Secure and Distributed Data Discovery and Dissemination in Wireless Sensor Networks

Daojing He, *Member, IEEE*, Sammy Chan, *Member, IEEE*, Mohsen Guizani, *Fellow, IEEE*, Haomiao Yang, *Member, IEEE*, and Boyang Zhou

Abstract—A data discovery and dissemination protocol for wireless sensor networks (WSNs) is responsible for updating configuration parameters of, and distributing management commands to, the sensor nodes. All existing data discovery and dissemination protocols suffer from two drawbacks. First, they are based on the centralized approach; only the base station can distribute data items. Such an approach is not suitable for emergent multi-owner-multi-user WSNs. Second, those protocols were not designed with security in mind and hence adversaries can easily launch attacks to harm the network. This paper proposes the first secure and distributed data discovery and dissemination protocol named *DiDrip*. It allows the network owners to authorize multiple network users with different privileges to simultaneously and directly disseminate data items to the sensor nodes. Moreover, as demonstrated by our theoretical analysis, it addresses a number of possible security vulnerabilities that we have identified. Extensive security analysis show DiDrip is provably secure. We also implement DiDrip in an experimental network of resource-limited sensor nodes to show its high efficiency in practice.

Index Terms—Distributed data discovery and dissemination, security, wireless sensor networks, efficiency

1 Introduction

After a wireless sensor network (WSN) is deployed, there is usually a need to update buggy/old small programs or parameters stored in the sensor nodes. This can be achieved by the so-called *data discovery and dissemination protocol*, which facilitates a source to inject small programs, commands, queries, and configuration parameters to sensor nodes. Note that it is different from the code dissemination protocols (also referred to as data dissemination or reprogramming protocols) [1], [2], which distribute large binaries to reprogram the whole network of sensors. For example, efficiently disseminating a binary file of tens of kilobytes requires a code dissemination protocol while disseminating several 2-byte configuration parameters requires data discovery and dissemination protocol. Considering the sensor nodes could be distributed in a harsh environment, remotely disseminating such small data to the sensor nodes through the wireless channel is a more preferred and practical approach than manual intervention.

In the literature, several data discovery and dissemination protocols [3], [4], [5], [6] have been proposed for WSNs. Among them, DHV [3], DIP [5] and Drip [4] are regarded as the state-of-the-art protocols and have been included in the TinyOS distributions. All proposed protocols assume that the operating environment of the WSN is trustworthy and has no adversary. However, in reality, adversaries exist and impose threats to the normal operation of WSNs [7], [8]. This issue has only been addressed recently by [7] which identifies the security vulnerabilities of Drip and proposes an effective solution.

More importantly, all existing data discovery and dissemination protocols [3], [4], [5], [6], [7] employ the centralized approach in which, as shown in the top sub-figure in Fig. 1, data items can only be disseminated by the base station. Unfortunately, this approach suffers from the single point of failure as dissemination is impossible when the base station is not functioning or when the connection between the base station and a node is broken. In addition, the centralized approach is inefficient, non-scalable, and vulnerable to security attacks that can be launched anywhere along the communication path [2]. Even worse, some WSNs do not have any base station at all. For example, for a WSN monitoring human trafficking in a country's border or a WSN deployed in a remote area to monitor illicit crop cultivation, a base station becomes an attractive target to be attacked. For such networks, data dissemination is better to be carried out by authorized network users in a distributed manner.

Additionally, distributed data discovery and dissemination is an increasingly relevant matter in WSNs, especially in the emergent context of shared sensor networks, where sensing/communication infrastructures from multiple owners will be shared by applications from multiple users. For example, large scale sensor networks are built in recent

- *D. He is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, P.R. China, and also with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, P.R. China. E-mail: hedaojinghit@gmail.com.*
- *S. Chan is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, P. R. China. E-mail: eeschan@cityu.edu.hk.*
- *M. Guizani is with Qatar University, Qatar. E-mail: mguizani@ieee.org.*
- *H. Yang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, P. R. China. E-mail: haomyang@uestc.edu.cn.*
- *B. Zhou is with the College of Computer Science, Zhejiang University, P. R. China. E-mail: zby@zju.edu.cn.*

Manuscript received 31 Dec. 2013; revised 29 Mar. 2014; accepted 31 Mar. 2014. Date of publication 10 Apr. 2014; date of current version 6 Mar. 2015. recommended for acceptance by V. B. Misic.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2316830

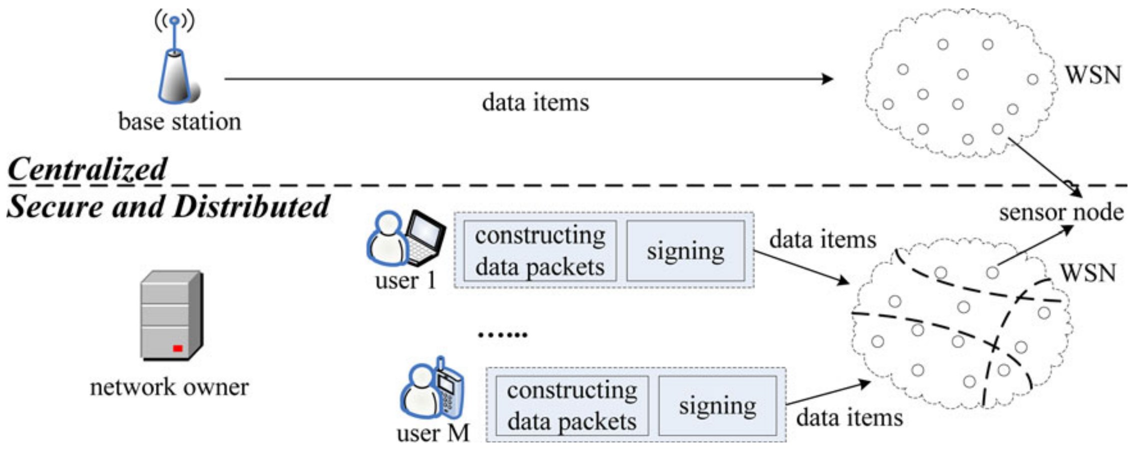

Fig. 1. System overview of centralized and distributed data discovery and dissemination approaches.

projects such as Geoss [9], NOPP [10] and ORION [11]. These networks are owned by multiple owners and used by various authorized third-party users. Moreover, it is expected that network owners and different users may have different privileges of dissemination. In this context, distributed operation by networks owners and users with different privileges will be a crucial issue, for which efficient solutions are still missing.

Motivated by the above observations, this paper has the following main contributions:

1) The need of distributed data discovery and dissemination protocols is not completely new, but previous work did not address this need. We study the functional requirements of such protocols, and set their design objectives. Also, we identify the security vulnerabilities in previously proposed protocols.
2) Based on the design objectives, we propose *DiDrip*. It is the first distributed data discovery and dissemination protocol, which allows network owners and authorized users to disseminate data items into WSNs without relying on the base station. Moreover, our extensive analysis demonstrates that DiDrip satisfies the security requirements of the protocols of its kind. In particular, we apply the provable security technique to formally prove the authenticity and integrity of the disseminated data items in DiDrip.
3) We demonstrate the efficiency of DiDrip in practice by implementing it in an experimental WSN with resource-limited sensor nodes. This is also the first implementation of a secure and distributed data discovery and dissemination protocol.

