SDRP: A Secure and Distributed Reprogramming Protocol for Wireless Sensor Networks

Daojing He, *Student Member, IEEE*, Chun Chen, *Member, IEEE*, Sammy Chan, *Member, IEEE*, and Jiajun Bu, *Member, IEEE*

***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 this case, the network owner can also assign different reprogramming privileges to different users. Motivated by this consideration, we develop a secure and distributed reprogramming protocol named *SDRP*, which is the first work of its kind. The protocol uses identity-based cryptography to secure the reprogramming and to reduce the communication and storage requirements of each node. Moreover, our theoretical analysis demonstrates the security properties of our protocol. We also implement SDRP in a network of resource-limited sensor nodes to show its high efficiency in practice.**

***Index Terms*—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 [1]–[5]. 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 images[1] in WSNs (e.g., [6]–[8]). 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 [11]–[17]. Among them, Seluge [17] 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

Manuscript received February 26, 2011; revised July 6, 2011 and September 20, 2011; accepted November 14, 2011. Date of publication December 6, 2011; date of current version June 19, 2012. 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 the Research Grants Council of Hong Kong under Project CityU 111208.

D. He, C. Chen, and J. Bu are with the National Engineering Research Center for Intelligent Train, College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: hedaojinghit@gmail.com).

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

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.2011.2178214

[1]Note that "code image" and "program image" will be used interchangeably throughout this paper.

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.

1) **Authenticity** and **integrity of code images**: The source of a program image must be verified by a sensor node prior to installation, ensuring that only a trusted source can install a program. In addition, integrity means that an updated program image cannot be modified undetectably.
2) **Freshness**: An earlier version of a program image cannot be installed over the program with the same or greater version number, ensuring that a node always installs the newest version of a program image.
3) **Node compromise tolerance**: A compromised node must be prevented from causing an uncompromised node to violate the aforementioned security requirements.

Other than meeting the aforementioned requirements, a distributed reprogramming protocol should also have the following properties.

1) **Distributed**: The authorized network users are able to simultaneously and directly update code images on the nodes without involving the base station. At the same time, the protocol should prevent unauthorized users from updating sensor nodes.
2) **Supporting different user privileges**: To ensure smooth functioning for a WSN, the level of each user privilege should be limited by the network owner. For example, a user is only allowed to reprogram the sensor nodes set with specified identities or/and within a particular localized area during his subscription period.
3) **Partial reprogram capability**: To prevent sensor nodes from being totally controlled by network users, the special modules (e.g., authentication module for each new program image) on each sensor node cannot be overwritten by anyone except the network owner.
4) **User traceability**: In most application scenarios, traceability is highly desirable, particularly for reprogramming.
5) **Being efficient**: Mobile devices, particularly sensor nodes, usually have limited resources (e.g., CPU processing power, memory, bandwidth, and energy). Thus, energy efficiency (with respect to both communication and computation) and small storage overhead should be given priority to cope with the resource-constrained nature of WSNs.
6) **Scalability**: First, the protocol needs to be efficient even in a large-scale WSN with thousands of sensor nodes, and second, the protocol should be able to support a large number of users.

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 motes [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 pairwise keys between neighbor 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 bootloader 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 nonexisting 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 $\mathbb{G}$ be a cyclic additive group generated by P and $\mathbb{G}_T$ be a cyclic multiplicative group. $\mathbb{G}$ and $\mathbb{G}_T$ have the same primer order q, i.e., $|\mathbb{G}| = |\mathbb{G}_T| = q$. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a computable bilinear map, which satisfies the following properties.

1) Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$, where $P, Q \in \mathbb{G}$ and a, $b \in \mathbb{Z}_q$.
2) Nondegenerate: There exists P, $Q \in \mathbb{G}$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_T}$.
3) Computable: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}$.

We call such a bilinear map $\hat{e}$ 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 $\hat{e}$ is called a bilinear group, on which the decisional Diffie–Hellman problem is easy to solve while the computational Diffie–Hellman problem is believed to be hard. For example, given $P, aP, bP, cP \in \mathbb{G}$ and any $a, b, c \in \mathbb{Z}_q$, there exists an efficient algorithm to determine whether $ab = c \mod q$ by checking $\hat{e}(aP, bP) \stackrel{?}{=} \hat{e}(P, cP)$, while there exists no algorithm that can compute $abP \in \mathbb{G}$ with nonnegligible 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.

