Security Analysis and Improvement of a Secure and Distributed Reprogramming Protocol for Wireless Sensor Networks

Daojing He, *Student Member, IEEE*, Chun Chen, *Member, IEEE*, Sammy Chan, *Member, IEEE*, Jiajun Bu, *Member, IEEE*, and Laurence T. Yang, *Member, IEEE*

***Abstract*—Wireless reprogramming in a wireless sensor network (WSN) is the process of propagating a new code image or relevant commands to sensor nodes. As a WSN is usually deployed in hostile environments, secure reprogramming is and will continue to be a major concern. While all existing insecure/secure reprogramming protocols are based on the centralized approach, it is important to support distributed reprogramming in which multiple authorized network users can simultaneously and directly reprogram sensor nodes without involving the base station. Very recently, a novel secure and distributed reprogramming protocol named *SDRP* has been proposed, which is the first work of its kind. However, in this paper, we identify an inherent design weakness in the user preprocessing phase of SDRP and demonstrate that it is vulnerable to an impersonation attack by which an adversary can easily impersonate any authorized user to carry out reprogramming. Subsequently, we propose a simple modification to fix the identified security problem without losing any features of SDRP. Our experimental results demonstrate that it is possible to eliminate the design weakness by adding 1-B redundant data and that the execution time of the suggested solution in a 1.6-GHz laptop PC is no more than 1 ms. Therefore, our solution is feasible and secure for real-world applications. Moreover, we show that, in order to further improve the security and efficiency of SDRP, any better established identity-based signature algorithm can be directly employed in SDRP. Based on implementation results, we demonstrate efficiency improvement over the original SDRP.**

***Index Terms*—Reprogramming, security, sensor networks, user privilege.**

I. Introduction

WIRELESS reprogramming is the process of propagating a new code image or relevant commands to sensor nodes through wireless links after a wireless sensor network (WSN) is deployed. Due to the need of removing bugs and adding new functionalities, reprogramming is an important operation function of WSNs [1]–[9]. As a WSN is usually deployed in hostile environments such as the battlefield, an adversary may exploit the reprogramming mechanism to launch various attacks. Thus, secure programming is and will continue to be a major concern.

There has been a lot of research focusing on secure reprogramming, and many interesting protocols have been proposed in recent years [10]–[18]. However, all of them are based on the centralized approach which assumes the existence of a base station, and only the base station has the authority to reprogram sensor nodes, as shown in the upper subfigure in Fig. 1. Unfortunately, the centralized approach is not reliable because, when the base station fails or when some sensor nodes lose connections to the base station, it is impossible to carry out reprogramming. Moreover, there are WSNs having no base station at all, and hence, the centralized approach is not applicable. Also, the centralized approach is inefficient, weakly scalable, and vulnerable to some potential attacks along the long communication path.

Alternatively, as shown in the lower subfigure in Fig. 1, a distributed approach can be employed for reprogramming in WSNs. It allows multiple authorized network users to simultaneously and directly update code images on different nodes without involving the base station. Another advantage of distributed reprogramming is that different authorized users may be assigned different privileges of reprogramming sensor nodes. This is particularly important in large-scale WSNs owned by an owner and used by different users from both public and private sectors [19], [20].

Quite recently, He *et al.* have proposed a secure and distributed reprogramming protocol named *SDRP* [21], which is the first work of its kind. Since a novel identity-based signature scheme is employed in generating public/private key pair of each authorized user, SDRP is efficient for resource-limited sensor nodes and mobile devices in terms of communication and storage requirements. Furthermore, SDRP can achieve all requirements of distributed reprogramming listed in [21], while keeping the merits of the well-known mechanisms such as Deluge [22] and Seluge [17]. Also, SDRP has been implemented in a network of resource-limited sensor nodes to show its high efficiency in practice. However, in this paper, we demonstrate that a design weakness exists in the *user*

Manuscript received May 4, 2012; revised August 7, 2012; accepted August 17, 2012. Date of publication September 12, 2012; date of current version June 6, 2013. This work was supported in part by the National Science Foundation of China under Grant 61070155, by the Program for New Century Excellent Talents in University under Grant NCET-09-0685, and by a grant from the Research Grants Council of Hong Kong (Project CityU 111208). The work of L. T. Yang was supported by the Natural Sciences and Engineering Research Council of Canada and the Canada Foundation for Innovation.

