



Optimizing Improved Secure and Distributed Reprogramming Protocol for Wireless Sensor Network

J.Kanimozhi, M.NaveenKumar, K.Karthick, P.Christianraj

Department of Electronics and Communication Engineering, K.L.N College of Information Technology. India

Department of Electronics and Communication Engineering, K.L.N College of Information Technology. India

Department of Electronics and Communication Engineering, K.L.N College of Information Technology. India

Department of Electronics and Communication Engineering, K.L.N College of Information Technology. India

ABSTRACT—Wireless reprogramming for a wireless sensor network is the process of uploading new code or changing the functionality of existing code. 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. However, it is desirable and sometimes necessary for multiple authorized network users to simultaneously and directly reprogram sensor nodes without involving the base station, which is referred to as distributed reprogramming. In exiting the network owner can also assign different reprogramming privileges to different users by using a secure and distributed reprogramming protocol named SDRP. Here identify an inherent design weakness in the user pre-processing phase of SDRP that is any authorized user to carry out reprogramming But it have to use identity-based cryptography to secure the reprogramming and to reduce the communication and storage requirements of each node. And latter adding 1-B redundant data to eliminate this design weakness. But it is also not reduce design weakness much more better ,so we proposed new improved SDRP to achieve the better security and efficiency of SDRP by increasing the

redundant data to 4-B. And finally the results demonstrate the efficiency improvement over the original SDRP.

KEYWORDS—Authentication, Reprogramming, Security, Wireless Sensor Networks (WSNs).

I. INTRODUCTION

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. 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. Among these protocols, Deluge [7] is generally regarded as the state of the art and included in TinyOS distributions [9]. It uses an epidemic protocol [10] 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 [6]–[8], [11]–[17] are based on 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 [18]. 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. Examples of such networks include a WSN deployed along an international border to monitor weapon smuggling and human trafficking. Having a base station in these WSNs introduces a very attractive attack target. Obviously, for such networks, it is necessary to have authorized network users to be able to carry out reprogramming in a distributed manner.

Another advantage of distributed reprogramming is that, while multiple authorized users are supported, each user may have a different privilege of reprogramming sensor nodes. This is particularly important in large-scale sensor networks owned by an owner and used by different users from both public and private sectors. For example, some current projects, including the Global Earth Observation System of Systems (GEOSS) [19], the National Oceanographic Partnership Program (NOPP) [20], and the Ocean Research Interactive Observatory Networks (ORION) [21], are constructing large-scale sensor networks to adaptively observe the Earth–ocean–atmosphere system. Here, the GEOSS project involves 61 countries, and the NOPP project involves the Defense Advanced Research Projects Agency, the Department of State, and the Department of Homeland Security among others. For such sensor networks, not only the network owners but also some third parties admitted by the network owners will be the users.

In this case, it is expected that network owners and external users should have different reprogramming privileges.

Therefore, the distributed reprogramming approach is more suitable for WSNs. It allows authorized network users to simultaneously and directly update code images on the nodes without involving the base station. Unfortunately, to the best of our knowledge, distributed reprogramming in WSNs has so far received no attention, despite a rich literature on the centralized approach. Similar to the centralized reprogramming protocols, a secure distributed reprogramming protocol should satisfy the following requirements.

The main contributions of this paper are as follows.

1) The need of distributed reprogramming is not completely new, but previous work did not address this need. To set the design objectives of distributed reprogramming, we study the functional requirements of distributed reprogramming.

2) We propose the Secure and Distributed Reprogramming Protocol (SDRP), which extends Deluge to be a secure protocol. The main idea of SDRP is to map the identity and reprogramming privilege of an authorized user into a public-/private-key pair. Based on the public key, user identity and his reprogramming privilege can be verified, and user traceability and different levels of user authorities can be supported. Since a novel identity-based signature scheme is employed in generating the public-/private-key pair of each authorized user, the proposed protocol is efficient for resource-limited sensor nodes and mobile devices in terms of communication and storage requirements. Furthermore, the proposed protocol can achieve all the requirements of distributed reprogramming listed earlier, while keeping the merits of Deluge and Seluge. To the best of our knowledge, this is the first proposed protocol for distributed reprogramming in WSNs.

3) We also implement the proposed protocol in a network of MicaZ nodes [22]. Experimental results show its high efficiency in practice. This is also the first implemented secure distributed reprogramming protocol for WSNs.