The rest of this paper is structured as follows. In Section 2, we first survey the existing data discovery and dissemination protocols, and then discuss their security weaknesses. Section 3 describes the requirements for a secure and distributed extension of such protocols. Section 4 presents the network, trust and adversary models. Section 5 describes DiDrip in details. Section 6 provides theoretical analysis of the security properties of DiDrip. Section 7 describes the implementation and experimental results of DiDrip via real sensor platforms. Finally, Section 8 concludes this paper.

2 Security Vulnerabilities in Data Discovery and Dissemination

2.1 Review of Existing Protocols

The underlying algorithm of both DIP and Drip is Trickle [12]. Initially, Trickle requires each node to periodically broadcast a summary of its stored data. When a node has received an older summary, it sends an update to that source. Once all nodes have consistent data, the broadcast interval is increased exponentially to save energy. However, if a node receives a new summary, it will broadcast this more quickly. In other words, Trickle can disseminate newly injected data very quickly. Among the existing protocols, Drip is the simplest one and it runs an independent instance of Trickle for each data item.

In practice, each data item is identified by a unique key and its freshness is indicated by a version number. For example, for Drip, DIP and DHV, each data item is represented by a 3-tuple $<key, version, data>$, where *key* is used to uniquely identify a data item, *version* indicates the freshness of the data item (the larger the version, the fresher the data), and *data* is the actual disseminated data (e.g., command, query or parameter).

2.2 Security Vulnerabilities

An adversary can first place some intruder nodes in the network and then use them to alter the data being disseminated or forge a data item. This may result in some important parameters being erased or the entire network being rebooted with wrong data. For example, consider a new data item (*key*, *version*, *data*) being disseminated. When an intruder node receives this new data item, it can broadcast a malicious data item (*key*, $version^*$, $data^*$), where $version^* > version$. If $data^*$ is set to 0, the parameter identified by *key* will be erased from all sensor nodes. Alternatively, if $data^*$ is different from *data*, all sensor nodes will update the parameter according to this forged data item. Note that the above attacks can also be launched if an adversary compromises some nodes and has access to their key materials.

In addition, since nodes executing Trickle are required to forward all new data items that they receive, an adversary can launch denial-of-service (DoS) attacks to sensor nodes by injecting a large amount of bogus data items. As a result, the processing and energy resources of nodes are expended to process and forward these bogus data items, rather than on the intended functions. Any data discovery and dissemination protocol based on Trickle or its variants is vulnerable to such a DoS attack.

3 Requirements and Design Consideration

A secure and distributed data discovery and dissemination protocol should satisfy the following requirements:

1) *Distributed*. Multiple authorized users should be allowed to simultaneously disseminate data items into the WSN without relying on the base station.
2) *Supporting different user privileges*. To provide flexibility, each user may be assigned a certain privilege level by the network owner. For example, a user can only disseminate data items to a set of sensor nodes with specific identities and/or in a specific localized area. Another example is that a user just has the privilege to disseminate data items identified by some specific keys.
3) *Authenticity and integrity of data items*. A sensor node only accepts data items disseminated by authorized users. Also, a sensor should be able to ensure that received data items have not been modified during the dissemination process.
4) *User accountability*. User accountability must be provided since bad user behaviors and insider attacks should be audited and pinpointed. That is, a sender should not be able to deny the distribution of a data item. At the same time, an adversary cannot impersonate any legitimate user even if it has compromised the network owner or the other legitimate users. In many applications, accountability is desirable as it enables collection of users' activities. For example, from the dissemination record in sensor nodes, the network owner can find out who disseminates most data. This requires the sensor nodes to be able to associate each disseminated data with the corresponding user's identity.
5) *Node compromise tolerance*. The protocol should be resilient to node compromise attack no matter how many nodes have been compromised, as long as the subset of non-compromised nodes can still form a connected graph with the trusted source.
6) *User collusion tolerance*. Even if an adversary has compromised some users, a benign node should not grant the adversary any privilege level beyond that of the compromised users.
7) *DoS attacks resistance*. The functions of the WSN should not be disrupted by DoS attacks.
8) *Freshness*. A node should be able to differentiate whether an incoming data item is the newest version.
9) *Low energy overhead*. Most sensor nodes have limited resources. Thus, it is very important that the security functions incur low energy overhead, which can be decomposed to communication and computation overhead.
10) *Scalability*. The protocol should be efficient even for large-scale WSNs with thousands of sensors and large user population.
11) *Dynamic participation*. New sensor nodes and users can be dynamically added to the network.

In order to ensure security, each step of the existing data discovery and dissemination protocol runs should be identified and then protected. In other words, although code dissemination protocols may share the same security requirements as listed above, their security solutions need to be designed in accordance with their characteristics. Considering the well known open-source code dissemination protocol *Deluge* [1] as an example. Deluge uses an epidemic protocol based on a page-by-page dissemination strategy for efficient advertisement of metadata. A code image is divided into fixed-size pages, and each page is further split into same-size packets. Due to such a way of decomposing code images into packets, our proposed protocol is not applicable for securing Deluge.

The primary challenge of providing security functions in WSNs is the limited capabilities of sensor nodes in terms of computation, energy and storage. For example, to provide authentication function to disseminated data, a commonly used solution is digital signature. That is, users digitally sign each packet individually and nodes need to verify the signature before processing it. However, such an asymmetric mechanism incurs significant computational and communication overhead and is not applicable to sensor nodes. To address this problem, TESLA and its various extensions have been proposed [13], [14], which are based on the delayed disclosure of authentication keys, i.e., the key used to authenticate a message is disclosed in the next message. Unfortunately, due to the authentication delay, these mechanisms are vulnerable to a flooding attack which causes each sensor node to buffer all forged data items until the disclosed key is received.

Another possible approach to authentication is by symmetric key cryptography. However, this approach is vulnerable to node compromise attack because once a node is compromised, the globally shared secret keys are revealed.

Here we choose digital signatures over other forms for update packet authentication. That is, the network owner assigns to each network user a public/private key pair that allows the user to digitally sign data items and thus authenticates himself/herself to the sensor nodes. We propose two hybrid approaches to reduce the computation and communication cost. These methods combine digital signature with efficient data Merkle hash tree and data hash chain, respectively. The main idea is that signature generation and verification are carried out over multiple packets instead of individual packet. In this way, the computation cost per packet is significantly reduced. Since elliptic curve cryptography (ECC) is computational and communication efficient compared with the traditional public key cryptography, DiDrip is based on ECC.

To prevent the network owner from impersonating users, user certificates are issued by a certificate authority of a public key infrastructure (PKI), e.g., local police office.