D. He, C. Chen, and J. Bu are with the Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: hedaojinghit@gmail.com; chenc@cs.zju.edu.cn; bjj@zju.edu.cn).

S. Chan is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: eeschan@cityu.edu.hk).

L. T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: ltyang@stfx.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2012.2218562

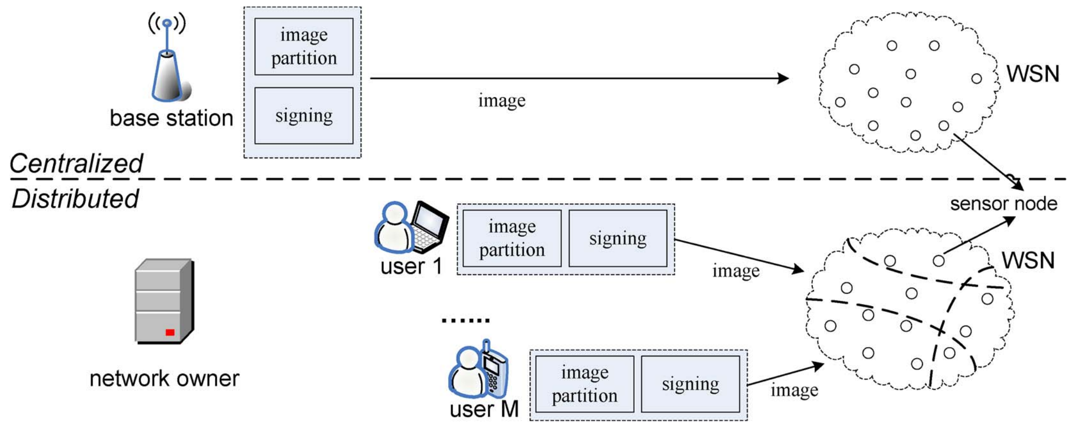

Fig. 1. System overview of centralized and distributed reprogramming approaches.

preprocessing phase of SDRP, and an adversary can easily impersonate any authorized user to carry out reprogramming. To eliminate the identified security vulnerability, we propose a simple modification on SDRP without losing any features (such as distributed reprogramming, supporting different user privileges, dynamic participation, scalability, high efficiency, and robust security) of the original protocol.

Moreover, we show that, for security and efficiency consideration, any efficient identity-based signature algorithm which has survived many years of public scrutiny can be directly employed in SDRP. This paper also reports the experimental results of the improved SDRP in laptop PCs and resource-limited sensor nodes, which show its efficiency in practice.

The remainder of this paper is organized as follows. We briefly review SDRP in Section II and then identify its security weakness in Section III. Section IV presents a modification which remedies the identified weakness. Section V provides an approach to further improve the security and efficiency of SDRP. Section VI concludes this paper and points out future research directions.

II. Brief Overview of SDRP

The 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 public parameters are loaded on each sensor node before deployment. In the user preprocessing phase, if a network user enters the WSN and has a new code image, it 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 code image.

A. System Initialization Phase

The network owner executes the following steps.

1) Let $\mathbb{G}$ be a cyclic additive group and $\mathbb{G}_T$ be a cyclic multiplicative group of the same primer order q. Let P be a generator of $\mathbb{G}$. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map.
2) Pick random $s \in \mathbb{Z}_q^*$ as the master key, and compute public key $PK_{\text{owner}} = s \cdot P$.
3) Choose two secure cryptographic hash functions H_1 and H_2, where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. Then, the public parameters $\{\mathbb{G}, \mathbb{G}_T, \hat{e}, q, P, PK_{\text{owner}}, H_1, H_2\}$ are loaded in each sensor node before deployment.
4) For a user U_j with identity $UID_j \in \{0,1\}^*$, the network owner sets U_j's public key as $PK_j = H_1(UID_j \| Pri_j) \in \mathbb{G}$, computes the private key $SK_j = s \cdot PK_j$, and then sends $\{PK_j, SK_j, Pri_j\}$ back to U_j using a secure channel, such as the wired transport layer security protocol. Here, Pri_j denotes the level of user privilege (e.g., the sensor node set within a specific region that user U_j is allowed to reprogram) and subscription period.

B. User Preprocessing Phase

User U_j takes the following actions.