TABLE I
NOTATIONS

Notation	Descriptions
U_j:	the jth network user
S_j:	the jth 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
$\mathbb{G}$:	a cyclic additive group
$\mathbb{G}_T$:	a cyclic multiplicative group
$\hat{e}$:	a bilinear map: $\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{G}_T$
P:	the generator of the cyclic additive group $\mathbb{G}$
q:	the order of the group $\mathbb{G}$
H_1:	a MapToPoint hash [26] function maps from $\{0,1\}^*$ to $\mathbb{G}$
H_2:	a hash function maps from $\{0,1\}^*$ to $\mathbb{Z}_q^*$

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 the

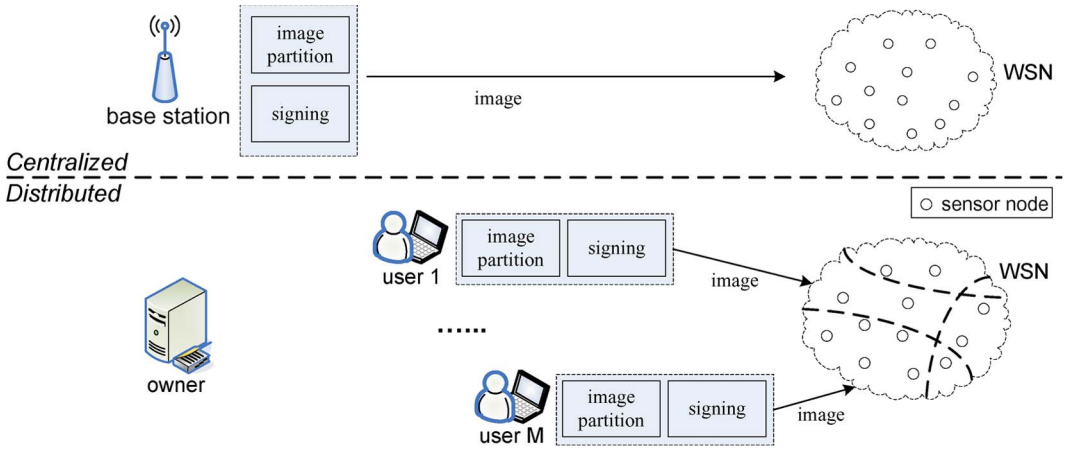


Fig. 1. System overview of centralized and distributed reprogramming approaches.

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 behavior 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: $Cert_j = \{UID_j, PK_j, ExpT, SIG_{SK_{\text{owner}}}\{h(UID_j \| PK_j \| ExpT)\}\}$, where UID_j denotes user U_j's identity, PK_j indicates U_j's public key, $ExpT$ denotes the certificate expiration time, SK_{owner} denotes the network owner's private key, and $SIG_{SK_{\text{owner}}}\{h(UID_j \| PK_j \| ExpT)\}$ is a signature over $h(UID_j \| PK_j \| ExpT)$ with SK_{owner}. Hence, a simple broadcast message is $\{M, SIG_{SK_j}\{h(UID_j \| M)\}, Cert_j\}$, 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 (PK_{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 $\{SIG_{SK_j}\{h(UID_j \| M)\}, Cert_j\}$ is at least 126 B, when ECDSA-160 is used. Also, the length of $\{SIG_{SK_j}\{h(UID_j \| M)\}, Cert_j\}$ 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 reprogramming 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 $\mathbb{G}$ be a cyclic additive group generated by P, $\mathbb{G}_T$ be a cyclic multiplicative group, and $\mathbb{G}$ and $\mathbb{G}_T$ have the same primer order q. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ 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 $PK_{\text{owner}} = s \cdot P$.
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 system public parameters are $params = \{\mathbb{G}, \mathbb{G}_T, \hat{e}, q, P, PK_{\text{owner}}, H_1, H_2\}$, which are loaded in each sensor node before deployment.
4) Consider a user U_j with identity $UID_j \in \{0,1\}^*$ who registers to the network owner. After verifying his registration information, the network owner first sets U_j's public key as $PK_j = H_1(UID_j \| Pri_j) \in \mathbb{G}$ and computes the corresponding private key $SK_j = s \cdot PK_j$. Then, the network owner 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 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).

C. User Preprocessing

Assume that user U_j enters the WSN and has a new program image. U_j takes the following actions.