The rest of this paper is organized as follows. In Section II, background and preliminary knowledge related to the proposed distributed reprogramming protocol is given. Section III presents the design considerations of distributed reprogramming. In Section IV, SDRP is described in detail. Section V provides theoretical security evaluation of SDRP, demonstrating that the security requirements are satisfied. Section VI describes the implementation and experimental evaluation of SDRP in

a network of MicaZ motes. Section VII concludes this paper and points out future research direction.

II. BACKGROUND AND PRELIMINARIES

A. Network Model

As shown in the lower subfigure in Fig. 1, a WSN consists of a large number of resource-constrained sensor nodes, many sensor network users, and a single network owner. The network users (e.g., soldiers) use mobile devices such as personal digital assistants (PDAs) or laptop PCs to reprogram the sensor nodes. The network owner can be offline, who has bootstrapped the keying materials for the mobile devices to enforce reprogramming privilege policy. It is assumed that the network owner cannot be compromised and has unlimited computational power compared with sensor nodes. Such sensor networks are under construction or planning by many multisponsor programs and projects (e.g., [19]–[21]). The sensor nodes can only perform a limited number of asymmetric cryptographic operations, such as signature verification, due to the large energy consumption of these operations. We also assume that sensor nodes are able to establish pair wise keys between neighbour nodes, for example using the scheme in [23].

B. Trust Model

The network owner only delegates his reprogramming privilege to those network users who have registered. We assume that the special modules (e.g., authentication module for each new program image proposed in this paper and the user access log module) reside in the boot loader section on each sensor node and cannot be overwritten by anyone except the network owner. To achieve this goal, some existing approaches can be employed, such as hardware-based approaches (e.g., security chips) and software-based approaches (e.g., binary translation [24]). Additionally, we assume that the network owner does not impersonate any network user to propagate a new program image.

C. Threat Model

An adversary can launch a wide range of attacks against the network, which can be divided into two kinds, namely, outside and insider attacks. In an outside attack, the adversary does not control any valid sensor nodes in the WSN. The adversary may eavesdrop, modify, forge,

or replay any network traffic in the WSN. It may also inject false messages or forge non existing links in the network by launching a wormhole attack (e.g., [25]). In an insider attack, the adversary can compromise both network users and sensor nodes and then learn the keying materials stored on them. However, we do assume that the adversary cannot compromise an unlimited number of sensor nodes.

As described in Section II-B, an authorized user cannot totally control a sensor node. However, he may load malicious program on some nodes. SDRP can provide user traceability, which will be described in Section V-F. That is, a sensor node can inform the network owner by delivering the identity of such a malicious user.

D. Bilinear Pairing

The notations used throughout this paper are listed in Table I. Let G be a cyclic additive group generated by P and G_T be a cyclic multiplicative group. G and G_T have the same primer order q , i.e., $|G| = |G_T| = q$. Let $e: G \times G \rightarrow G_T$ be a computable bilinear map, which satisfies the following properties.

- 1) Bilinear: $e(aP, bQ) = e(P, Q)^{ab}$, where $P, Q \in G$ and $a, b \in \mathbb{Z}_q$.
- 2) Non degenerate: There exists $P, Q \in G$ such that $e(P, Q) \neq 1_{G_T}$.

Computable: There exists an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G$.

We call such a bilinear map e an admissible pairing, and the modified Weil [27] or Tate pairing on elliptic curves can give a good implementation of such an admissible bilinear pairing. The group that possesses such a map e is called a bilinear group, on which the decisional Diffie–Hellman problem is easy to solve while the computational Diffie–Hellman problem is

Notation	Descriptions
U_j :	the j th network user
S_j :	the j th sensor node
UID_j :	the identity of user U_j
SK_j :	the private key of user U_j
PK_j :	the public key of user U_j
SK_{owner} :	the private key of the network owner
PK_{owner} :	the public key of the network owner
G :	a cyclic additive group
G_T :	a cyclic multiplicative group
e :	a bilinear map: $G \times G \rightarrow G_T$
P :	the generator of the cyclic additive group G
q :	the order of the group G
H_1 :	a MapToPoint hash [26] function maps from $\{0, 1\}^*$ to G
H_2 :	a hash function maps from $\{0, 1\}^*$ to \mathbb{Z}_q^*

believed to be hard. For example, given $P, aP, bP, cP \in G$ and any $a, b, c \in \mathbb{Z}_q$, there exists an efficient algorithm to determine whether $ab=c \pmod q$ by checking $e^{\wedge}(aP, bP) = e^{\wedge}(P, cP)$, while there exists no algorithm that can compute $abP \in G$ with non negligible probability within polynomial time. Note that an open-source pairing-based cryptographic library [28] has been given. The authors have shown that pairing-based cryptosystems are feasible and applicable in resource-limited WSNs.