1) U_j partitions the code 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 $Y-1$. For example, the hash value of packet $Pkt_{Y,1}$ $h(Pkt_{Y,1})$ is included in packet $Pkt_{Y-1,1}$. Here, $Pkt_{Y,1}$ presents the first packet of page Y. Similarly, the hash value of each packet in page $Y-1$ is included in the corresponding packet in page $Y-2$. This process continues until U_j finishes hashing all the packets in page 2 and including their hash values in the corresponding packets in page 1. Then, a Merkle hash tree [23] is used to facilitate the authentication of the hash values of the packets in page 1. We refer to the packets related to this Merkle hash tree collectively as page 0. The root of the Merkle hash tree, the metadata about the code image (e.g., version number, targeted node identity set, and code image size), and a signature over all of them are included in a signature

message. The detailed information can be referred to in [17]. Assume that the message m represents the root of the Merkle hash tree and the metadata about the code image. Then, in order to ensure the authenticity and integrity of the new code image, U_j takes the following actions to construct the signature message.

2) With the private key SK_j, U_j can compute the signature σ_j of the message m, where $\sigma_j = H_2(m) \cdot SK_j$.
3) U_j transmits to the targeted nodes the signature message $\{UID_j, Pri_j, m, \sigma_j\}$, which serves as the notification of the new code image. SDRP relies on the underlying Deluge protocol to distribute packets for a given code image.

C. Sensor Node Verification Phase

Upon receiving a signature message $\{UID_j, Pri_j, m, \sigma_j\}$, each sensor node verifies it as follows.

1) The sensor node first pays attention to the legality of the programming privilege Pri_j and the message m. Only if they are valid, the verification procedure goes to the next step.
2) Given the public parameters, the sensor node performs the following verification:

$$\hat{e}(\sigma_j, P) = \hat{e}\left(H_2(m) \cdot H_1(UID_j|Pri_j), PK_{\text{owner}}\right). \quad (1)$$

If the equation holds, the signature σ_j is valid.

3) 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 $i = 1, 2, \ldots, Y-1$. Only if all verification procedures described previously pass, the sensor node accepts the code image.

III. Security Weakness of SDRP

Recall that, in the user preprocessing phase of SDRP, the signature σ_i is computed as $H_2(m) \cdot SK_j$. This is, however, a design weakness because it enables an adversary $\mathcal{A}$ to obtain the private key SK_j of user U_j as shown in the following.

1) While U_j transmits to the targeted nodes the signature message $\{UID_j, Pri_j, m, \sigma_j\}$, the adversary $\mathcal{A}$ can obtain $\{m, \sigma_j\}$ by eavesdropping. With the public parameter H_2, the adversary $\mathcal{A}$ can compute $\upsilon = (H_2(m))^{-1} \pmod q$, where $H_2(m) \in \mathbb{Z}_q^*$.

   Note that the aforementioned equation involves modular exponentiation with a negative exponent, which can be performed by finding the modular multiplicative inverse u of $H_2(m)$ modulo q using the extended Euclidean algorithm. That is, $\upsilon = (H_2(m))^{-1} \pmod q = u \pmod q$, where $u \cdot H_2(m) \equiv 1 \pmod q$. It should be noted that any integer in $\mathbb{Z}_q^*$ has a multiplicative inverse if and only if that integer is relatively prime to q. That is, the multiplicative inverse u of $H_2(m)$ modulo q exists if and only if $H_2(m)$ and q are coprime (i.e., $\gcd(H_2(m), q) = 1$). In SDRP, since q is prime, all of the nonzero integers in $\mathbb{Z}_q^*$ are relatively prime to q, and therefore, there exists a multiplicative inverse for all of the nonzero integers in $\mathbb{Z}_q^*$.
2) The adversary $\mathcal{A}$ computes the private key $SK_j = \upsilon \cdot \sigma_i$.

   Consequently, the adversary $\mathcal{A}$ can impersonate user U_j to inject bogus code images to take over the control of the whole WSN. Of course, the damage which the adversary $\mathcal{A}$ can make is consistent with the reprogramming privilege of user U_j.

IV. Security Improvement of SDRP

Obviously, if $H_2(m)$ and q are not coprime, an adversary cannot compute the private key SK_j. Therefore, the design weakness of the user preprocessing phase does not exist, and the resulting attack is invalid. To achieve this goal, the following step is suggested to be added into SDRP.