1) 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 $Y-1$. 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 [31] 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 program image size), and a signature over all of them are included in a signature message. The detailed information can be referred to [17]. Here, we assume that the message m represents the root of the Merkle hash tree and the metadata about the code image. An example format of the message m is shown in Fig. 2. As the output of a one-way hash function (e.g., SHA-1), the length of the root of the Merkle hash tree is 20 B. 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 SDRP should be set according to the specified application scenario. Here, the targeted node identity set field indicates the identities of the sensor nodes which the network user wishes to reprogram. We assume that the length of the identity of each sensor node is 2 B, in which case the protocol supports up to 65 535 nodes. Obviously, the targeted node identity set field is set according to the reprogramming privilege Pri_j of the user. 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.

Note that there are two different cases for user U_j to reprogram the sensor nodes. One case is that U_j wants to reprogram one or more particular sensor nodes, for example, $\{S_1, \ldots, S_j\}$, with identities $\{ID_1, \ldots, ID_j\}$. Here, $j \geq 1$. We assume that sensor nodes do not know their geographical locations. Obviously, this assumption makes SDRP more applicable in the real world. In this case, the identities are added into the targeted node identity set field of m. As will be proved in Section V-A, the authenticity and integrity of the message m (including the identities of the targeted nodes) can be ensured by identity-based signature. Therefore, no adversary can modify the identities and then pass the verification of any sensor node. Of course, reprogramming all sensor nodes via broadcast also

belongs to this case. Referring to Fig. 2, by setting the targeted node identity set field to "0," it indicates that U_j wants to reprogram all sensor nodes. The other case is that U_j wants to reprogram the nodes in a specific region. We assume that the nodes know their geographical locations. More specifically, the nodes know which region they belong to. This can be acquired via deployment knowledge or many existing secure localization schemes (e.g., [25] and [32]). In this case, U_j needs to add the information about the specific region into the targeted node identity set field of m.

The signature of the proposed SDRP has two main merits. On the one hand, the signature overhead is very low. ECDSA signature technique has been widely used in many centralized reprogramming protocols. The length of a signature in SDRP is 21 B (i.e., $|\sigma_j| = 161$ b[2]), while the ECDSA-160 [29] signature is 40 B. Moreover, SDRP does not need any user certificate to be sent along with the signature message due to the adoption of identity-based cryptography; instead, only a 2-B user identity is sent, i.e., $|UID_j| = 2$ B. In this case, SDRP can support 65 536 network users. Note that user identity is attached into the signature message, which can provide user traceability. In contrast, as described in Section III, the ECDSA technique has to incorporate a certificate in the message, which is at least 86 B. On the other hand, from the perspective of signing speed, the proposed protocol does not add any extra signature generation delay compared with that in ECDSA, where both of them take one elliptic curve point scalar multiplication (ECSM) operation for signing.

D. Sensor Node Verification

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. 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.
2) Given the system public parameters $\{\mathbb{G}, \mathbb{G}_T, \hat{e}, q, P, PK_{\text{owner}}, H_1, H_2\}$ assigned by the network owner, 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 because

$$\begin{aligned}\hat{e}(\sigma_j, P) &= \hat{e}\left(H_2(m) \cdot SK_j, P\right) = \hat{e}\left(H_2(m) \cdot s \cdot PK_j, P\right) \\ &= \hat{e}\left(H_2(m) \cdot PK_j, s \cdot P\right) \\ &= \hat{e}\left(H_2(m) \cdot PK_j, PK_{\text{owner}}\right) \\ &= \hat{e}\left(H_2(m) \cdot H_1(UID_j \| Pri_j), PK_{\text{owner}}\right).\end{aligned}$$

[2]It should be noted that, in order to get a short signature, an Miyaji, Nakabayashi and Takano (MNT) curve [33] with 160-b q is employed in SDRP, where the bilinear map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is asymmetric, $\mathbb{G}_1 \neq \mathbb{G}_2$, and elements in $\mathbb{G}_1$ are 161 b long.

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.

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.

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 satisfied.

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 $\mathbb{G}$, it is difficult to derive the private SK_j by way of UID_j, PK_j, P, H_1, H_2, and PK_{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 $i = 1, 2, \ldots, Y-1$. 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 $params = \{\mathbb{G}, \mathbb{G}_T, \hat{e}, q, P, PK_{\text{owner}}, H_1, H_2\}$ 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. 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 Pri_j, which records the levels of user privileges. Since U_j's public/private key is generated with Pri_j as input, nobody except the network owner can modify Pri_j contained in the signature message and then pass the verification from the sensor nodes.