4 Network, Trust and Threat Models

4.1 Network Model

As shown in the bottom subfigure in Fig. 1, a general WSN comprises a large number of sensor nodes. It is

TABLE 1
Notations

Notation	Description
$SIG_k\{M\}$	the signature on message M with the key k
, or $\|$	concatenation operator of the two bit streams
$h(.)$	public one-way cryptographic hash function (e.g., SHA-1)
$h(M)$	the hash value of message M

administrated by the owner and accessible by many users. The sensor nodes are usually resource constrained with respect to memory space, computation capability, bandwidth, and power supply. Thus, a sensor node can only perform a limited number of public key cryptographic operations during the lifetime of its battery. The network users use some mobile devices to disseminate data items into the network. The network owner is responsible for generating keying materials. It can be offline and is assumed to be uncompromisable.

4.2 Trust Model

Networks users are assigned dissemination privileges by the trusted authority in a PKI on behalf of the network owner. However, the network owner may, for various reasons, impersonate network users to disseminate data items.

4.3 Threat Model

The adversary considered in this paper is assumed to be computationally resourceful and can launch a wide range of attacks, which can be classified as external or insider attacks. In external attacks, the adversary has no control of any sensor node in the network. Instead, it would eavesdrop for sensitive information, inject forged messages, launch replay attack, wormhole attacks, DoS attacks and impersonate valid sensor nodes. The communication channel may also be jammed by the adversary, but this can only last for a certain period of time after which the adversary will be detected and removed.

By compromising either network users or sensor nodes, the adversary can launch insider attacks to the network. The compromised entities are regarded as insiders because they are members of the network until they are identified. The adversary controls these entities to attack the network in arbitrary ways. For instance, they could be instructed to disseminate false or harmful data, launch attacks such as Sybil attacks or DoS attacks, and be non-cooperative with other nodes.

5 DiDrip

Referring to the lower sub-figure in Fig. 1, DiDrip consists of four phases, system initialization, user joining, packet pre-processing and packet verification. For our basic protocol, in system initialization phase, the network owner creates its public and private keys, and then loads the public parameters on each node before the network deployment. In the user joining phase, a user gets the dissemination privilege through registering to the network owner. In packet pre-processing phase, if a user enters the network and wants to disseminate some data items, he/she will need to construct the data dissemination packets and then send them to the nodes. In the packet verification phase, a node verifies each received packet. If the result is positive, it updates the data according to the received packet. In the following, each phase is described in detail. The notations used in the description are listed in Table 1. The information processing flow of DiDrip is illustrated in Fig. 2.

5.1 System Initialization Phase

In this phase, an ECC is set up. The network owner carries out the following steps to derive a private key x and some public parameters $\{y, Q, p, q, h(.)\}$. It selects an elliptic curve E over $GF(p)$, where p is a big prime number. Here Q denotes the base point of E while q is also a big prime number and represents the order of Q. It then selects the private key $x \in GF(q)$ and computes the public key $y = xQ$. After that, the public parameters are preloaded in each node of the network. We consider 160-bit ECC as an example. In this case, y and Q are both 320 bits long while p and q are 160 bits long.

5.2 User Joining Phase

This phase is invoked when a user with the identity UID_j, say U_j, hopes to obtain privilege level. User U_j chooses the private key $SK_j \in GF(q)$ and computes the public key $PK_j = SK_j{\cdot}Q$. Here the length of UID_j is set to 2 bytes, in this case, it can support 65,536 users. Similarly, assume that 160-bit ECC is used, PK_j and SK_j are 320 bits and 160 bits long, respectively. Then user U_j sends a 3-tuple $<UID_j, Pri_j, PK_j>$ to the network owner, where Pri_j denotes the dissemination privilege of user U_j. Upon receiving this message, the network owner generates the certificate $Cert_j$. A form of the certificate consists of the following contents: $Cert_j = \{UID_j, PK_j, Pri_j, SIG_x\{h(UID_j\|PK_j\|Pri_j)\}$, where the length of Pri_j is set to 6 bytes, thus the length of $Cert_j$ is 88 bytes.

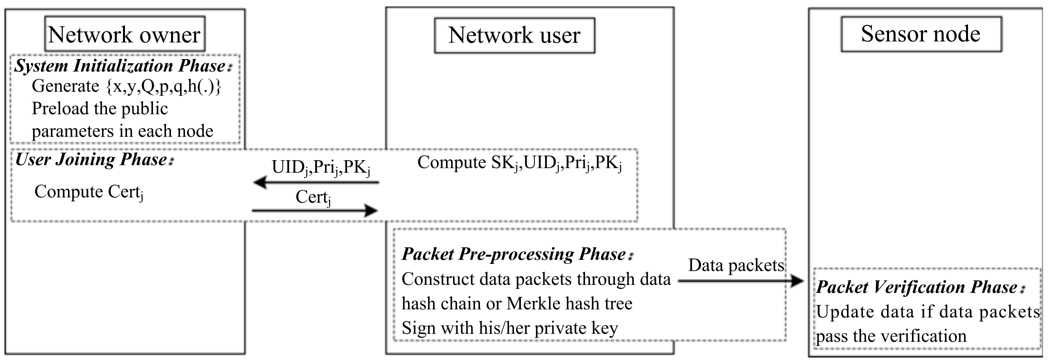
Network owner
Network user
Sensor node
System Initialization Phase:
Generate {x,y,Q,p,q,h(.)}
Preload the public parameters in each node
User Joining Phase:
Compute Cert$_j$
UID$_j$,Pri$_j$,PK$_j$
Cert$_j$
Compute SK$_j$,UID$_j$,Pri$_j$,PK$_j$
Packet Pre-processing Phase:
Construct data packets through data hash chain or Merkle hash tree
Sign with his/her private key
Data packets
Packet Verification Phase:
Update data if data packets pass the verification

Fig. 2. Information processing flow in DiDrip.

5.3 Packet Pre-Processing Phase

Assume that a user, say U_j, enters the WSN and wants to disseminate n data items: $d_i = \{key_i, version_i, data_i\}$, $i = 1, 2, \ldots, n$. For the construction of the packets of the respective data, we have two methods, i.e., data hash chain and the Merkle hash tree [15].

For data hash chain approach, a packet, say P_i is composed of packet header, d_i, and the hash value of packet P_{i+1} (i.e., $H_{i+1} = h(P_{i+1})$) which is used to verify the next packet, where $i = 1, \ldots, n-1$. Here each cryptographic hash H_i is calculated over the full packet P_i, not just the data portion d_i, thereby establishing a chain of hashes. After that, user U_j uses his/her private key SK_j to run an *ECDSA sign* operation to sign the hash value of the first data packet $h(P_1)$ and then creates an advertisement packet P_0, which consists of packet header, user certificate $Cert_j$, $h(P_1)$ and the signature $SIG_{SK_j}\{h(P_1)\}$. Similarly, the network owner assigns a predefined key to identify this advertisement packet.