In the system initiation phase, the order q of cyclic additive group $\mathbb{G}$ and cyclic multiplicative group $\mathbb{G}_T$ should be set to a large composite number. Note that Boneh *et al.* have introduced composite-order bilinear groups [24], which have been used to successfully solve many challenging problems in cryptography. In the user preprocessing phase, when user U_j computes m, it can check whether $H_2(m)$ and q are coprime. If yes, before a signature on m is computed, redundant bits are appended into m such that $H_2(m)$ and q are not coprime; otherwise, as described in Section II-B, user U_j directly computes a signature on m. On the other hand, the sensor node verification phase remains the same. That is, compared to the original SDRP, the suggested modification does not incur any overhead on the sensor node side.

In the design of SDRP, the length of m is 29 B. Also assume that the hash function H_2 is implemented using SHA-1 with a 20-B output. Taking q as a 160-b random composite number, we carry out experiments of coprime checking on laptop PCs with different computational powers. In each experiment, q is randomly generated for 1000 times. For each q, m is randomly generated for 1000 times. Thus, each experiment has one million measurements. The experimental results show that, without the addition of any redundant bit, the probability that $H_2(m)$ and q are not coprime is 58.0212%. Also, our implementation results about the average search time of appropriate redundant data and the failure rate with the addition of one or two redundant bytes are summarized in Table I. Here, we consider a 1.6-GHz processor and the addition of one redundant byte as an example. The failure rate for searching appropriate redundant data is 0.4597% for this experiment (i.e., the probability that $H_2(m)$ and q are not coprime is $1 - 0.4597\% = 99.5403\%$), and the search of appropriate redundant data is very fast (i.e., the average execution time is 68.12 μs). Clearly, failure rates

TABLE I
FAILURE RATES AND SEARCH TIME FOR THE ADDITION OF ONE OR TWO REDUNDANT BYTES WHEN q IS A COMPOSITE NUMBER

	Experiment 1		Experiment 2		Experiment 3		Experiment 4		Experiment 5	
processor speed (GHz)	1.6		2		2.4		2.6		3.1	
redundancy (byte)	1	2	1	2	1	2	1	2	1	2
search time (μs)	68.12	608.91	50.82	598.01	49.34	529.85	48.18	368.77	40.02	300.65
failure rate (%)	0.4597	0.0691	0.7380	0.1851	0.7550	0.1780	0.9752	0.1566	0.9916	0.2772

TABLE II
SEARCH TIME FOR THE ADDITION OF ONE OR TWO REDUNDANT BYTES WHEN q IS AN EVEN NUMBER

	Experiment 1		Experiment 2		Experiment 3		Experiment 4		Experiment 5		Experiment 6	
processor speed (GHz)	1.6		1.8		2		2.4		2.6		3.1	
redundancy (byte)	1	2	1	2	1	2	1	2	1	2	1	2
search time (μs)	40.38	40.88	36.50	36.05	32.63	32.48	27.05	27.35	24.93	24.91	21.01	21.10

only depend on the bit length of the added redundant data but not on processor speed.

Furthermore, taking q as a 160-b random even number, we repeat the aforementioned experiments of coprime checking. The experimental results show that, without the addition of any redundant bit, the probability that $H_2(m)$ and q are not coprime is 59.4491%. Also, with the addition of one or two redundant bytes, the failure rates for searching appropriate redundant data are all zero for each experiment (i.e., the probability that $H_2(m)$ and q are not coprime is 100%). On the other hand, our implementation results about the average search time of appropriate redundant data of 1 or 2 B are summarized in Table II. It can be seen that the search of appropriate redundant data is very fast. For example, with the addition of one redundant byte, the average execution times are 40.38 and 36.50 μs on 1.6- and 1.8-GHz laptop PCs, respectively. Here, it is suggested to only use one redundant byte when q is a 160-b random even number. With this setting, not only zero failure rate is achieved but also many advantages in the user preprocessing procedure are obtained in terms of computation, memory usage, and transmission and reception powers.

As shown in Tables I and II, our experimental results demonstrate that the search time for 2-B redundant data is no more than 1 ms in a 1.6-GHz laptop PC. Considering that the clock frequencies of typical mobile devices (e.g., iPhones or laptop PCs) are more than 1 GHz, the suggested modification is efficient for most of mobile devices. According to the aforementioned analysis, the suggested solution is feasible and secure for real-world applications.

V. FURTHER IMPROVEMENT OF SDRP

Designing a secure reprogramming protocol is a difficult task, because there are so many details involved (e.g., the complicated interactions with the environment) that the designer can only try his/her best to make sure his/her protocol is infallible. This holds regardless of whether security proofs are supported by heuristic arguments or formal ways. In reality, the degree of confidence accompanying a security mechanism increases with time only if the underlying algorithms can survive many years of public scrutiny [18].

