signal processing and many others led to this invention. The WSN consists of large number of inexpensive and resources constrained sensor nodes which work in a cooperative data-forwarding method to perform some sensing tasks. Sensors communicate in peer-to-peer mechanism in an open air environments that enables any man-in-the-middle (MITM), Sybil or even node replication attacks [7].

The growth of WSN applications which involved many ranging sensitive environments brought the necessity to provide security rules to guard the communication traffic between the sensor nodes. For more than five years, security research in WSN was limited to Symmetric Key Cryptography (SKC) protocols that require a key distribution. Notwithstanding of SKC limitations on the

* This work was supported by INHA University Research Grant.

J. Ma et al. (Eds.): UIC 2006, LNCS 4159, pp. 864–873, 2006.

side of the connectivity and resiliency, it showed a fastness and resources usage feasibility that is suitable for resources constrained sensor nodes. For the same reason, PKC wasn't used or even researched in WSN. Recently, some works on the PKC protocols (e.g. ECC, RSA) evaluation and efficiency measurements on sensor node platform showed optimistic results[10,11,12,13]. Based on recent researches, we believe that the advancement of processors equipped to the sensor nodes and the improvement of the PKC protocols will make it possible to take the advantage of the PKC into the WSN security. Once PKC is used in WSN, the resiliency and connectivity problems will not exist anymore since the PKC behaves perfectly in both sides (i.e. connectivity and resiliency equal 100%). [1] Recalling that, sooner or later PKC will be deployed in WSN. Therefore, Public Key Authentication problem should be solved. In this paper, we state the problem, briefly present the related work, and based on this we present our solution.

1.1 Problem Statment

PKC (*e.g.* RSA[9], ECC[16]) operates in a way that no need for a node to know other node's private key when encrypting a message for it. However, before the encryption, a node i requires node's j public key to encrypt a message for it. The most significant problem therefore is to determine whether a received key is really belonging to node j or not. One possible naive solution is to store all of the network public keys in each node that requires $(N-1) \times P$ bits/node for a network and keys of size N, P respectively, when two key authentication is required, the received key is simply searched in the keys set of the receiver node. Another solution is to store the keys corresponding hashed values which cost $(N-1) \times L$ bits/node where L is the length of the hashed key. It's clear that both of the solutions are inefficient in both memory and computation.

1.2 Related Work: Symmetric Key Cryptography (SKC)

There are intensive studies on SKC in WSN. Many schemes to secure WSN were developed. Perrig *et al.* developed SPINS that uses μTesla as a building block [8]. Additionally, for efficient SK management, many techniques and mechanisms were developed. Eschenauer-Gligor[5] proposed a key distribution scheme based on the random graph theory. In this scheme, a pair of node can establish a key when they have a shared key in their key ring. Chan et al. proposed q-composite scheme [3] based on [5]. In q-composite, a pair of nodes can communicate only they share q number of common keys in their key's ring. Du *et al.* also developed several schemes based on [1]. In [4], a similar results as in [6] was independently achieved to distribute pairwise keys for WSN. Liu *et al* [6] developed several schemes based on the Symmetric Bivariate Polynomial Protocol [2]. Moreover, different schemes based on [4,6,1,2] was developed using the deployment knowledge to reduce the used resources.

[1] Currently, BTnode[17] is equipped by ATMega128L which operates at 8 MHz and Crossbow mote is equipped by ATmega128 which dually operates at 8,16 MHz[18].

1.3 Related Work: Public Key Cryptography (PKC)

The recent results of the PKC protocols on sensor nodes showed relevant acceptable efficiency. In Gura *et al* work [10], practical measurements for ECC[16] and RSA[9] signatures verification was obtained. It was shown that ECC signature verification consumes 1.62 *ms* on the 8-bit ATmega128 processor which operates at 8 MHz. An extension of [10] on PKC protocols' enery consumption was developed in [11]. Watro *et al.* developed another limited PKC architecture with a practical evaluation of consumed resources per sensor node TinyPK [12]. Key distribution in TinyOS based on Elliptic Curve Cryptography (ECC) with real measurement and evaluation was also considered in Malan's *et al* work [13].