F. User Traceability

In many application scenarios, traceability is highly desirable, particularly for reprogramming, where it is used for collecting the network users' activities for some purposes. For instance, with the knowledge of the network users' reprogramming history, the network owner is able to find out which nodes are frequently reprogrammed, and then can improve the network deployment. In this case, the sensor nodes should have the ability to inform the network owner by delivering the user's identity.

In SDRP, a sensor node obtains the identity of a network user (UID_j) from the signature message. Since every private key is generated based on the corresponding user identity, nobody except the network owner can modify the identity of the network user included in the signature message and then pass the signature verification from the sensor nodes. Therefore, SDRP can provide user traceability.

VI. Implementation and Performance Evaluation

In addition to the security evaluation of SDRP given in Section V, we further evaluate SDRP 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 OpenSSL [35] running on a 3.2-GHz desktop PC. The sensor network user side programs are C programs using OpenSSL running on a 1.6-GHz laptop PC. In addition, the sensor node side programs are written in nesC [36] and run on MicaZ motes. The MicaZ mote features an 8-b 8-MHz Atmel microcontroller with 4-kB RAM and 128-kB ROM. Our MicaZ motes run TinyOS [9] v1.x. We use an indoor test bed consisting of MicaZ motes to evaluate the efficiency of SDRP. In addition, the key size of ECC is set to 160 b. Note that the 160-b ECC key length is considered secure enough for now and immediate future.

We add the following functionalities in the Java tools on the sensor network user side: computation of the hash values of the data packets from the last to the first page, construction of page 0 (i.e., the Merkle hash tree) and hashing packets from the hash values of page 1, and construction of the signature message from the root of the Merkle hash tree and the metadata of the program image.

We modify the *PacketVerifier* module of the Seluge nesC library to perform verification of signature messages, hash packets, and data packets. The system public parameters are generated by OpenSSL [35] and then predistributed to all sensor nodes. The pairwise keys used to distribute cluster keys are also predistributed to all nodes. The pairing operation of pairing-based cryptographic library [28] and the ECSM operation of William&Mary (WM)-ECC library [37] are optimized and then employed in SDRP.

B. Evaluation Results

We use the following five metrics to evaluate SDRP, namely, memory overhead, signature message overhead, execution time, propagation delay, and energy overhead. The execution time measures the time duration for each operation of SDRP (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.

TABLE II
WHOLE CODE SIZE ON MICAZ

	ROM	RAM
Deluge (bytes)	50,562	673
Seluge (bytes)	92,642	2,556
SDRP (bytes)	91,802	2,424

TABLE III
EXECUTION TIME FOR EACH PHASE OF SDRP

system initialization (ms)	0.8
user public/private key generation (ms)	0.9
user signing (ms)	1.8
signature verification (s)	8.0

First, the implementation of signature verification in SDRP on a MicaZ mote uses only 172 B of RAM and 16 220 B of ROM, respectively. This implementation mainly involves PairingC, PointArithM, and NNM components of the pairing-based cryptographic library. The resulting size of our implementation corresponds to only 4.2% and 12.4% of the RAM and ROM capacities of MicaZ, respectively. Table II shows the ROM and RAM usages of SDRP on MicaZ motes. The code sizes of Deluge and Seluge are also included for reference purposes. It is clear that SDRP takes less ROM and RAM than Seluge.

Here, we consider the signature message overhead of SDRP without considering packet headers. The overhead is $|UID_j| + |Pri_j| + |m| + |\sigma_j| = 2 + 16 + 29 + 21 = 68$ B. Obviously, the transmission overhead of SDRP is very low, which is very suitable for low-bandwidth WSNs. Table III gives the execution time of each operation in SDRP. Here, system initialization indicates the generation of the network owner's public key and private key. As shown in Table III, the time for user signing on a 1.6-GHz laptop PC is 1.8 ms. Considering that the clock frequency of a typical PDA is more than 800 MHz, our protocol is efficient for most of mobile devices (e.g., laptop PCs or PDAs). Note that, for the cryptographic operations on desktop PC and laptop PC, we perform the same operation 1000 times and take an average over them. Our experiments show that the time for signature verification on a MicaZ mote is 8 s. Note that the proportion of this signature verification time in the total reprogramming time is very small. Considering Seluge as an example, the total reprogramming time for a 40-kB program image is longer than 380 s in a WSN consisting of 65 MicaZ nodes. Furthermore, the signature verification time of SDRP takes less than 2.1% of the total dissemination time. Considering the benefits that SDRP provides, this time consumption is acceptable.