SDRP is based on a novel and newly designed identity-based signature algorithm. The simple modification presented in Section III can fix the identified security problem of this signature algorithm, but it is still uncertain whether there is any other security weakness in this modified identity-based signature algorithm. To address this issue, it is suggested that, instead of this novel identity-based signature algorithm, some efficient identity-based signature algorithms which have survived many years of public scrutiny can be directly employed in SDRP. For example, we can choose the provably secure identity-based signature proposed by Barreto *et al.* [25]. Aside from providing better security, the method by Barreto *et al.* also improves the efficiency of SDRP due to the following two reasons. First, its signature verification operation only needs one pairing computation and, hence, is among the most efficient ones. Second, the length of its signature is reduced due to bilinear pairing.

A. Improved SDRP

1) System Initialization Phase: The network owner executes the following steps.

1) *Key setup*: Generate the public parameters params $= \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g_1, g_2, g, \hat{e}, \psi, Q_{\text{pub}}, H_3, H_4\}$, and load them in each sensor node before deployment, where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3)$ represents bilinear groups of large prime order p with generators $g_2 \in \mathbb{G}_2$, $g_1 = \psi(g_2) \in \mathbb{G}_1$, and $g = \hat{e}(g_1, g_2)$. The network owner picks a random number $s \in \mathbb{Z}_p$ as the master key and computes public key $Q_{\text{pub}} = s \cdot g_2 \in \mathbb{G}_2$. H_3 and H_4 are cryptographic hash functions, where $H_3 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_4 : \{0,1\}^* \times \mathbb{G}_3 \to \mathbb{Z}_p^*$.
2) *User public/private key generation*: For a user U_j with identity $UID_j \in \{0,1\}^*$, the network owner sets U_j's public key as $P_j = H_3(UID_j \| Pri_j) \in \mathbb{Z}_p^*$, computes the private key $S_j = (1/(P_j + s)) \cdot g_1 = (1/(H_3(UID_j \| Pri_j) + s)) \cdot g_1$, and then sends $\{P_j, S_j, Pri_j\}$ back to U_j through a secure channel. Here, Pri_j denotes the level of user privilege (e.g., the sensor node set within a specific region that user U_j is allowed to reprogram) and subscription period.

2) User Preprocessing Phase: User U_j takes the following actions.

1) This step is the same as step 1) of the *user preprocessing phase* of the original SDRP.
2) With the private key S_j, U_j can compute the signature σ_j of the message m as described in the following. Pick a random number $x \in \mathbb{Z}_p^*$, and compute $r = g^x$.

TABLE III
RUNNING TIME FOR EACH PHASE OF THE IMPROVED SDRP (EXCEPT THE SENSOR NODE VERIFICATION PHASE)

	Key setup	User public/private key generation	User signing
Time (CPU = 1.6 GHz) (μs)	5709.5	1216.5	6617.5
Time (CPU = 1.8 GHz) (μs)	5094.5	1050	5909.5
Time (CPU = 2 GHz) (μs)	4595.5	995.5	5211
Time (CPU = 2.2 GHz) (μs)	4153	852.5	4841
Time (CPU = 2.4 GHz) (μs)	3813	801	4437
Time (CPU = 2.6 GHz) (μs)	3505	720.5	4099
Time (CPU = 3.1 GHz) (μs)	2933	621	3414.5

Set $h = H_4(m, r) \in \mathbb{Z}_p^*$, and compute $W = (x + h) \cdot S_j$. The signature σ_j is the pair $(h, W) \in \mathbb{Z}_p^* \times \mathbb{G}_1$.

3) This step is the same as step 3) of the *user preprocessing phase* of the original SDRP.

3) Sensor Node Verification Phase: Upon receiving a signature message $\{UID_j, Pri_j, m, \sigma_j\}$, each sensor node verifies it as follows.

1) This step is the same as step 1) of the *sensor node verification phase* of the original SDRP.
2) Given the public parameters, the sensor node computes

$$h^* = H_4\left(m, e\left(W, H_3(UID_j \| Pri_j) \cdot g_2 + Q_{\text{pub}}\right) g^{-h}\right)$$

and then sees whether h^* is equal to h or not, where h is from σ_j. If the result is positive, the signature σ_j is valid; otherwise, the node simply drops the signature.

3) This step is the same as step 3) of the *sensor node verification phase* of the original SDRP.