With the method of Merkle hash tree, user U_j builds a Merkle hash tree from the n data items in the following way. All the data items are treated as the leaves of the tree. A new set of internal nodes at the upper level is formed; each internal node is computed as the hash value of the concatenation of two child nodes. This process is continued until the root node H_{root} is formed, resulting in a Merkle hash tree with depth $\mathcal{D} = \log_2(n)$. Before disseminating the n data items, user U_j signs the root node with his/her private key SK_j and then transmits the advertisement packet P_0 comprising user certificate $Cert_j$, H_{root} and $SIG_{SK_j}\{H_{root}\}$. Subsequently, user U_j disseminates each data item along with the appropriate internal nodes for verification purpose. Note that as described above, user certificate $Cert_j$ contains user identity information UID_j and dissemination privilege Pri_j. Before the network deployment, the network owner assigns a predefined key to identify this advertisement packet.

5.4 Packet Verification Phase

When a sensor node, say S_j, receives a packet either from an authorized user or from its one-hop neighbours, it first checks the packet's *key* field

1) If this is an advertisement packet ($P_0\{Cert_j, h(P_1), SIG_{SK_j}\{h(P_1)\}\}$ for the data hash chain method while $P_0 = \{Cert_j, root, SIG_{SK_j}\{root\}\}$ for the Merkle hash tree method), node S_j first pays attention to the legality of the dissemination privilege Pri_j. For example, node S_j needs to check whether the identity of itself is included in the node identity set of Pri_j. If the result is positive, node S_j uses the public key y of the network owner to run an *ECDSA verify* operation to authenticate the certificate. If the certificate $Cert_j$ is valid, node S_j authenticates the signature. If yes, for the data hash chain method (respectively, the Merkle hash tree method), node S_j stores $< UID_j, H_1 >$ (respectively, $< UID_j, root >$) included in the advertisement packet; otherwise, node S_j simply discards the packet.
2) Otherwise, it is a data packet P_i, where $i = 1, 2, \ldots, n$. Node S_j executes the following procedure:
   For the data hash chain method, node S_j checks the authenticity and integrity of P_i by comparing the hash value of P_i with H_i which has been received in the same round and verified. If the result is positive and the version number is new, node S_j then updates the data identified by the key stored in P_i and replaces its stored $< round, H_i >$ by $< round, H_{i+1} >$ (here H_{i+1} is included in packet P_i); otherwise, P_i is discarded.
   For Merkle hash tree method, node S_j checks the authenticity and integrity of P_i through the already verified root node received in the same round. If the result is positive and the version number is new, node S_j then updates the data identified by the key stored in P_i; otherwise, P_i is discarded.

Remark: To prevent the network owner from impersonating users, system initialization and issue of user certificates can be carried out by the certificate authority of a PKI rather than the network owner.

Comparing the two methods, the data hash chain method incurs less communication overhead than the Merkle hash tree method. In the data hash chain method, only one hash value of a packet is included in each packet. On the contrary, in the Merkle hash tree method, $\mathcal{D}$ (the tree depth) hash values are included in each packet. However, a limitation of the data hash tree method is that it just works well in networks with in-sequence packet delivery. Such a limitation does not exist in the Merkle hash tree method since it allows each packet to be immediately authenticated upon its arrival at a node. Therefore, the choice of each method depends on this characteristic of the WSNs.

5.5 Enhancements

We can enhance the efficiency and security of DiDrip by adding additional mechanisms. Readers are referred to the Appendix of this paper for details.

6 Security Analysis of DiDrip

In the following, we will analyze the security of DiDrip to verify that the security requirements mentioned in Section 3 are satisfied.

Distributed. As described in Section 5.2, in order to pass the signature verification of sensor nodes, each user has to submit his/her private key and dissemination privilege to the network owner for registration. In addition, as described above, authorized users are able to carry out dissemination in a distributed manner.

Supporting different user privileges. Activities of network users can be restricted by setting user privilege Pri_j, which is contained in the user certificate. Since each user certificate is generated based on Pri_j, it will not pass the signature verification at sensor nodes if Pri_j is modified. Thus, only the network owner can modify Pri_j and then updates the certificate accordingly.

Authenticity and integrity of data items. With the Merkle hash tree method (rsp. the data hash chain method), an authorized user signs the root of the Merkle hash tree (rsp. the hash value of the first data packet $h(P_1)$) with his/her private key. Using the network owner's public key, each sensor node can authenticate the user certificate and obtains the user's public key. Then, using the user's public key, each node can authenticate the root of the Merkle hash tree (rsp.

the hash value of the first data packet $h(P_1)$). Subsequently, each node can authenticate other data packets based on the Merkle hash tree (rsp. the data hash chain). With the assumption that the network owner cannot be compromised, it is guaranteed that any forged or modified data items can be easily detected by the authentication process.

User accountability. Users' identities and their dissemination activities are exposed to sensor nodes. Thus, sensor nodes can report such records to the network owner periodically. Since each user certificate is generated according to the user identity, except the network owner, no one can modify the user identity contained in the user certificate which passes the authentication. Therefore, users cannot repudiate their activities.

Node compromise and user collusion tolerance. As described above, for basic protocol, only the public parameters are preloaded in each node. Even for the improved protocol, the public-key/dissemination-privilege pair of each network user is loaded into the nodes. Therefore, no matter how many sensor nodes are compromised, the adversary just obtains the public parameters and the public-key/dissemination-privilege pair of each user. Clearly, the adversary cannot launch any attack by compromising sensor nodes. As described in Section 5, even if some users collude, a benign node will not grant any dissemination privilege that is beyond those of colluding users.

Resistance to DoS attacks. There are DoS attacks against basic DiDrip by exploiting: (1) authentication delays, (2) the expensive signature verifications, and (3) the Trickle algorithm.

First, with the use of Merkle hash tree or data hash chain, each node can efficiently authenticate a data packet by a few hash operations. Second, using the message specific puzzle approach, each node can efficiently verify a puzzle solution to filter a fake signature message and to forward a data packet using Trickle without waiting for signature verification. Therefore, all the above DoS attacks are defended.

DiDrip can successfully defeat all three types of DoS attacks even if there are compromised network users and sensor nodes. Indeed, without the private key and the unreleased puzzle keys of the network users, even an inside attacker cannot forge any signature/data packets.

Ensurance of freshness. If the privilege of a user allows him/her to disseminate data items to his/her own set of nodes, the version number in each item can ensure the freshness of DiDrip. On the other hand, if a node receives data items from multiple users, the version number can be replaced by a timestamp to indicate the freshness of a data item. More specifically, a timestamp is attached into the root of the Merkle hash tree (or the hash value of the first data packet).

Scalability. Different from centralized approaches, an authorized user can enter the network and then disseminates data items into the targeted sensor nodes. Moreover, as to be demonstrated by our experiments in a testbed with 24 TelosB motes in the next section, the security functions in our protocol have low impact on propagation delay. Note that the increase in propagation delay is dominated by the signature verification time incurred at the one-hop neighboring nodes of the authorized user. Thus, the proposed protocol is efficient even in a large-scale WSN with thousands of sensor nodes. Also, as shown in Section 5.2, our protocol can support a large number of users.

