ACCEPTED FROM OPEN CALL

SMALL DATA DISSEMINATION FOR WIRELESS SENSOR NETWORKS: THE SECURITY ASPECT

DAOJING HE, SAMMY CHAN, MOHSEN GUIZANI

ABSTRACT

In wireless sensor networks small data dissemination protocols are used to adjust configuration parameters of sensors, or distribute management commands and queries to sensors. For security reasons every disseminated data item should be authenticated to prevent an adversary from installing malicious data items in the network. Unfortunately, security is not among the design considerations of existing small data dissemination protocols. In this article we identify the security vulnerabilities of these protocols and review a recently proposed solution addressing this issue. Furthermore, we suggest an enhancement to this solution to make the security function more efficient.

INTRODUCTION

Wireless sensor networks (WSNs) have been widely deployed for civilian applications in the past decade. These applications include environmental/industrial/health care monitoring, security surveillance and information collection in disaster areas. A typical WSN consists of a large number of small-sized battery-operated sensor nodes with sensing, communication, and limited computing capabilities [1]. After a WSN is deployed, from time to time some common variables stored in each node of the network need to be updated. The *small data dissemination protocols* add, delete, and modify such variables by requesting each node to exchange packets so that they eventually become consistent across the network. Common applications of such protocols include injecting commands and queries to nodes, and changing configuration parameters of nodes. For example, installing a micro program throughout the entire network can be viewed as the process of establishing consistency on a variable containing the program. Note that it is different from the well studied code dissemination (sometimes called data dissemination or reprogramming) protocols [2, 3], which distribute binary files of tens of kilobytes into a network, enabling complete system reprogramming. In other words, efficiently disseminating a large binary file requires code dissemination protocols, while disseminating several bytes of configuration parameters requires small data dissemination protocols.

Existing approaches of data dissemination and gathering can be classified as non-coding and coding based, respectively. For non-coding-based approaches, their performance degrades as the channel error rate increases. For coding-based approaches, network coding techniques are commonly employed to make dissemination more robust to packet loss. However, they involve complex arithmetic operations, which are not suitable for energy-constrained sensor nodes. For this reason data gathering approaches based on distributed rateless codes [4] are more attractive as they only involve bitwise XOR operations on a small number of packets.

Recently several small data dissemination protocols have been proposed. Among them, Drip [5], DIP [6], and DHV [7] are generally regarded as the state of the art and included in TinyOS distributions [8]. However, all these small data dissemination protocols focus on how to improve the efficiency and reliability of the operation, but do not consider any security aspects. On the other hand, many secure code dissemination protocols have been developed; readers are referred to [3] and references therein for more information.

In this article we first briefly review small data dissemination protocols of WSNs, and then discuss their security vulnerabilities. Such vulnerabilities allow an adversary to reboot the entire network with wrong data, or erase an important variable from all nodes. Not addressing them makes all the existing small data dissemination protocols vulnerable to security attacks. Subsequently, we discuss the requirements for a secure and efficient small data dissemination protocol. Then we report a recently proposed secure version of Drip, SeDrip, which is designed for generic WSNs. Finally, we demonstrate a possible enhancement of SeDrip to make its security function more efficient so that it is suitable for more resource-constrained WSNs such as wireless body sensor networks (WBSNs).

SMALL DATA DISSEMINATION

Ideally, when the base station injects new data into the WSN, each node should receive a copy of the data. However, in practice this may not be the case for two reasons. First, in a static network, nodes operating in sleeping mode may

Daojing He is with South China University of Technology.

Sammy Chan is with City University of Hong Kong.

Mohsen Guizani is with Qatar University.

1536-1284/14/$25.00 © 2014 IEEE

Protocol	Characteristic	Total number of transmitted messages	Security vulnerabilities
Drip	Disseminates each data item with a separate instance of Trickle	$O(T)$	Eavesdropping, injection of forged messages, message replaying, impersonating valid sensor nodes, wormhole attacks, DoS attack,node compromise, and so on.
DIP	A summary message is periodically broadcasted by a node that only contains hashes of keys and version numbers	$O(N \log(T))$	
DHV	Allows nodes to carefully select and transmit only necessary bit level information to detect a newer code version in the network	$O(N)$	

Table 1. Characteristics and security vulnerabilities of various small data dissemination protocols of WSNs.

miss the data update. Also, as the wireless channel is error prone, some nodes may not be able to receive the data correctly. Second, in a dynamic network in which nodes join and leave the network randomly, the newly joined nodes would not have the data disseminated earlier. These two issues are addressed by the Trickle algorithm [9], which is the building block of most dissemination protocols. According to the algorithm, each node periodically broadcasts a summary of the data that it has unless it has recently heard an identical summary. Once the data in all nodes is consistent, the broadcast interval is exponentially increased (up to a maximum value) to save energy. On the other hand if a node detects that other nodes have new data, it starts reporting more quickly. If a node hears an older summary. it returns an update to the sender about it. Thus, after the base station has injected new data to the nodes within its transmission range, they will be disseminated quickly because those nodes advertise that they have an update with the smallest broadcast interval. Upon hearing the advertisement, their neighboring nodes respond quickly that they want the update. This process propagates outward until all nodes in the network have the update.