B. Implementation and Performance Evaluation

First, we evaluate the performance of the improved SDRP with respect to the protocols operated by the network owner and user. In our experiment, the network owner and sensor network user side programs have been implemented in C++ (using the integer arithmetic in the publicly available cryptographic library A Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL) [26]) and executed in laptop PCs (with 2-GB RAM) under Ubuntu 11.04 environment with different computational powers. The ηT [27] pairing algorithm over the field $\mathbb{F}_{3^{97}}$ is used (using Barreto's open-source code [28]). Thus, the pairing is symmetric, the length of each element of $\mathbb{G}_1$ is 20 B, and the length of each element of $\mathbb{Z}_p^*$ is also 20 B. As the original SDRP, in our implementation, the byte lengths of UID_j, Pri_j, and m are set to 2, 16, and 29. Table III gives the execution time of some major operations in the improved SDRP. We have executed each operation for 20 000 times and taken an average over them. For example, the execution times of the network owner for the key setup phase and generating public/private key operation (for each network user) are 3.505 and 0.7205 ms on a 2.6-GHz laptop PC, respectively. Also, the signature generation time for a network user is 6.6175 ms on a 1.6-GHz laptop PC.

Next, we consider the signature message overhead of the improved SDRP without considering packet headers. The overhead is $|UID_j| + |Pri_j| + |m| + |\sigma_j| = 2 + 16 + 29 + 20 + 20 = 87$ B. Obviously, the transmission overhead of the improved SDRP is very low, which is very suitable for low-bandwidth WSNs.

TABLE IV
CODE SIZES OF VERIFICATION IMPLEMENTATION OF SIGNATURE MESSAGES IN THE ORIGINAL SDRP AND THE IMPROVED SDRP

		The original SDRP	The improved SDRP
TelosB	ROM (bytes)	22,990	22,504
	RAM (bytes)	660	864
MicaZ	ROM (bytes)	24,530	24,216
	RAM (bytes)	620	816

TABLE V
EXECUTION TIME FOR SIGNATURE VERIFICATION OF THE ORIGINAL SDRP AND THE IMPROVED SDRP

	The original SDRP	The improved SDRP
Time on MicaZ mote (second)	13.793164	10.654834
Time on TelosB mote (second)	33.578467	24.848052

In typical WSNs, sensor nodes are usually resource constrained and battery limited, and they may not be able to execute expensive cryptographic operations efficiently and thus become the bottleneck of a security protocol. Compared to the signature verification algorithm of the original SDRP which mainly requires two pairing, one hash-to-point (with 18-B message (i.e., $(UID_j \| Pri_j)$) as input), and one point scalar multiplication operations on a sensor node, the signature verification algorithm of the improved SDRP mainly requires one pairing, one point scalar multiplication, and one exponentiation (over the field $F_{(3^{97})^6}$) operations on a sensor node and, thus, is more efficient.

We use the following two metrics to compare the original SDRP with the improved SDRP, namely, memory overhead and execution time. The memory overhead measures the exact amount of data space required in the real implementation. Similarly, the execution time measures the time duration for the signature verification operation of the two protocols.

The sensor node side programs are written in nesC and run on two kinds of resource-limited sensor nodes, i.e., MicaZ and TelosB motes. Our motes run TinyOS [29] version 2.x. The MicaZ mote features an 8-b 8-MHz Atmel microcontroller with 4-kB RAM and 128-kB ROM. Also, the TelosB mote has an 8-MHz CPU, 10-kB RAM, 48-kB ROM, 1 MB of flash memory, and an 802.15.4/ZigBee radio.

Different from that in [21], here, the implementation of sensor node side programs is completely based on TinyPairing (a pairing-based cryptographic library) [30], and we do not do any optimization. For example, the pairing, point scalar multiplication, and SHA-1 hash operations from TinyPairing are employed. This implementation mainly involves Pairing, BaseField, ExtField2, SHA1, PointArith, and NN components of TinyPairing. According to Barreto's open-source code for the ηT pairing operation, we implement the exponentiation operation over the field $F_{(3^{97})^6}$ based on TinyPairing.