Also, as described above, DiDrip can achieve dynamic participation. Moreover, in the next section, our implementation results will demonstrate DiDrip has low energy overhead.

In the following, we give the formal proof of the authenticity and integrity of the disseminated data items in DiDrip based on the three assumptions below:

Assumption 1: There exist pseudo-random functions which are polynomially indistinguishable from truly random functions.

Assumption 2: There exist target collision-resistance (TCR) hash functions [16], where if for all probabilistic-polynomial-time (PPT) adversaries, say $\mathcal{A}$, $\mathcal{A}$ have negligible probability in winning the following game: $\mathcal{A}$ first choose a message m, and then $\mathcal{A}$ are given a random function $h(.)$. To win, $\mathcal{A}$ must output $m' \neq m$ such that $h(m') = h(m)$. Note that in our scheme, the TCR hash function can be implemented by the common hash functions, such as SHA-1.

Assumption 3: ECC signature is existentially unforgeable under adaptive chosen-message attacks. Note that in our scheme, ECC signature can use the standard ECDSA of 160 bits.

Theorem 1. *DiDrip achieves the authenticity and integrity of data items, assuming the indistinguishability between pseudo-randomness and true randomness, and assuming that $h(.)$ is a TCR hash function and ECC signature is existentially unforgeable under adaptive chosen-message attacks.*

Proof. Our theorem follows from Theorem A.1 in [16], and thus here we only give a proof sketch briefly. To begin with, we assume ECC signature generation and verification guarantee authenticity and integrity of signed messages, and every receiver has obtained an authentic copy of the legitimate sender's public key. And we also assume that $h(.)$ is a TCR hash function. Therefore, here the security of DiDrip is proven based on the indistinguishability between pseudo-randomness and true randomness.

First, there exists a PPT adversary $\mathcal{A}$, which can defeat authenticity of data items in DiDrip. This means that $\mathcal{A}$ controls the communication links and manages, with non-negligible probability, to deliver a message m to a receiver R, such that the sender S has not sent m but R accepts m as authentic and coming from S. Then there is a PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the indistinguishability between pseudo-randomness and true randomness with non-negligible advantage. That is, $\mathcal{B}$ gets access to h (as an oracle) and can tell with non-negligible probability if h is a pseudorandom function ($PRF(.)$) or if h is a totally random function.

To this end, $\mathcal{B}$ can query on inputs x of its choice and be answered with $h(x)$. Hence first $\mathcal{B}$ simulates for $\mathcal{A}$ a network with a sender S and a receiver R. Then $\mathcal{B}$ works by running $\mathcal{A}$ in the way similar to that in [17]. Namely, $\mathcal{B}$ chooses a number $l \in \{1, \ldots, n\}$ at random, where n is the total number of packets to be sent in the data dissemination. Note that $\mathcal{B}$ hopes that $\mathcal{A}$ will forge the lth packet P_l.

$\mathcal{B}$ grants access to the oracle h, which is either $PRF(.)$ or an ideal random function. $\mathcal{B}$ can adaptively query an arbitrarily chosen x to the oracle and get the output which is either $PRF(x)$ or a random value uniformly selected from $\{0,1\}^*$. After performing polynomially many queries, $\mathcal{B}$ finally makes the decision of whether or

TABLE 2
The Pros and Cons of All Related Protocols

Protocol	Description	Advantages	Disadvantages
Drip [4]	Data discovery and dissemination	The simplest protocol	Without security consideration & based on centralized approach
DiDrip1	An implementation of the basic DiDrip	Attack resistance & based on distributed approach	-
DiDrip2	An implementation of DiDrip including the function of the message specific puzzle	More efficient than DiDrip1	-

not the oracle is $PRF(.)$ or the ideal random function. As a result, $\mathcal{B}$ wins the game if the decision is correct.

We argue that $\mathcal{B}$ succeeds with non-negligible probability sketchily. If h is a truly random function then $\mathcal{A}$ has only negligible probability to successfully forge the packet P_l in the data items. Therefore, if h is random then $\mathcal{B}$ makes the wrong decision only with negligible probability. On the other hand, we have assumed that if the authentication is done using $PRF(.)$ then $\mathcal{A}$ forges some packet with non-negligible probability ϵ. It follows that if h is $PRF(.)$ then $\mathcal{B}$ makes the right decision with probability at least ϵ/l (which is also non-negligible).

In addition, if the adversary $\mathcal{A}$ is able to cause a receiver R to accept a forged packet P_l, it implies the adversary $\mathcal{A}$ is able to find a collision on $h(P_l^{'}) = h(P_l)$ in the packet P_{l-1}. However, according to Assumption 2, $h(.)$ is a TCR hash function. Moreover, due to the unforgeability of ECC signature (Assumption 3), it is impossible that $\mathcal{A}$ hands R a forged initial packet from S (For data hash chain approach, $\mathcal{A}$ forges the signature $SIG_{SK_j}(P_1)$; For Merkle hash tree method, $\mathcal{A}$ forges the signature $SIG_{SK_j}(root)$. Therefore, the above contradictions means also that the authenticity and integrity of data items in DiDrip. □

7 Implementation and Performance Evaluation

We evaluate DiDrip by implementing all components on an experimental test-bed. Also, we choose Drip for performance comparison.

7.1 Implementation and Experimental Setup

We have written programs that execute the functions of the network owner, user and sensor node. The network owner and user side programs are C programs using OpenSSL [18] and running on laptop PCs (with 2 GB RAM) under Ubuntu 11.04 environment with different computational power. Also, the sensor node side programs are written in nesC and run on resource-limited motes (MicaZ and TelosB). The MicaZ mote features an 8-bit 8-MHz Atmel microcontroller with 4-kB RAM, 128-kB ROM, and 512 kB of flash memory. Also, the TelosB mote has an 8-MHz CPU, 10-kB RAM, 48-kB ROM, 1MB of flash memory, and an 802.15.4/ZigBee radio. Our motes run TinyOS 2.x. Additionally, SHA-1 is used, and the key sizes of ECC are set to 128 bits, 160 and 192 bits, respectively. Throughout this paper, unless otherwise stated, all experiments on PCs (respectively, sensor nodes) were repeated 100,000 times (respectively, 1,000 times) for each measurement in order to obtain accurate average results.

To implement DiDrip with the data hash chain method (rsp. the Merkle hash tree method), the following functionalities are added to the user side program of Drip: construction of data hash chain (rsp. Merkle hash tree) of a round of dissemination data, generation of the signature packet and all data packets. For obtaining version number of each data item, the DisseminatorC and DisseminatorP modules in the Drip nesC library has been modified to provide an interface called DisseminatorVersion. Moreover, the proposed hash tree method is implemented without and with using the message specific puzzle approach presented in Appendix, resulting in two implementations of DiDrip; DiDrip1 and DiDrip2. In DiDrip1, when a node receives a signature/data packet with a new version number, it authenticates the packet before broadcasting it to its next-hop neighbours. On the other hand, in DiDrip2, a node only checks the puzzle solution in the packet before broadcasting the packets. We summarize the pros and cons of all related protocols in Table 2.