To the best of our knowledge, the public key authentication in WSN has been studied in a unique scheme by Du *et al*[15]. In this scheme, Merkle hash tree is used. Merkle hash tree is a binary tree of N leaves that represent the different node's hashed keys. Each internal parent till the root stores a hashed value of its corresponding children data block (*i.e.* using SHA1 of 160-bit). In [15], each node stores $log_2(N) + 1$ hashed values (which are selected from the node to the root of the tree). Once Bob's key authentication is required by Alice, Alice receives Bob's hashed values and public key. Locally, Alice perform SHA1 hashing for the received value and compares the resulting root with his own root. Depending on the equality of the resulting hashed value and the store root, Alice can determine whether Bob's key is real or not. Using the deployment knowledge, Merkle tree is split into sub trees (*i.e.* Merkle forest) to reduce the used memory per node. Therefore, each split reduces the used memory by one key.

1.4 Our Contribution

In this paper, we present a novel scheme for authenticating the public key in WSN. Our scheme uses distributed and cooperative mechanism to perform such a need. Our contribution relies in that we don't use any cryptographic operations to authenticate a key. In addition, each node stores a limited number of hashed keys for a set of different nodes that limits the used memory.

2 Network Model: Assumptions

A.1 Our network model assumes that there are hundreds of sensor nodes within the same radio range. Note that, the network size is dependent on the MAC layer and not Physical layer, which rationalizes the assumption.

A.2 The different nodes in the network can overhear every traffic between any pair of nodes. In the normal case, this overhearing is necessary to make the nodes decide whether they have to rely a frame or not even when it is not for them.

A.3 Static data is deliverable from one node to another while attackers reside in the same geographical area or the same radio coverage range.

A.4 Interception and modification of frame(s) are possible during one hope transmission only when the attacker knows who is about to send.

A.5 Attackers have the ability to inject forged frames.

3 Basic Protocol

To reduce the used resources in the sensor node in our protocol and to resist the flooding attack as well, some of the authentication decision is held in MAC layer. To enable such a decision, the following is a list of our modification on the MAC frame:

- Since we use specific frames for authentication, we added one bit flag **AC** to discriminate the authentication frames from the normal frames. Additionally, this bit will be used as a check bit for discarding any extra frames more than the required for the authentication (*i.e.* $> k$).
- The authentication frame can be authentication request frame, authentication response frame from the concerned node, or authentication response frame from an assisting node. Therefore, we added two bits (Authentication Request Reponse bits **ARR**) to discriminate those different frames. This addition is required for the flooding attacks resistance in the MAC layer.

Table 1 shows the corresponding meaning of initially assigned bits for **AC** and **ARR** bits. Making this clear, our protocol then consists of two phases: initialization and online procedure. In the following, two sections we will present it.

3.1 Protocol Initialization

In this protocol, we assume that: Each sensor node has the ability to do a random coin tossing with probability for head p. Probability p determines whether a node will assist in a key authentication or not.

Installation of PK information in the sensor nodes: For each sensor node i, the following procedure is performed for all the nodes in the network.

1. Security Authority (SA) randomly selects k number of nodes.
2. SA installs the public key information of a node i (K_i) to the k different sensors; where $K_i = hash$(Node i public key|Node i ID).

3.2 Authentication Protocol's Online Procedure

Eventually, a node j wants to get K_i which is the public key information of node i. At that time, the following procedure will be performed:

1. Sensor node j sends a request frame to sensor i informing that it needs K_i. Note that all the nodes in the network can overhear this request under the second assumption of section 2.
2. Not only sensor i, but also every sensor node that has the key K_i sends it to j. Different from the other nodes, sensor node i actually sends both K_i and its own public key.
3. As soon as node j receives the first response, it begins to count the number of the received response frames up to the threshold number of response. After this threshold, node j discards every incoming frame for the same key authentication.

4. Let k' be the number of incoming responses to node j. Defining e as the error bound (*i.e.* deviation from original k), the following will be performed:
   (a) If $|k' - k| \leq e$ then:
      i. Node j performs a majority voting to decide K_i.
      ii. Every node that has K_i must delete it from its memory.
   (b) If $|k' - k| > e$, that indicates a probability of an attack, and sensor node will discard the received frames to perform the request again.
5. Every other sensor node must perform the step 4 parallely, which is possible under the consumption **A.2**. Only if in case 3.(a), do the following:
   (a) Tosses a coin.
   (b) Only if the result is head, then store K_i. Note that, if we consider p as the probability of head (which is the same probability of keeping the K_i), then, $p = \frac{k}{N}$, where N is the number of nodes in the network and k is the number of nodes that will keep K_i on average.

