International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878
Volume 2 Issue 2, February 2014

A Protocol for Confidential and Distributed Reprogramming in Wireless Sensor Networks

M Surya

Anna University, MNSK College of Engineering,
Arandhangi Road, Pudukkottai, India

Abstract: ***A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure. and to cooperatively pass their data through the network to a main location. The process of uploading new code or changing the functionality of existing code is called Wireless reprogramming. For security reasons, every code update must be authenticated to prevent an adversary from installing malicious code in the network. All existing reprogramming protocols are based on the centralized approach in which only the base station has the authority to initiate reprogramming. This project develops A Protocol for confidential and distributed reprogramming in WSN named PCDR. The protocol uses identity-based cryptography to secure the reprogramming and to reduce the communication and storage requirements of each node.***

Keywords: Authentication, Reprogramming, Security

1. Introduction

When Wireless Sensor Networks (WSNs) may be deployed for long periods of time during which the requirements from the network owner and users or the environment in which the nodes are deployed may change. The change may necessitate uploading a new code image or retasking the existing code with different sets of parameters [1].We refer to both of these activities as reprogramming. As a WSN is usually deployed in hostile environments, secure reprogramming is and will continue to be a major concern. Several reprogramming protocols have been proposed to propagate new code images1 in WSNs [2].Among these protocols, Deluge is generally regarded as the state of the art and included in TinyOS distributions. It uses an epidemic protocol for efficient advertisement of metadata and spatial multiplexing for efficient propagation of code images. However, since the design of Deluge did not take security into consideration, there have been several extensions to Deluge to provide security protection for reprogramming. Among them, Seluge enjoys both strong security and high efficiency.

However, all existing reprogramming protocols are based on [3]-[5] the centralized approach in which only the base station has the authority to reprogram the sensor nodes. When the base station wants to disseminate a new code image to certain sensor nodes, it transmits the signed code image to those nodes via multihop routing, and those nodes only accept the code image signed by it. Unfortunately, the centralized approach is vulnerable to the single point of failure and not reliable because reprogramming becomes impossible when the base station fails or when some nodes lose connections to the base station. Also, it is inefficient, weakly scalable, and vulnerable to potential attacks along the long communication path. The base station has to be online and accessible to any user at any time during the network operation. Even worse, there are some WSNs that do not have any base station.

2. Design Considerations of Distributed Reprogramming

As shown in Fig. 1, a centralized reprogramming protocol involves only two kinds of participants, the base station and all sensor nodes. Only the base station can reprogram sensor nodes. Different from the centralized approach, a distributed reprogramming protocol consists of three kinds of participants, the network owner, authorized network users, and all sensor nodes. Here, the network owner can be offline. Also, after the users register to the owner, they can enter the WSN and then have redefined privileges to reprogram the sensor nodes without involving the owner. To provide secure and distributed reprogramming, a naïve solution is to pre-equip each sensor node with multiple public key/reprogramming-privilege pairs, each of which corresponds to one authorized user. This scheme allows a network user to sign a program image with his private key such that each sensor node can verify whether the program image originates from an authorized user.

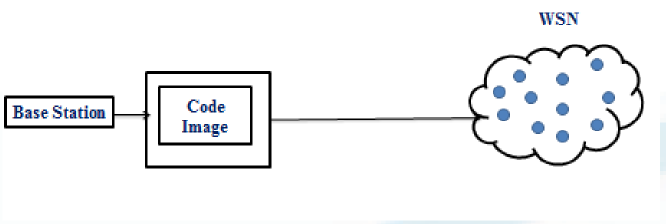

Figure 1: Centralized Approach

However, this solution is not applicable to WSNs due to the following facts. First, resource constraints on sensor nodes often make it undesirable to implement such an expensive algorithm. For the RSA-1024 public-key cryptosystem(1024-b keys)[6], the length of each public key is more than 1026 b. Additionally, for the ECC-160 [29] public-key cryptosystem (160-b keys),the length of each public key is 1120 b. Assuming that the length of reprogramming privilege is 32 B and either RSA-1024 or ECC-160 is used, the length of each public-key/reprogramming-privilege pair is more than 160 B. This means that not too many public key/reprogramming privilege pairs can be stored in a sensor node.