Based on the design of DiDrip, we implement the verification function for signature and data packets based on the ECDSA verify function and SHA-1 hash function of TinyECC 2.0 library [19] and add them to the Drip nesC library. Also, in our experiment, when a network user (i.e., a laptop computer) disseminates data items, it first sends them to the serial port of a specific sensor node in the network which is referred to as *repeater*. Then, the repeater carries out the dissemination on behalf of the user using DiDrip.

Similar to [7], we use a circuit to accurately measure the power consumption of various cryptographic operations executed in a mote. The Tektronix TDS 3034C digital oscilloscope accurately measures the voltage V_r across the resistor. Denoting the battery voltage as V_b (which is 3 volts in our experiments), the voltage across the mote V_m is then $V_b - V_r$. Once V_r is measured, the current through the circuit I can be obtained by using Ohm's law. The power consumed by the mote is then $V_m I$. By also measuring the execution time of the cryptographic operation, we can obtain the energy consumption of the operation by multiplying the power and execution time.

7.2 Evaluation Results

The following metrics are used to evaluate DiDrip; memory overhead, execution time of cryptographic operations and propagation delay, and energy overhead. The memory overhead measures the required data space in the implementation. The propagation delay is defined as the time from construction of a data hash chain until the parameters on all sensor nodes corresponding to a round of disseminated data items are updated.

TABLE 3
Running Time for Each Phase of the Basic Protocol of DiDrip (Except the Sensor Node Verification Phase)

	System Initialization	User public/private key generation	The certificate generation (i.e., signing a 20-byte message)
Time (CPU = 1.8 GHz) (μs)	1608.0	1576.31	634.8
Time (CPU = 2.6 GHz) (μs)	1111.3	1092.12	435.4
Time (CPU = 3.1 GHz) (μs)	931.1	915.18	372.3

Table 3 shows the execution times of some important operations in DiDrip. For example, the execution times for the system initialization phase and signing a random 20-byte message (i.e., the output of SHA-1 function) are 1.608 and 0.6348 ms on a 1.8-GHz Laptop PC, respectively. Thus, if SHA-1 is used, generating a user certificate or signing a message takes 0.6348 ms on a 1.8-GHz Laptop PC. Fig. 3 shows the execution times of SHA-1 hash function (extracted from TinyECC 2.0 [19]) on MicaZ and TelosB motes. The inputs to the hash function are randomly generated numbers with length varying from 24 to 156 bytes in increments of 6 bytes. Note that, in our protocol, the hash function is applied to an entire packet. There are several reasons that, possibly, a packet contains a few tens of bytes. First, the advertisement packet has additional information such as certificate and signature. Second, with the Merkle hash tree method, each packet contains the disseminated data item along with the related internal nodes of the tree for verification purpose. Third, although several bytes is a typical size of a data item, sometimes a disseminated data item may be a bit larger. Moreover, for sensors with IEEE 802.15.4 compliant radios, the maximum payload size is 102 bytes for each packet. Therefore, we have chosen a wider range of input size to SHA-1 to provide readers a more complete picture of the performance. We perform the same experiment 10,000 times and take an average over them. For example, the execution times on a MicaZ mote for inputs of 54 bytes, 114 bytes, and 156 bytes are 9.6788, 18.947, and 28.0515 ms, respectively. Also, the execution times on a TelosB mote for inputs of 54, 114, and 156 bytes are 5.7263, 10.7529, and 15.629 ms, respectively.

To measure the execution time of public key cryptography, as shown in Table 4[1], we have implemented the ECC verification operation (with a random 20-byte number as the output) of TinyECC 2.0 library [19] on MicaZ and TelosB motes. For example, it is measured that the signature verification times are 2.436 and 3.955 seconds, which are 252 and 691 times longer than SHA-1 hash operation with a 54-byte random number as input on MicaZ and TelosB motes, respectively. It can be seen packet authentication based on the Merkle hash tree (or data hash chain) is much more efficient. Therefore, it is confirmed that DiDrip is suitable for sensor nodes with limited resources.

Next, we compare the energy consumption of SHA-1 hash function and ECC verification under the condition that the radio of the mote is turned off. When a MicaZ mote is used in the circuit, $V_r = 138$ mV, $I = 6.7779$ mA, $V_m = 2.8620$ V, $P = 19.3983$ mW. When a TelosB mote is used, $V_r = 38$ mV, $I = 1.8664$ mA, $V_m = 2.9620$ V, $P = 5.5283$ mW. With the execution time obtained from Fig. 3, the energy consumption on the motes due to the SHA-1 operation can be determined. For example, the energy consumption of SHA-1 operation with a random 54-byte number as input on MicaZ and TelosB motes are 0.18775 and 0.03166 mJ, respectively. Also, the energy consumption of ECC signature verification operation on MicaZ and TelosB motes are 2835.2555 and 1316.1777 mJ, respectively.

Next, the impact of security functions on the propagation delay is investigated in an experimental network as shown in Fig. 4. The network has 24 TelosB nodes arranged in a 4×6 grid. The distance between each node is about 35 cm,

1. Note that ECC-160 is faster than ECC-128, because the column width of ECC-160 is set to 5 for hybrid multiplication optimization while that of ECC-128 is set to 4.

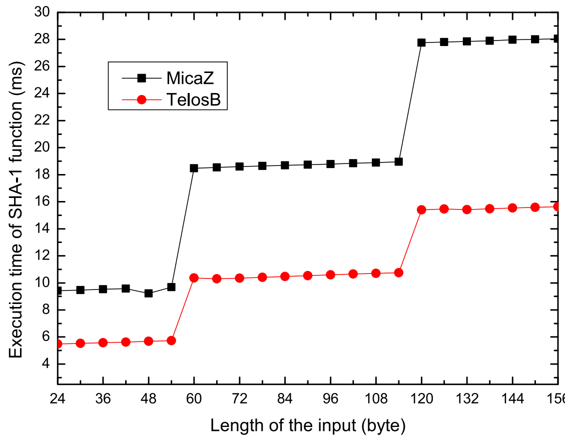

Fig. 3. The execution times of SHA-1 hash function on MicaZ and TelosB motes.

TABLE 4
Running Time for ECC Signature Verification

	ECC-128	ECC-160	ECC-192
Time (MicaZ) (ms)	2310	2436	3754
Time (TelosB) (ms)	3994	3955	5775

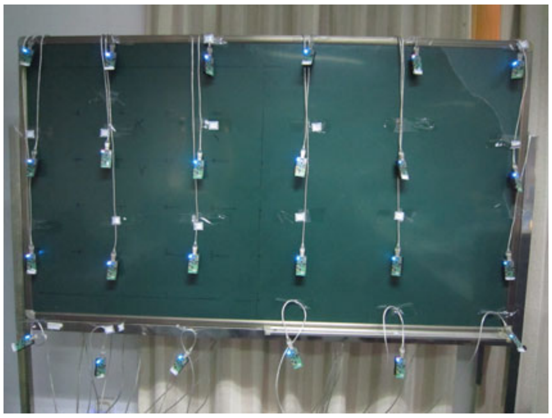
Fig. 4. The 4×6 grid network of TelosB motes for measuring propagation delay.