3.3 Hurdle: Tossing Deviation Control

Even if we assign the tossing probability $p = k/N$, we can only guarantee that the number of the nodes that hold the public key information K_i of a node i is k "at average". Recall that, the probability of such process has the binomial distribution. Given the network size N and k at average, we can obtain the standard deviation of tossing probability is $(\sigma) = \sqrt{k(1 - (k/N))}$. An illustration of the deviation is in Fig 1(a). Even if we reduce the average of the authenticating nodes k, the standard deviation is little bit large for a small sized network. So, the receiver might be confused whether the difference is from attacker or from the large tossing deviation and communication noise. In the case of a large network size (say 1000 nodes), as in Figure 1(b), the authentication is held successfully since the deviation is bounded by small value compared to the size authentication assisting group k.

4 Solution: Efficient Protocol

To overcome the high deviation in a small size networks, we propose two modified versions of our basic protocol. The first version guarantees optimal security for limited number of authentication rounds (This security is guaranteed under the condition that no key is authenticated by the same assisting node twice. That means even if an attacker could monitor the network traffic and know the authentication set for the current key authentication, this type of information is not valid any more after the current authentication round. The second modified version is long-living with limited security feature and rounds traceability scheme (reversed situation of the first protocol, since with some probability for success, given enough information about c authentication rounds for a given key, it's more possible for the attacker to have successful chances for alterning the groups authentication replies.

4.1 *c*-Rounds Protocol: Optimal Security with Limited Life

The sensor network is very static. Thus, we can pre-compute the exact public key information distribution before deployment. This protocol has two phases of initialization and online protocol as follows:

Initialization Phase: In the initialization phase, for each sensor node with public key information K_i, the following is performed:

1. Security authority randomly picks c groups of node of size k from WSN.
2. The key information is paired as $\langle r, P_i \rangle$, where r is a serial and $0 < r \leq c$. The different pairs are loaded to the corresponding group of nodes.

Online Authentication Phase: In the online authentication phase, the following is performed:

1. A node j requests public key information from node i. Initially, j sends the request including a serial that express the authentication round (*i.e.* $\langle 1, req \rangle$ at the first time K_i authentication is required).
2. Not only the node i, but also all of the other node that contains the pair $\langle r, K_i \rangle$ respond the request. All of the nodes participating in the authentication have listeners on the traffic so that they can receive all copies of K_i.
3. As soon as node j receives the first authentication response from a node, it starts counting the received key information frames till a threshold number. After that, node j begins discarding any incoming extra frames.
4. The majority voting is performed at node j and all of the other authentication group to decide the acceptance of K_i.
5. If the majority voting admits the received key K_i, all copies of K_i in the authentication group with the current authentication round are removed from each sensor node memory.
6. The authentication round counter is increased by one so that the next authentication group assist in authenticating K_i in the next round.

Note that, the modified protocol does not require any coin tossing to predict the number of the next authentication round, but it relies on the predistribution of the public key iformation which guarantees a *zero deviation*. In addition, this protocol is limited to r authentication round for any key K_i. Also, when the key authentication proccess is performed, each node has the ability to recognize every incoming frame based on the **AC** and **ARR** bits values. Based on the current status of the authentication proccess, each node also can determine whether to pass the received authentication frame to the upper layer or not.

4.2 Long Living Protocol

The noticeable problem of the the above protocol is the limited life. In some intelligent attacks, it's possible to inject many faked keys for just limiting the life of the protocol. In the following, we extend the protocol life giving up a small fraction of the overall security. Note that, the initialization phase is typically the same like the initialization of the c-rounds protocol explained in 4.1.