TABLE VI
EXECUTION TIME FOR EACH CRYPTOGRAPHIC OPERATION ON MICAZ AND TELOSB MOTES

	ηT pairing	Hash-to-point (18 bytes as input)	Exponentiation (over the field $F_{(3^{97})^6}$)	Point scalar multiplication
Time on MicaZ mote (second)	5.321611	0.642383	2.504658	2.498828
Time on TelosB mote (second)	13.013672	1.451131	5.208311	6.041389

Table IV shows the code sizes of verification implementation in the original SDRP and the improved SDRP. For example, the implementation of signature verification in the improved SDRP on a MicaZ mote uses only 816 B of RAM and 24 216 B of ROM. On the other hand, it uses only 864 B of RAM and 22 504 B of ROM on a TelosB mote. The resulting size of our implementation for the improved SDRP corresponds to only 19.92% and 18.47% of the RAM and ROM capacities of MicaZ, respectively. It is clear that the memory overhead of the improved SDRP is comparable to that of the original SDRP.

Table V gives the execution time for signature verification of the original SDRP and the improved SDRP. For example, with respect to the improved SDRP, the execution times for signature verification on a MicaZ mote and TelosB mote are about 10.65 and 24.85 s, respectively. As demonstrated in [21], the proportion of this signature verification time in the total reprogramming time is very small. Clearly, our experiment results show that, for the signature verification operation on sensor nodes, the improved SDRP is more efficient than the original SDRP.

Table VI gives the execution time for each cryptographic operation on MicaZ and TelosB motes. For example, the ηT pairing operation takes 5.3216 and 13.0137 s on a MicaZ mote and a TelosB mote, respectively. For the results in Tables V and VI, we perform each operation 200 times and take an average over them.

Note that the time for signature verification on sensor nodes can be reduced through the use of the optimized cryptographic operations. For example, in our implementation, the exponentiation operation can be optimized.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have pointed out an inherent design weakness in the user preprocessing phase of SDRP. The identified design weakness allows an adversary to impersonate any authorized user to reprogram sensor nodes. We have also presented a modification to fix the problem without sacrificing any desirable feature of SDRP. Moreover, we have chosen the identity-based signature algorithm by Barreto *et al.* as an example to show that, for security and efficiency consideration, any efficient identity-based signature algorithm which has survived many years of public scrutiny can be directly employed in SDRP.

Although Deluge is a de facto standard and has been included in TinyOS distributions [29], several more efficient centralized reprogramming protocols have recently been proposed for WSNs, such as Rateless Deluge [31] and DIssemination Protocol (DIP) [32]. For example, compared to Deluge, Rateless Deluge has many advantages such as reducing latency at moderate levels of packet loss, being more scalable to dense networks, and generally consuming far less energy, a premium resource in WSNs. Thus, in order to further improve the reprogramming efficiency of SDRP, future work should focus on how to integrate SDRP with a more efficient reprogramming protocol like Rateless Deluge, leading to more secure and efficient distributed reprogramming.

In some applications, the network owner and users are different entities. A user may want to hide his/her reprogramming privacy from anyone else, including the network owner. In future work, we will study how to support user privacy preservation in distributed reprogramming.

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. 3596–3604, 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] 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. 762–776, Jul. 2007.

[7] L. Mottola and G. Picco, "Programming wireless sensor networks: Fundamental concepts and state of the art," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 1–51, Apr. 2011.

[8] H. Song, V. Shin, and M. Jeon, "Mobile node localization using fusion prediction-based interacting multiple model in cricket sensor network," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4349–4359, Nov. 2010.

[9] R. C. Luo and O. Chen, "Mobile sensor node deployment and asynchronous power management for wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 59, no. 5, pp. 2377–2385, May 2012.

[10] H. Tan, J. Zic, S. Jha, and D. Ostry, "Secure multihop network programming with multiple one-way key chains," *IEEE Trans. Mobile Comput.*, vol. 10, no. 1, pp. 16–31, Jan. 2011.

[11] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in *Proc. IPSN*, 2006, pp. 326–333.

[12] C. Lim, "Secure code dissemination and remote image management using short-lived signatures in WSNs," *IEEE Commun. Lett.*, vol. 15, no. 4, pp. 362–364, Apr. 2011.

[13] I. Doh, J. Lim, and K. Chae, "Code updates based on minimal backbone and group key management for secure sensor networks," *Math. Comput. Model.*, 2012, to be published.

[14] Y. Law, Y. Zhang, J. Jin, M. Palaniswami, and P. Havinga, "Secure rateless deluge: Pollution-resistant reprogramming and data dissemination for wireless sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, no. 1, pp. 1–21, Jan. 2011.