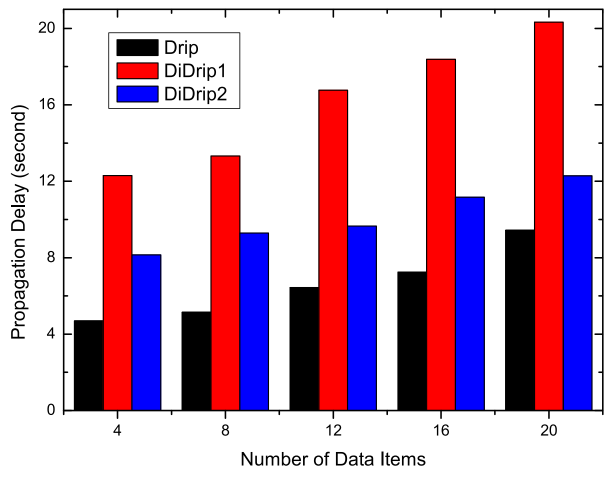

Fig. 5. Propagation delay comparison of three protocols when the data hash chain method is employed.

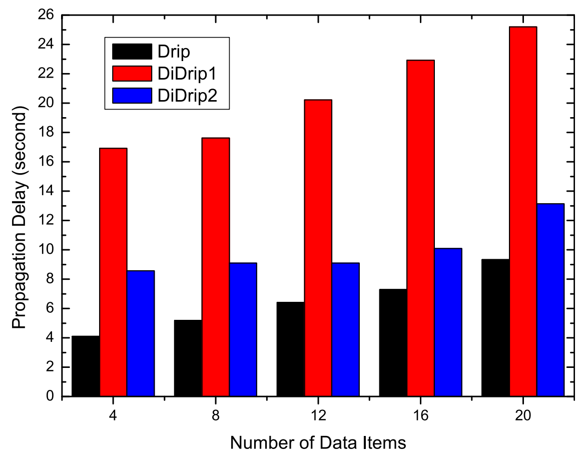

Fig. 6. Propagation delay comparison of three protocols when the Merkle hash tree method is employed.

and the transmission power is configured to be the lowest level so that only one-hop neighbours are covered in the transmission range. The repeater is acted by the node locating at the vertex of the grid.

In the experiments, the packet delivery rate from the network user is 5 packets/s. The lengths of *round* and *data* fields in a data item are set to 4 bits and 2 bytes, respectively. A hash function with 8-byte truncated output is used to construct data hash chains. An ECC-160 signature is 40 bytes long. Each experiment is repeated 20 times to obtain an average measurement. Figs. 5 and 6 plot the average propagation delays of Drip, DiDrip1, and DiDrip2 when the data hash chain and Merkle hash tree methods are employed, respectively. It can be seen that the propagation delay almost increases linearly with the number of data items per round for all three protocols. Moreover, the security functions in DiDrip2 have low impact on propagation delay. For these five experiments of the data hash chain method, DiDrip2 is just 3.448, 4.158, 3.222, 3.919 and 2.855 s more than that of Drip, respectively. Note that the increase in propagation delay is dominated by the signature verification time incurred at the one-hop neighboring nodes of the base station. This is because each node carries out signature verification only after forwarding data packets (with valid puzzle solutions).

Table 5 shows the memory (ROM and RAM) usage of DiDrip2 (with the data hash chain method) on MicaZ and TelosB motes for the case of four data items per round. The code size of Drip and a set of verification functions from TinyECC (secp128r1, secp160r1 and secp192r1, which are implementations based on various elliptic curves according to the Standards for Efficient Cryptography Group) are included for comparison. For example, the size of DiDrip implementation corresponds to 26.18 and 56.82 percent of the RAM and ROM capacities of TelosB, respectively. Clearly, the ROM and RAM consumption of DiDrip is more than that of Drip because of the extra security functions. Moreover, it can be seen that majority of the increased ROM is due to TinyECC.

8 Conclusion and Future Work

In this paper, we have identified the security vulnerabilities in data discovery and dissemination when used in WSNs, which have not been addressed in previous research. Also, none of those approaches support distributed operation. Therefore, in this paper, a secure and distributed data discovery and dissemination protocol named DiDrip has been proposed. Besides analyzing the security of DiDrip, this paper has also reported the evaluation results of DiDrip in an experimental network of resource-limited sensor nodes, which shows that DiDrip is feasible in practice. We have also given a formal proof of the authenticity and integrity of the disseminated data items in DiDrip. Also, due to the open nature of wireless channels, messages can be easily intercepted. Thus, in the future work, we will consider how to ensure data confidentiality in the design of secure and distributed data discovery and dissemination protocols.

Appendix

Further Improvement of DiDrip Security and Efficiency

By the basic DiDrip protocol, we can achieve secure and distributed data discovery and dissemination. To further enhance the protocol, here we propose two modifications to

TABLE 5
Code Sizes (Bytes) on MicaZ and TelosB Motes

		Drip	DiDrip2			TinyECC in DiDrip		
			secp128r1	secp160r1	secp192r1	secp128r1	secp160r1	secp192r1
MicaZ	ROM	14,756	31,778	32,682	32,080	16,986	17,950	17,370
	RAM	423	2,256	2,556	2,856	1,370	1,670	1,970
TelosB	ROM	14,808	27,882	27,930	27,976	13,908	13,992	14,062
	RAM	453	2,381	2,681	2,981	1,467	1,767	2,067

improve the efficiency and security of DiDrip. For brevity, only those parts of the basic protocol that require changes will be presented.

Avoiding the Generation, Transmission and Verification of Certificates

There are some efficiency problems caused by the generation, transmission, and verification of certificates. First, it is not efficient in communication, as the certificate has to be transmitted along with the advertisement packet 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. Second, to authenticate each advertisement packet, it always takes two expensive signature verification operations because the certificate should always be authenticated first. To address these challenges, a feasible approach is that before the network deployment, the public-key/dissemination-privilege pair of each network user is loaded into the sensor nodes by the network owner. Once a new user joins the network after the network deployment, the network owner can notify the sensor nodes of the user's public key/dissemination privilege through using the private key of itself. The detailed description is as follows.

User Joining Phase

According to the basic protocol of DiDrip, user U_j generates its public and private keys and sends a 3-tuple $< UID_j, Pri_j, PK_j >$ to the network owner. When the network owner receives the 3-tuple, it no longer generates the certificate $Cert_j$. Instead, it signs the 3-tuple with its private key and sends it to the sensor nodes. Finally, each node stores the 3-tuple.

Packet Pre-Processing Phase

The user certificate $Cert_j$ stored in packet P_0 is replaced by UID_j.