III. DESIGN CONSIDERATIONS OF DISTRIBUTED REPROGRAMMING

As shown in Fig. 1, a centralized reprogramming protocol involves only two kinds of participants, the base station (administered by the network owner) 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 predefined privileges to reprogram the sensor nodes without involving the owner.

To provide secure and distributed reprogramming, a naive 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. 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), 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. We consider

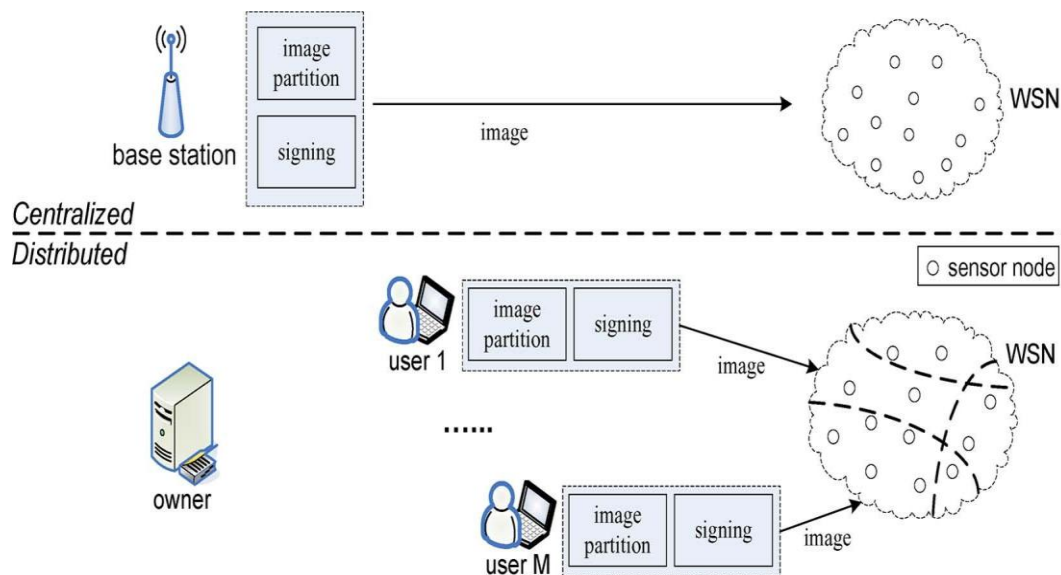


Fig. 1. System overview of centralized and distributed reprogramming approaches

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).

In CBA, each user is equipped with a public-/private-key pair. Each user signs the new code image with his private key using a digital signature scheme such as the elliptic curve digital signature algorithm (ECDSA) [29]. To prove the user's ownership over his public key, the network owner is also equipped with a public-/private-key pair and serves as the certification authority. The owner issues each user, for example, U_j , a public-key certificate, which, to its simplest form, consists of the following contents: $Cert_j = \{U ID_j, P K_j, ExpT, SIG_{SK_{owner}} \{h(U ID_j P K_j ExpT)\}\}$, where $U ID_j$ denotes user U_j 's identity, $P K_j$ indicates U_j 's public key, $ExpT$ denotes the certificate expiration time, SK_{owner} denotes the network owner's private key, and $SIG_{SK_{owner}} \{h(U ID_j P K_j ExpT)\}$ is a signature over $h(U ID_j P K_j ExpT)$ with SK_{owner} . Hence, a simple broadcast message is $\{M, SIG_{SK_j} \{h(U ID_j M)\}, Cert_j\}$, where M denotes the updated code image and SK_j denotes the private key of user U_j . For the purpose of code image authentication, each sensor node is preloaded with the owner's public key ($P K_{owner}$) before the network deployment, and code image verification on each node contains two steps: the user certificate verification and the code image signature verification. CBA has three main disadvantages. First and foremost, it is not efficient in communication, as the certificate has to be transmitted along with the code image across every hop as the message propagates in the WSN. A large per-message overhead will result in more energy consumption on each sensor node. As reported in [30], the length of $\{SIG_{SK_j} \{h(U ID_j M)\}, Cert_j\}$ is at least 126 B, when ECDSA-160 is used. Also, the length of $\{SIG_{SK_j} \{h(U ID_j M)\}, Cert_j\}$ is at least 390 B, when RSA-1024 is used. Second, to authenticate each code image, it always takes two expensive signature verification operations. This is because the certificate should always be authenticated in the first place. Third, the network owner cannot specify a reprogramming privilege for each user.