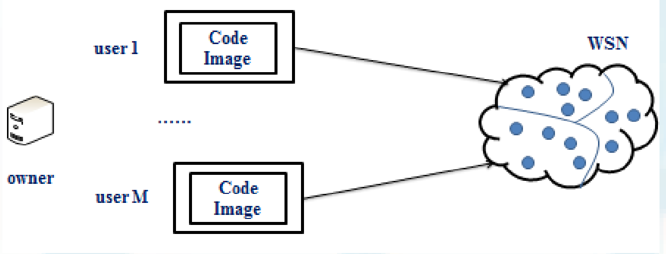

Figure 2: Distributed Approach

We consider the commonly used MicaZ platform as an example. The 512-kB Flash memory is not suitable for storing these parameters, since it is much slower and more energy consuming than ROM. On the other hand, MicaZ platform only has 128-kB ROM, while most of ROM needs to be used for storing program. In this case, not too many users can be supported. Second, it is clear that the network owner has no ability to predefine the reprogramming privileges of the new joining users before the WSN deployment. Once a new user registers to the network owner, the owner needs to sign a new public-key/reprogramming-privilege pair and then broadcasts it to all sensor nodes. Obviously, this behaviour is not efficient and weakly scalable, particularly in large scale WSNs. We naturally shift our attention to certificate-based approach (CBA) [7].

A more suitable approach is for each authorized user to send a new program image to the nodes through a standard group signature technique. A group signature scheme [8] allows one member of the group to sign a message such that any verifier can verify that the message originated from a group member. Thus, only the group public key is preloaded onto each sensor node. Meanwhile, any group signature can be "opened" by the group manager (i.e., the network owner) to reveal unambiguously the identity of the actual signer. Unfortunately, a group signature algorithm does not support different levels of user authorities. That is, the network owner cannot specify a reprogramming privilege for each user.

3. PCDR: The Protocol

PCDR consists of three phases: system initialization, user preprocessing, and sensor node verification. In the system initialization phase, the network owner creates its public and private keys and then assigns the reprogramming privilege and the corresponding private key to the authorized users. Only the system public parameters from the network owner are loaded on each sensor node before deployment. In the user preprocessing phase, if a network user enters the WSN and has a new program image, he will need to construct the reprogramming packets and then send them to the sensor nodes. In the sensor node verification phase, if the packet verification passes, then the nodes accept the program image. The detailed description of each phase is as follows.

3.1 System Initialization

The network owner creates its public and private keys and then assigns the reprogramming privilege and the corresponding private key to the authorized user(s). Only the system public parameters from the network owner are loaded on each sensor node before deployment. Title of the paper is centered 17.8 mm (0.67") below the top of the page in 24 point font. Right below the title (separated by single line spacing) are the names of the authors.

3.2 User Preprocessing

In the user preprocessing phase, if a network user enters the WSN and has a new program image, he will need to construct the reprogramming packets and then send them to the sensor nodes. Assume that user U_j enters the WSN and has a new program image. U_j takes the following actions.

- U_j partitions the program image to Y fixed-size pages, denoted as page 1 through page Y . U_j splits page i ($1 \leq i \leq Y$) into N fixed-size packets, denoted as $Pkt_{i,1}$ through Pkt_i , N . The hash value of each packet in page Y is appended to the corresponding packet in page $Y - 1$.Note that, in order to support a variety of applications, the formats and lengths of the message m and the reprogramming privilege Pri_j in PCDR should be set according to the specified application scenario.

- With the private key SK_j, U_j can compute the signature σ_j of the message m, where $\sigma j = H_2(m) \cdot SK_j$.
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. Thi U_j transmits to the targeted nodes the signature message {UIDj, Pri_j,m, σ_j}, which serves as the notification of the new code image. PCDR relies on the underlying Deluge protocol to distribute packets for a given code image.

3.3 User Verification

In the sensor node verification method, if the packet verification passes, then the nodes accept the program image. The detailed description of each phase is as follows. Upon receiving a signature message {UIDj, Pri_j,m, σ_j}, each sensor node verifies it as follows

- Sensor node first pays attention to the legality of the programming privilege Pri_j and the message m. For example, the node needs to check whether the identity of itself is included in the node identity set of Pri_j. Only if they are valid, the verification procedure goes to the next step.
- Given the system public parameters {G, G_T, ˆe, q, P, PK_{owner},H_1, H_2} assigned by the network owner.
- If the aforementioned verification passes, the sensor node believes that the message m and the privilege Pri_j are from an authorized user with identity UID_j. Hence, the sensor node accepts the root of the Merkle hash tree constructed for page 0. Thus, the nodes can authenticate the hash packets in page 0 once they receive such packets, based on the security of the Merkle hash tree. The hash packets include the hash values of the data packets in page 1. Therefore, after verifying the hash packets, a node can easily verify the data packets in page 1 based on the one-way property of hash functions. Likewise, once the data packets in page i have been verified, a sensor node can easily authenticate the data packets in page i + 1, where i = 1, 2. . . Y − 1.

Obviously, the computation cost by each sensor node for verifying a signature message is dominantly composed of one MapToPoint hash, one ECSM operation, and two pairing operations.

4. Security Analysis of PCDR

We analyze the security of PCDR to verify that the security requirements mentioned in Section I is satisfied.