Packet Verification Phase

If this is an advertisement packet, according to the received identity UID_j, node S_j first picks up the dissemination privilege Pri_j from its storage and then pays attention to the legality of Pri_j. If the result is positive, node S_j uses the public key PK_j from its storage to run an *ECDSA verify* operation to authenticate the signature; otherwise, node S_j simply discards the packet. Note that node S_j does not need to authenticate the certificate.

As described above, the public-key/dissemination-privilege pair $< UID_j, Pri_j, PK_j >$ of each network user is just $2 + 6 + 40 = 48$ bytes. Therefore, assuming the protocol supports 500 network users, the code size is about 23 KB. We consider the resource-limited sensor nodes such as TelosB motes as examples. The 1-MB Flash memory is enough for storing these public parameters.

Message Specific Puzzle Approach for Resistance to DoS attacks

DiDrip uses a digital signature to bootstrap the authentication of a round of data discovery and dissemination. This authentication is vulnerable to DoS attacks. That is, an adversary may flood a lot of illegal signature message (i.e., advertisement messages in this paper) to the sensor nodes to exhaust their resources and render them less capable of processing the legitimate signature messages. Such an attack can be defended by applying the message specific puzzle approach [2]. This approach requires each signature message to contain a puzzle solution. When a node receives a signature message, it first checks that the puzzle solution is correct before verifying the signature. There are two characteristics of the puzzles. First, the puzzles are difficult to be solved but their solutions are easy to be verified. Second, there is a tight time limit to solve a puzzle. This discourages adversaries to launch the DoS attack even if they are computationally powerful. More details about this approach can be found in [2].

Another advantage of applying the message specific puzzle is to reduce the dissemination delay, which is the time for a disseminated packet to reach all nodes in a WSN. Recall that in step 1.a) of the packet verification phase, when a node receives the signature packet, it first carries out the signature verification before using the Trickle algorithm to broadcast the signature packet. This means that the dissemination delay depends on the signature verification time t_{sv}. On the other hand, when the message specific puzzle approach is applied, a node can just verify the validity of puzzle solution before broadcasting the signature packet. Then, the dissemination delay only depends on the puzzle solution verification time t_{pv}. Since $t_{pv} \ll t_{sv}$, the dissemination delay is significantly reduced. Moreover, the reduction in dissemination delay is proportional to the network size. This is demonstrated by the experiments presented in Section 7.2.

Acknowledgments

This research is supported by a strategic research grant from City University of Hong Kong [Project No. 7004054], the Fundamental Research Funds for the Central Universities, and the Specialized Research Fund for the Doctoral Program of Higher Education. D. He is the corresponding author of this article.

References

[1] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 81–94.

[2] D. He, C. Chen, S. Chan, and J. Bu, "DiCode: DoS-resistant and distributed code dissemination in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 5, pp. 1946–1956, May 2012.

[3] T. Dang, N. Bulusu, W. Feng, and S. Park, "DHV: A code consistency maintenance protocol for multi-hop wireless sensor networks," in *Proc. 6th Eur. Conf. Wireless Sensor Netw.*, 2009, pp. 327–342.

[4] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Proc. Eur. Conf. Wireless Sensor Netw.*, 2005, pp. 121–132.

[5] K. Lin and P. Levis, "Data discovery and dissemination with DIP," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2008, pp. 433–444.

[6] M. Ceriotti, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment," in *Proc. IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2009, pp. 277–288.

[7] D. He, S. Chan, S. Tang, and M. Guizani, "Secure data discovery and dissemination based on hash tree for wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4638–4646, Sep. 2013.

[8] M. Rahman, N. Nasser, and T. Taleb, "Pairing-based secure timing synchronization for heterogeneous sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, 2008, pp. 1–5.

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

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

[11] ORION. [Online]. Available: http://www.joiscience.org/ocean observing/advisors

[12] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks," in *Proc. 1st Conf. Symp. Netw. Syst. Design Implementation*, 2004, pp. 15–28.

[13] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and secure source authentication for multicast," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2001, pp. 35–46.

[14] Y. Chen, I. Lin, C. Lei, and Y. Liao, "Broadcast authentication in sensor networks using compressed bloom filters," in *Proc. 4th IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, 2008, pp. 99–111.

[15] R. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Security Privacy*, 1980, pp. 122–134.

[16] M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," in *Proc. Adv. Cryptology*, 1997, pp. 56–73.

[17] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proc. IEEE Security Privacy*, 2000, pp. 56–73.

[18] OpenSSL. [Online]. Available: http://www.openssl.org

[19] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. ACM/IEEE Inf. Process. Sensor Netw.*, 2008, pp. 245–256.

Daojing He (S'07-M'13) received the BEng and MEng degrees from the Harbin Institute of Technology, China, and the PhD degree from Zhejiang University, China, all in computer science in 2007, 2009, and 2012, respectively. He is with the School of Computer Science and Engineering, South China University of Technology, P.R. China, and also with the College of Computer Science and Technology, Zhejiang University, P.R. China. His research interests include network and systems security. He is an associate editor or on the editorial board of some international journals such as *IEEE Communications Magazine*, *Springer Journal of Wireless Networks*, *Wiley's Wireless Communications and Mobile Computing Journal*, *Journal of Communications and Networks*, *Wiley's Security and Communication Networks Journal*, and *KSII Transactions on Internet and Information Systems*. He is a member of the IEEE.

Sammy Chan (S'87-M'89) received the BE and MEngSc degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and the PhD degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. From 1989 to 1994, he was with Telecom Australia Research Laboratories, first as a research engineer, and between 1992 and 1994 as a senior research engineer and project leader. Since December 1994, he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an associate professor. He is a member of the IEEE.

Mohsen Guizani (S'85-M'89-SM'99-F'09) received the BS (with distinction) and MS degrees in electrical engineering, the MS and PhD degrees in computer engineering in 1984, 1986, 1987, and 1990, respectively, from Syracuse University, Syracuse, New York. He is currently a professor and the associate vice president for Graduate Studies at Qatar University, Qatar. His research interests include computer networks, wireless communications and mobile computing, and optical networking. He currently serves on the editorial boards of six technical journals and the founder and EIC of "*Wireless Communications and Mobile Computing*" Journal published by John Wiley (http://www.interscience.wiley.com/jpages/1530-8669/). He is a fellow of the IEEE and a senior member of ACM.

Haomiao Yang (M'12) received the MS and PhD degrees in computer applied technology from the University of Electronic Science and Technology of China (UESTC) in 2004 and 2008, respectively. From 2012 to 2013, he worked as a postdoctoral fellow at Kyungil University, Republic of Korea. Currently, he is an associate professor at the School of Computer Science and Engineering, UESTC, China. His research interests include cryptography, cloud security, and big data security. He is a member of the IEEE.

Boyang Zhou is currently working toward the PhD degree from the College of Computer Science at Zhejiang University. His research areas include software-defined networking, future internet architecture and flexible reconfigurable networks.