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

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

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

[18] D. He, S. Chan, C. Chen, and J. Bu, “Secure and efficient dynamic program update in wireless sensor networks,” *Secur. Commun. Netw.*, vol. 5, no. 7, pp. 823–830, Jul. 2012.
[19] Geoss, 2011. [Online]. Available: http://www.epa.gov/geoss/
[20] NOPP, 2012. [Online]. Available: http://www.nopp.org/
[21] D. He, C. Chen, S. Chan, and J. Bu, “SDRP: A secure and efficient reprogramming protocol for wireless sensor networks,” *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4155–4163, Nov. 2012.
[22] J. W. Hui and D. Culler, “The dynamic behavior of a data dissemination protocol for network programming at scale,” in *Proc. SenSys*, 2004, pp. 81–94.
[23] R. Merkle, “Protocols for public key cryptosystems,” in *Proc. IEEE Secur. Privacy*, 1980, pp. 122–134.
[24] D. Boneh, E. Goh, and K. Nissim, “Evaluating 2-DNF formulas on ciphertexts,” in *Proc. TCC*, 2005, vol. 3378, pp. 325–341.
[25] P. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, “Efficient and provably-secure identity-based signatures and signcryption from bilinear maps,” in *Proc. ASIACRYPT*, 2005, pp. 515–532.
[26] M. Scott, “MIRACL—A multiprecision integer and rational arithmetic C/C++ library,” Shamus Softw. Ltd., Dublin, Ireland, 2005.
[27] P. Barreto, S. Galbraith, C. O. hEigeartaigh, and M. Scott, “Efficient pairing computation on supersingular abelian varieties,” *Des., Codes Cryptogr.*, vol. 42, no. 3, pp. 239–271, Mar. 2007.
[28] Paulo S. L. M. Barreto’s Main-Page , 2012. [Online]. Available: http://www.larc.usp.br/~pbarreto/
[29] TinyOS, “An open-source OS for the networked sensor regime,” 2012. [Online]. Available: http://www.tinyos.net/
[30] X. Xiong, D. S. Wong, and X. Deng, “TinyPairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks,” in *Proc. IEEE WCNC*, 2010, pp. 1–6.
[31] A. Hagedorn, D. Starobinski, and A. Trachtenberg, “Rateless Deluge: Over-the-air programming of wireless sensor networks using random linear codes,” in *Proc. ACM/IEEE IPSN*, 2008, pp. 457–466.
[32] K. Lin and P. Levis, “Data discovery and dissemination with DIP,” in *Proc. IPSN*, 2008, pp. 433–444.

Daojing He (S’09) received the B.Eng. and M.Eng. degrees in computer science from Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively. He is currently working toward the Ph.D. degree at Zhejiang University, Hangzhou, China.

His research interests include network and systems security with focus on wireless security. He is an Associate Editor or on the Editorial Boards of international journals such as Wiley’s *Wireless Communications and Mobile Computing*, Wiley’s *Security and Communication Networks*, and *KSII Transactions on Internet and Information Systems*.

Mr. He has served as a Member of the Technical Program Committees for IEEE Globecom 2011, IEEE Symposium on Personal, Indoor, Mobile, and Radio Communications 2011–2012, IEEE International Conference on Communications 2012–2013, and IEEE Wireless Communications and Networking Conference 2011–2012, among others.

Chun Chen (M’06) received the B.S. degree in mathematics from Xiamen University, Xiamen, China, in 1981, and the M.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1984 and 1990, respectively.

He is currently a Professor with the College of Computer Science, Zhejiang University, where he is also the Director of the Institute of Computer Software. His research activity is in image processing, computer vision, and embedded system.

Sammy Chan (S’87–M’90) received the B.E. and M.Eng.Sc. degrees in electrical engineering from The University of Melbourne, Melbourne, Australia, in 1988 and 1990, respectively, and the Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Melbourne, in 1995.

From 1989 to 1994, he was with Telecom Australia Research Laboratories, where he was, first, a Research Engineer and, then, a Senior Research Engineer and a Project Leader between 1992 and 1994. Since December 1994, he has been with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong, where he is currently an Associate Professor.

Jiajun Bu (M’06) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1995 and 2000, respectively.

He is currently a Professor with the College of Computer Science, Zhejiang University, where he is also the Deputy Director of the Institute of Computer Software. His research interests include embedded systems, mobile multimedia, and data mining.

Laurence T. Yang (M’97) received the B.E. degree in computer science from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada.

He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, and the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research interests include parallel and distributed computing and embedded and ubiquitous computing.