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 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.

From the aforementioned discussion, it is clear that how to enforce secure and distributed reprogramming is an important and challenging issue in WSNs.

IV. SDRP: THE PROTOCOL

A. Overview of SDRP

Based on the aforementioned design considerations, we propose a novel identity-based signature scheme for distributed re-programming in WSNs. Through the proposed scheme, efforts on certificate management and the transmission overhead can be significantly reduced. Meanwhile, only the system public parameters are loaded on each sensor node. Compared with the traditional public-key cryptosystems, elliptic curve cryptography (ECC) provides a good solution in terms of key size, computational efficiency, and communication efficiency. For example, a 160-b ECC key provides the same level of security as a 1024-b RSA key. Hence, the proposed protocol is based on ECC. In conclusion, we design the proposed protocol very carefully so that it is efficient for resource-constrained sensor nodes and mobile devices.

version num (1)	targeted node identities set (6)	code image size (2)	root of the Merkle hash tree (20)
--------------------	-------------------------------------	------------------------	--------------------------------------

Fig. 2. Example of the format of the message m of SDRP. The byte size of each field is indicated below the label.

Referring to Fig. 1, SDRP 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 user(s). 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.

B. System Initialization

In this phase, the network owner executes the following steps.

- 1) Let G be a cyclic additive group generated by P , G_T be a cyclic multiplicative group, and G and G_T have the same primer order q . Let $e : G \times G \rightarrow G_T$ be a bilinear map.
- 2) Randomly pick a random number $s \in \mathbb{Z}_q^*$ as the master key, and compute the corresponding public key $P K_{owner} = s \cdot P$.
- 3) Choose two secure cryptographic hash functions H_1 and H_2 , where $H_1 : \{0,1\}^* \rightarrow G$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. Then, the system public parameters are $params = \{G, G_T, e, q, P, P K_{owner}, H_1, H_2\}$, which are loaded in each sensor node before deployment.
- 4) Consider a user U_j with identity $U ID_j \in \{0,1\}^*$ who registers to the network owner. After verifying his registration information, the network owner first sets U_j 's public key as $P K_j = H_1(U ID_j \parallel P ri_j) \in G$ and computes the corresponding Private keys $K_j = s \cdot P K_j$. Then, the network owner sends $\{P K_j, SK_j, P ri_j\}$ back to U_j using a secure channel, such as the wired Transport Layer Security protocol. Here, $P ri_j$ denotes the level of user privilege such as the sensor nodes set with specified identities or/and within a specific region that user U_j is allowed to reprogram, and subscription period (i.e., the beginning time and the end time).

V. SECURITY ANALYSIS OF SDRP

In this section, we analyze the security of SDRP to verify that the security requirements mentioned in Section I are

A. Authenticity and Integrity of Code Images

In SDRP, the signature $\sigma_j = H_2(m) \cdot SK_j$ is actually an identity-based signature. Without knowing the private key SK_j , 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 SK_j by way of $U ID_j, P K_j, P, H_1, H_2$, and $P K_{owner}$. 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 $i = 1, 2, \dots, Y - 1$. In summary, if an adversary injects a forged modified program satisfied.

B. Authenticity and Integrity of Code Images

In SDRP, the signature $\sigma_j = H_2(m) \cdot SK_j$ is actually an identity-based signature. Without knowing the private key SK_j ,

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 SK_j by way of $U ID_j, P K_j, P, H_1, H_2$, and $P K_{owner}$. 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 $i = 1, 2, \dots, Y - 1$. 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.

B. 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 (or different sets of sensor nodes according to their identities), 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 SDRP. 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. In step 1) of the sensor node verification phase, a sensor node first checks whether the timestamp included in the message m 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 (e.g., [34]).

C. Resistance to Node- and User-Compromised Attacks

As described in Section IV-B, only the system public parameters $params = \{G, G_T, e, \hat{q}, P, P, K_{owner}, H_1, H_2\}$ are pre-loaded 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. Also, as described in Section IV, 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.

D. Distributed

Here, it is demonstrated that the network owner can enforce strict reprogramming so that the reprogramming privilege is only accessible to users willing to register. As described in Section IV-B–D, in order to pass the signature verification of sensor nodes, each user has to obtain a private key from the network owner. In addition, it is clear that the authorized users are able to carry out reprogramming in a distributed manner.