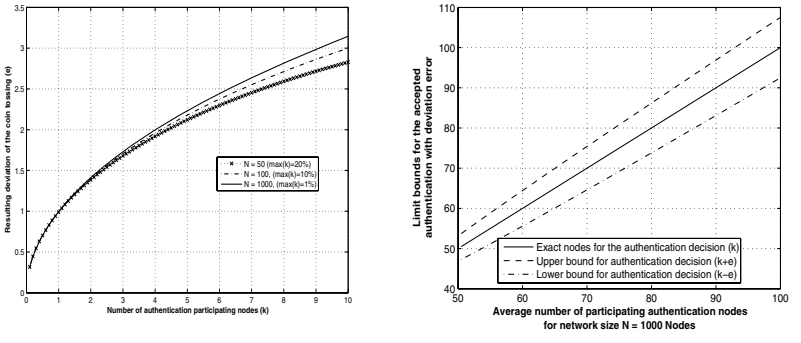
Online Authentication Phase

1. Once node K_i authentication is required by node j, node j firstly sends a request as $\langle r, req \rangle$ where r is a random integer as $0 < r \leq c$.
2. Not only node i, but all nodes holding the K_i and the random round number as a pair answer the request. Exceptionally, node i sends its own public key as well.
3. As in the early protocols, node i starts couting the number of the received K_i copies till a threshold number of frames. After the threshold value, j begins rejecting any addition frames for K_i authentication.
4. After the majority voting is performed, none of the round's nodes contents is deleted.
5. Again, an authentication is required for the same key, r is picked randomly and the request is performed.
6. The attackers has the ability to perform an attack, if and only if, he knows r and the authentication group in advance (say $\tau-$time before) that enables him to alter the authentication contents.

5 Evaluation

5.1 Authentication Decision and Tradeoffs

In our basic protocol, the voting decision can be performed as long as there are limited number of the forged messages (*i.e.* The attacker has a chance to deliver forged message with the timer of the authentication). Figure 1(a) shows the deviation for different network size when using limited number of authenticating nodes (*i.e.* up to 10 nodes per group). Figure 1(b) shows the upper and lower bounds for successful authentication when using a group of 5% to 10% of the overall network size.



(a) Tossing Deviation (b) Authentication Bounds

Fig. 1. The tossing deviation and the authentication upper and lower bounds for the actual required number of trusted nodes to authenticate a key

In the other hand, the $c-$ROUNDS and the LONG LIVING protocols don't generate any type of deviation, because the group of the different authentication proccesses is determined in advance before the installation of the protocol.

5.2 Overhead Evaluation and Comparison with Other Works

The resources overhead generated by our protocol is analyzed in terms of the memory, communication and computation. In the following we detail each of the required overhead.

- **Memory Overhead:** recall that each node can assist in k nodes' authentication. In addition, each node has r authentication rounds. Therefore, the required memory per node is $k \times r \times L$ bit, where L is the size of the hashed key in bit (*e.g.* SHA1 generates 160 bit hashes).
- **Communication Overhead** CT_{OH}: cosidering the distributed performance to perform a majority voting for a key, the required commuication is to send/receive k keys' hashes.
- **Computation Overhead:** using 8-bit processor (*i.e.* ATmega128L, $f = 8$ MHz) and 160 bit hashed keys, the required computation per authenticating one key is $(2.5k)\mu sec$. In general, assuming that W is the proccessor word size, L is the hashed key size and f is the frequency of the processor in Hz, the required computation in seconds is $CM_{OH} = \frac{L}{W} \times \frac{k}{f}$.

A detailed comparison of the consumed resources for the authentication is shown in Table 2.

5.3 Security Analysis

The security in our protocol relies on performing the authentication successfully without making the atackers affect the decision of the voting. In our first, second

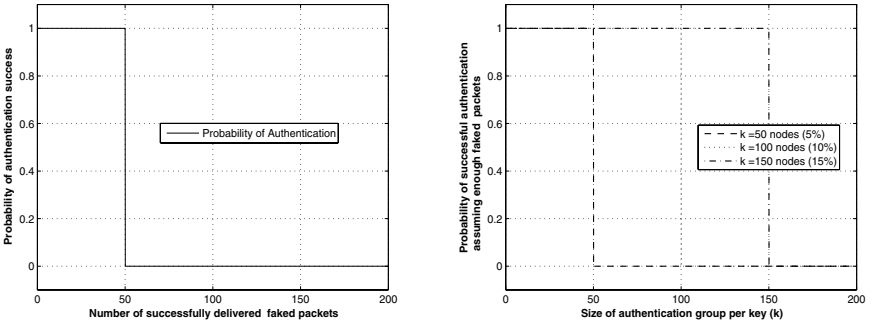
Table 1. Authentication (ARR) and (AC) flags indication

Feild value	**Stands for**
ARR(00)	Authentication request frame
ARR(01)	Authentication response frame from concerned node
ARR(10)	Authentication response frame from an assisting node
AC$(0, 1)$	Normal, Authentication frame respectively

Table 2. Resources Comparison between our scheme, RSA[9], ECC[16] and Du *et al* Scheme [15]. CT_{OH}, CM_{OH} stand for communication and computation overhead.