To further investigate the impact of signature verification on the propagation delay of reprogramming in multihop networks, a multihop experiment was conducted. In this distributed reprogramming experiment, an authorized network user directly reprogrammed six MicaZ motes without involving the network owner. The motes were deployed in a line with the same intervals, where a code image is propagated from one side to the other side. Fig. 3 shows the propagation delays of Deluge, Seluge, and SDRP measured from the experiment. As the code image size increases, the propagation delays of all schemes increase almost linearly. From Fig. 3, it is concluded

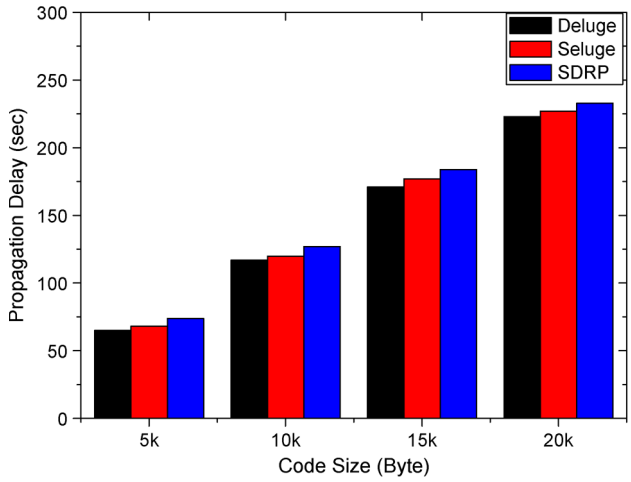


Fig. 3. Propagation delay comparison of three protocols.

that the signature verification by sensor nodes in SDRP only has low impact on the propagation delay of reprogramming. For example, when the code size of a program is 20 kB, the propagation delay of SDRP is only 4.5% more than that of Deluge. This is because, upon receiving the signature packet, each sensor node forwards the signature packet to the next-hop node. Subsequently, it can start to check the validity of the signature packet. Only the signature verification time of the first node has impact on the propagation delay. Further speaking, the increased propagation delay due to signature verification is constant, which is independent against the code size or the number (or deployment) of the targeted nodes.

When a sensor node's radio is always on during the reprogramming process, the energy consumption of the node depends chiefly on the completion time (i.e., propagation delay). The aforementioned experiment shows that the energy overhead of SDRP is similar to that of Deluge or Seluge.

VII. 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 SDRP has been proposed. In addition to analyzing the security of SDRP, this paper has also reported the evaluation results of SDRP in an experimental network of resource-limited sensor nodes, which shows that SDRP 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] 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 Programming User Reference, 2003.

[7] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. ACM SenSys*, 2004, pp. 81–94.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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, pp. 326–333.

[13] P. E. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure dissemination of code updates in sensor networks," in *Proc. ICDCS*, 2006, p. 53.

[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, pp. 1–21, 2010.

[15] C. Parra and J. 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. IEEE PERCOM Workshops*, 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. ACM/IEEE IPSN*, 2008, pp. 445–456.

[18] R. Zhang, Y. Zhang, and K. Ren, "DP^{2}AC: Distributed privacy-preserving access control in sensor networks," in *Proc. IEEE INFOCOM*, 2009, pp. 1251–1259.

[19] Geoss. [Online]. Available: http://www.epa.gov/geoss/

[20] NOPP. [Online]. Available: http://www.nopp.org/

[21] 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 lightweight 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.

[31] 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.

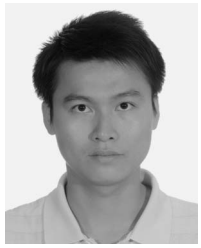
[33] A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for FR-reduction," *IEICE Trans. Fundam.*, vol. E84-A, no. 5, pp. 1234–1243, 2001.

[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.



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 system security with focuses on wireless security.

Mr. He serves as Technical Program Committee member for IEEE Globecom 2011, IEEE Symposium on Personal, Indoor, Mobile, and Radio Communications 2011, IEEE International Conference on Communications 2012, and IEEE Wireless Communications and Networking Conference 2012.

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, between 1992 and 1994, a Senior Research Engineer and a Project Leader. 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 Dean of the Department of Digital Media and Network Technology. His research interests include embedded system, mobile multimedia, and data mining.