E. Supporting Different User Privileges

The network owner can restrict user U_j 's activities by defining the reprogramming privilege P_{rij} , which records the levels of user privileges. Since U_j 's public/private key is generated with P_{rij} as input, nobody except the network owner can modify P_{rij} contained in the signature message and then pass the verification from the sensor nodes.

VII. IMPLEMENTATION AND PERFORMANCE EVALUATION

In addition to the security evaluation of ISDRP given in Section V, we further evaluate ISDRP by implementing all components on an experimental test bed. Since it has been demonstrated that Seluge [17] exceeds the security and efficiencies of other centralized reprogramming techniques, here, we choose Seluge for performance comparison.

A. Implementation and Experimental Setup

Our implementation has the network owner, sensor network user, and sensor node side programs. The network owner side programs are C programs using Open SSL [35] running on a 3.2-GHz desktop PC. The sensor network user side programs are C programs using Open SSL running on a 1.6-GHz laptop PC

B. Evaluation Results

We use the following five metrics to evaluate ISDRP, namely, memory overhead, signature message overhead, execution time, propagation delay, and energy overhead. The execution time measures the time duration for each operation of ISDRP (i.e., system initialization, user public-/private-key generation, user signing, and signature verification). The propagation delay is the time required to finish disseminating a code image to all the nodes in the network

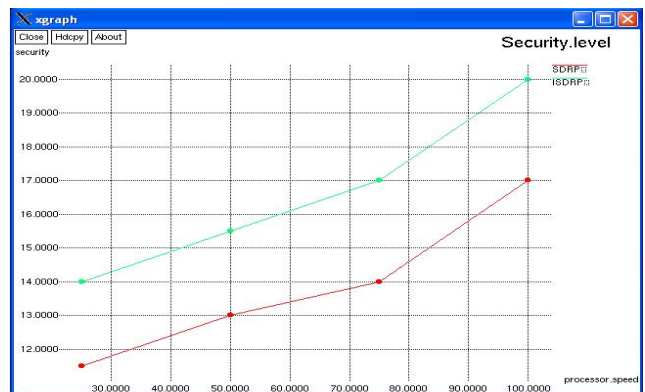


Fig1. Security level

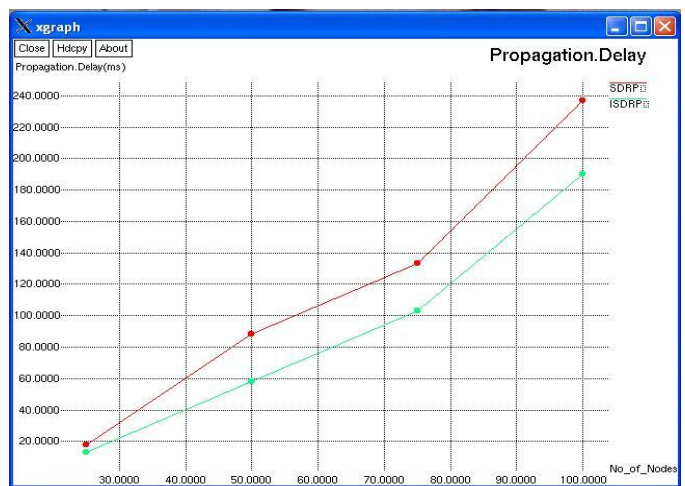


Fig2. propagation Delay

VIII. CONCLUSION AND FUTURE WORK

In the literature, 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 Improved SDRP has been proposed. In addition to analyzing the security of ISDRP, this paper has also reported the evaluation results of ISDRP in an experimental network of resource-limited sensor nodes, which shows that ISDRP 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.

REFERENCES

[1] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4258–4265, Oct. 2009.

[2] 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, Oct. 2010.

[3] 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, Dec. 2010.

[4] X. Cao, J. Chen, Y. Xiao, and Y. Sun, "Building-environment control with wireless sensor and actuator networks: Centralized versus distributed," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3596–3605, Nov. 2010.

[5] J. Carmo, P. Mendes, C. Couto, and J. Correia, "A 2.4-GHz CMOS short-range wireless-sensor-network interface for automotive applications," *IEEE Trans. Ind. Electron.*, vol. 57, no. 5, pp. 1764–1771, May 2010.

[6] Crossbow Technology Inc., Milpitas, CA, Mote In-Network Program-ming User Reference, 2003.
pp. J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. ACM SenSys*, 2004, 81–94.