	Key/Hash size (bit)	CT_{OH} (bit)	CM_{OH} (ms)
RSA[9]	1024	1024	430
ECC[16]	160	320	1620
Du *et al.*[15]	160	$160k$	$7.2k$
Our scheme	160	$160k$	$2.5 \times 10^{-3}k$

and third protocols, this goes fine for a threshold number of injected faked keys for one key authentication. Assuming that the number of the nodes that assists in performing the authentication voting is k with an error range at e, each node requires $k - e/2$ number of faked keys to redirect the result of authentication. Even though, using our modification for the MAC frame and the threshold for accepting the required $k \pm e$ keys for authenticating a concerned node's key K_j will make it hard to deliver more than the k attacker's faked keys. For illustration, Figure 2(a) shows the behavior of the authentication chances per node for single attacker. Figure 2(b) shows the behavior when different k's are used.



(a) Single Node/Attacker (b) Different k's

Fig. 2. For 1000 nodes, (a) Single attacker behavior is simliar to the multiple when the number of attackers and nodes is equal. (b) different thresholds to authenticate a key.

6 Conclusion and Further Work

We introduced a novel distributed and cooperative protocol to authenticate public keys in WSN. Our protocol does not require any computational cryptographic overhead. In addition, our protocol considers the different constrained-resources of the sensor node. Even in the our basic protocol, the authentication can be held successfully with limited deviation. Our protocol is designed for one hop authentication. In the further work, we will investigate its extenssion to work for multi-hop. In addition, we will study the usage of parameters except of the deviation to perform authentication voting. More detailed mathematical evaluation for the effect of the attackers will be studied.

References

1. Blom, R.: An optimal class of symmetric key generation systems, Advances in Cryptography, Proc. EUROCRYPT 84 , LNCS-209, pp: 335-338, 1985.
2. Blundo, C., DE Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., and Yung, M.: Perfectly secure key distribution for dynamic conferences, CRYPTO '92, LNCS-740, pp: 471-486, 1993.

3. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks, IEEE SS&P, pp. 197-213, May 2003.
4. Du, W., Deng, J., Han, Y. S., and Varshney, P.: A pairwise key pre-distribution scheme for wireless sensor networks, ACM CCS'03, pp. 42-51, 2003.
5. Eschenauer, L., Gligor, V. D.: A key management scheme for distributed sensor networks, ACM CCS'02, pp. 41-47, 2002
6. Liu, D., Ning, P.: Establishing Pairwise keys in distributed sensor networks, ACM CCS'03, pp. 52-61, 2003.
7. Parno, B., Perrig, A., and Gligor V.: Distributed Detection of Node Replication Attacks in Sensor Networks, IEEE SS&P'05, May 2005.
8. Perrig, A., Szewczyk, R., Wen, V., Culler, D. E., Tygar, J. D.: SPINS: security protocols for sensor networks, MOBICOM'01, pp. 189-199, 2001.
9. Rivest, R. L., Shamir, A., Adleman, L. M.: A method for obtaining digital signatures and PK cryptosystems, Comm. of the ACM, 21(2): pp. 120-126, 1978.
10. Gura N., Patel A., Wander A., Eberle A., Shantz S. C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs, CHES 2004: 119-132
11. Wander A., Gura N., Eberle H., Gupta V., Shantz S.C.: Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks, PerCom'05, pp. 324-328.
12. Watro R.J., Kong D., Cuti S.F., Gardiner Ch., Lynn Ch., Kruus P.: TinyPK: securing sensor networks with public key technology, SASN'04, 59-64, 10-2004.
13. Malan D.J., Welsh A., Smith M.D.: A Public-Key Infrastructure for Key Distribution in TinyOS Based on ECC, IEEE SECON'04, pp. 71-80.
14. Pietro R.D., Law Y.W., Etalle S., Hartel P.H., Havinga P.: State of the Art in Security of Wireless Sensor Networks, IEEE Computer, 35(10): pp. 1-10.
15. Du W., Wang R., and Ning P.: An Efficient Scheme for Authenticating Public Keys in Sensor Networks. Sixth ACM MobiHoc, pp: 58-67.
16. Koblitz N., Menezes A., Vanstone S.: The State of Elliptic Curve Cryptography, Designs, Codes and Cryptography, 19, 173-193 (2000).
17. BTnode Project - ETH-Zurich: `http://www.btnode.ethz.ch/`
18. Crossbow Tech. Inc. Wireless Sensor Networks: `http://www.xbow.com/`