4.1 Authenticity and Integrity of Code Images

In PCDR, the signature $\sigma j = H2(m) \cdot SKj$ is actually an identity-based signature. Without knowing the private key SKj, it is infeasible to forge a valid signature. Because of the non-deterministic polynomial-time (NP)-hard computation complexity of the Diffie–Hellman problem in G, it is difficult to derive the private SKj by way of UIDj , PKj , P, H1, H2,and PKowner. Therefore, the message m (as well as the root of the Merkle hash tree in page 0) is unforgeable. Thus, the nodes can authenticate each hash packet in page 0 once they receive such packets; based on the security of the Merkle hash tree. The hash packets include the hash values of the data packets in page 1. Therefore, after verifying the hash packets, a node can easily verify the data packets in page 1 based on the one-way property of hash functions. Likewise, once the data packets in page i are verified, a sensor node can easily authenticate the data packets in page i + 1, where i = 1, 2. . . Y − 1. In summary, if an adversary injects a forged modified program image, each receiving node can detect it easily because of the immediate authentication of reprogramming packets.

4.2 Ensurance of Freshness

Obviously, there are two cases for the network users to administrate the program update of a WSN. In the first case, each network user has the privilege to reprogram the sensor nodes in different zones, and there exists no sensor node which is allowed to be reprogrammed by two network users. In step 1 of the sensor node verification phase, a sensor node first checks whether the version number from the received message m is valid. Only if it is valid, the verification procedure goes to the next step. Therefore, the use of the version number of the updated program image can ensure the freshness of PCDR. The other case is that a sensor node may be assigned to multiple network users by the network owner. A feasible approach for achieving the freshness is that a timestamp is used instead of the version number of the updated code image, of the sensor node verification phase, a sensor node first checks whether the timestamp included in the message is fresh. This can ensure that a node always installs the most recent version of a program. In this case, we assume that the WSN is loosely synchronized via some existing efficient time synchronization mechanism.

4.4 Resistance to User and node compromised attacks

only the system public parameters params = {G,GT , ˆe, q, P, PKowner,H1,H2} are preloaded on every sensor node. Thus, no matter how many sensor nodes are compromised, the adversary just obtains params. Obviously, the adversary cannot impersonate any authorized network user by compromising sensor nodes. In other words, no matter how many sensor nodes are compromised, a benign sensor node will not grant the adversary any reprogramming privilege. Even if some network users are compromised, a benign node will not grant the adversary any reprogramming privilege that is beyond the privileges of the compromised users.

4.4 Supporting Different user Privileges

The network owner can restrict user Uj’s activities by defining the reprogramming privilege Prij , which records the levels of user privileges. Since Uj’s public/private key is generated with Prij as input, nobody except the network owner can modify

Prij contained in the signature message and then pass the verification from the sensor nodes.

5. Conclusion

A number of secure reprogramming protocols have been proposed, but none of these approaches support distributed operation. Therefore, in this paper, a secure distributed reprogramming protocol named PCDR has been proposed. In addition to analyzing the security of PCDR, this paper has also reported the evaluation results of PCDR in an experimental network of resource-limited sensor nodes, which shows that PCDR is feasible in practice. To the best of our knowledge, until now, our protocol is the only one that allows authorized users to reprogram sensor nodes in a distributed manner. In some applications, data are also required to be kept confidential due to the possibility of message interception. In future work, we will study how to support confidentiality in distributed reprogramming.

References

[1] Crossbow Technology Inc., Milpitas, CA, Mote In-Network Programming User Reference, 2003.
[2] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, “Distributed collaborative control for industrial automation with wireless sensor and actuator networks,” IEEE Trans. Ind. Electron., vol. 57, no. 12, pp. 4219–4230, December 2010.
[3] J. Carmo, P. Mendes, C. Couto, and J. Correia, “A 2.4-GHz CMOS shortrange wireless-sensor-network interface for automotive applications,” IEEE Trans. Ind. Electron., vol. 57, no. 5, pp. 1764–1771, May 2010.
[4] J. Deng, R. Han, and S. Mishra, “Secure code distribution in dynamically programmable wireless sensor networks,” pp. 292–300, 2006.
[5] N. Bui, O. Ugus, M. Dissegna, M. Rossi, and M. Zorzi, “An integrated system for secure code distribution in wireless sensor networks,” pp. 575–581, 2010.
[6] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, “Securing the deluge network programming system,” pp. 326–333,2006.
[7] V. C. Gungor, B. Lu, and G. P. Hancke, “Opportunities and challenges of wireless sensor networks in smart grid,” IEEE Trans. Ind. Electron., vol. 57, no. 10, pp. 3557–3564, October 2010.
[8] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” 2004.