[7] V. Naik, A. Arora, P. Sinha, and H. Zhang, "Sprinkler: A reliable and energy efficient data dissemination service for extreme scale wireless networks of embedded devices," *IEEE Trans. Mobile Comput.*, vol. 6, no. 7, pp. 777–789, Jul. 2007.

[8] TinyOS: An open-source OS for the networked sensor regime. [Online]. Available: <http://www.tinyos.net/>

[9] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proc. NSDI*, 2004, p. 2.

[10] J. Deng, R. Han, and S. Mishra, "Secure code distribution in dynamically programmable wireless sensor networks," in *Proc. ACM/IEEE IPSN*, 2006, pp. 292–300.
pp. P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in *Proc. ACM/IEEE IPSN*, 2006, 326–333.

[11] P. E. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure dissemination of code updates in sensor networks," in *Proc. ICDCS*, 2006, p. 53. Y. Law, Y. Zhang, J. Jin, M. Palaniswami, and P. Havinga, "Secure rate-less deluge: Pollution-resistant reprogramming and data dissemination for wireless sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, pp.

1–21, 2010.

[12] C. Parra and J. Macias, "A protocol for secure and energy-aware reprogramming in WSN," in *Proc. IWCMC*, 2009, pp. 292–297.

[13] N. Bui, O. Ugus, M. Dissegna, M. Rossi, and M. Zorzi, "An integrated system for secure code distribution in wireless sensor networks," in *Proc. IEEE PERCOM Workshops*, 2010, pp. 575–581.

[14] S. Hyun, P. Ning, A. Liu, and W. Du, "Seluge: Secure and dos-resistant code dissemination in wireless sensor networks," in *Proc. ACM/IEEE IPSN*, 2008, pp. 445–456.

[15] R. Zhang, Y. Zhang, and K. Ren, "DP²AC: Distributed privacy-preserving access control in sensor networks," in *Proc. IEEE INFOCOM*, 2009, pp. 1251–1259.

[16] Geoss. [Online]. Available: <http://www.epa.gov/geoss/>

[17] NOPP. [Online]. Available: <http://www.nopp.org/>

[18] ORION. [Online]. Available: <http://www.oceanleadership.org/2004/ocean-research-interactive-observatory-networks-project-office-manager-selected/>

[22] Crossbow Technology, Inc., Wireless sensor networks. [Online]. Available: http://www.xbow.com/Products/Wireless_Sensor_Networks.htm

[23] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Trans. Inf. Syst. Security*, vol. 8, no. 2, pp. 228–258, May 2005.

[24] L. Gu and J. A. Stankovic, "t-kernel: Providing reliable OS support for wireless sensor networks," in *Proc. ACM SenSys*, 2006, pp. 1–14.

[25] D. He, L. Cui, H. Huang, and M. Ma, "Design and verification of enhanced secure localization scheme in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 7, pp. 1050–1058, Jul. 2009.

[26] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proc. Asiacrypt*, 2001, vol. 2248, pp. 514–532.

[27] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Crypto*, vol. 2139, LNCS, 2001, pp. 213–229.

[28] X. Xiong, D. S. Wong, and X. Deng, "TinyPairing: A fast and light-weight pairing-based cryptographic library for wireless sensor networks," in *Proc. IEEE WCNC*, 2010, pp. 1–6.

[29] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York: Springer-Verlag, 2004.

[30] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Chang, "Energy analysis for public-key cryptography for wireless sensor networks," in *Proc. IEEE PERCOM*, 2005, pp. 324–328.

R. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE S&P*, 1980, pp. 122–133.

[32] H. Guo, K.-S. Low, and H.-A. Nguyen, "Optimizing the localization of a wireless sensor network in real time based on a low-cost microcontroller," *IEEE Trans. Ind. Electron.*, vol. 58, no. 3, pp. 741–749, Mar. 2011.

[33] A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for FR-reduction," *IEICE Trans. Fundam.*, vol. E84-A, no. 5, pp. 1234–1243, 2001.

[34] Z. An, H. Zhu, X. Li, C. Xu, Y. Xu, and X. Li, "Non-identical linear pulse-coupled oscillators model with application to time synchronization in wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2205–2215, Jun. 2011.

[35] OpenSSL. [Online]. Available: <http://www.openssl.org>

[36] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proc. ACM SIGPLAN*, 2003, pp. 1–11.

[37] H. Wang, B. Sheng, C. C. Tan, and Q. Li, "WM-ECC: An elliptic curve cryptography suite on sensor motes," College of William and Mary, Computer Science, Williamsburg, VA, Tech. Rep. WM-CS-2007-11, 2007.