Since redundant advertisements by each node are suppressed, the Trickle algorithm enables dissemination protocols to scale well with the number of nodes. However, for nodes to compare the freshness of a data item, each node must either transmit or receive a summary of the data item. A summary of a data item can be represented by a tuple (*key*, *version*), where *key* uniquely identifies a data item and *version* indicates if the data item is old or new (the higher the version number, the newer the data). Therefore, Trickle scales with $O(T)$, where T is the total number of data items. Drip [5] is one of the earliest small data dissemination protocols. It disseminates each data item with a separate instance of Trickle. This means that Drip also scales linearly with T. To overcome this inefficiency problem, Lin *et al.* proposed DIP [6]. In DIP, a summary message is periodically broadcast by a node that only contains hashes of keys and version numbers. The use of hashing enables differences to be detected with $O(1)$. However, once a difference is detected, multiple iterations are needed to pinpoint the exact data items that need to be updated. Overall, if there are N new items, the total number of transmitted messages required to identify which items have newer versions is $O(N \log(T))$. Later on, DHV [7] was proposed to further reduce the complexity. The development of DHV was motivated by the observation that if two versions are different, most likely only a few least significant bits of the version numbers are different. Therefore, DHV does not transmit and compare the complete version numbers, and hence reduces redundant transmission and processing. As a result, the total number of transmitted messages can be reduced to $O(N)$ only. The characteristics of various small data dissemination protocols are illustrated in Table 1.

Security Vulnerabilities in Small Data Dissemination

Although small dissemination protocols become more and more efficient, an important aspect of such protocols, security, has received little attention. This is because all existing small data dissemination protocols assume that WSNs are used/operated in benign environments. However, in practice the operating environment could be hostile and both external and internal attacks could exist.

External attacks include eavesdropping, injection of forged messages, message replaying, impersonating valid sensor nodes, and wormhole attacks. Because there is no mechanism to authenticate the data items distributed to the nodes, an adversary may exploit this vulnerability to reboot the entire network with wrong data, or erase an important data from all nodes. For instance, by simply injecting a fake data item (*key*, *version*, $data_i$), an adversary can corrupt all those data in the network identified by *key*, as long as *version* is larger than all version numbers of those data. Similarly, to erase the data with *key* on all nodes, the adversary just needs to send a fake data item (*key*, *version*, *null*) using a small data dissemination protocol, where *version* is a large enough number. Moreover, since the disseminated data has no security protection, they can be intercepted easily by adversaries.

Since Trickle requires each node to broadcast its new data, an adversary can exploit this feature to launch denial-of-service (DoS) attacks to drain the limited resources (e.g. battery power, memory) of sensor nodes. For example, an adversary can gradually increase the version number of the data with a specific key and quickly inject fake data items to the network. Under Trickle any receiving node would accept these data items and then broadcast them. As a result,

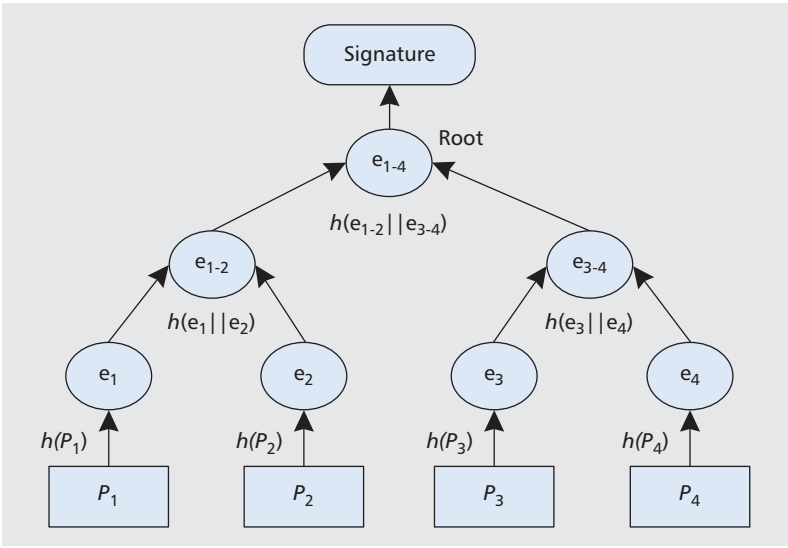

Figure 1. Example of the Merkle hash tree.

the adversary forces all nodes to waste energy in updating data and broadcasting fake data items. As most small data dissemination protocols reliably deliver data to each node based on some variants of Trickle, they are vulnerable to such a DoS attack.

Besides external attacks, adversaries can compromise a node to launch insider attacks in the network. Such an insider is regarded as a member of the network until it is identified and removed. It is manipulated by the adversary to attack other nodes. For example, it may launch Sybil attacks, disseminate false data, and violate the normal operation of protocols.

Required Security Functions

In view of the above issues, it is important to equip small data dissemination protocols with security functions without sacrificing their efficiency. Such functions should support data authenticity and integrity, data confidentiality, data freshness, dynamic data, semantic security, and provide resistance against node compromise attacks and DoS attacks. At the same time, the security functions should be communication- and computation-efficient, delay tolerant, and allow immediate authentication. These properties are discussed below in details.

- **Data authenticity and integrity:** Data authenticity provides a means to ensure that the data items are sent by the claimed sender. Also, data integrity ensures that the received data items have not been tampered with.
- **Data freshness:** Data freshness guarantees that the received data item is recent and not a replayed old message to cause disruption.
- **Dynamic data:** The sequence of broadcast data items need not to be known before-hand in its entirety by the base station.
- **Resistance to node compromise:** The protocol should be able to stand against a node compromise attack irrespective of the number of compromised nodes, as long as the non-compromised nodes still have connectivity with the base station.
- **Data Confidentiality:** In some critical applications the data items from the base station are strictly private and confidential. They should be encrypted to protect the data privacy from eavesdroppers. If the broadcast data items are chosen from a small finite set, the encryption should produce ciphertext that does not give information to an intruder on which of these messages was sent.
- **Delay tolerance and immediate authentication:** No time constraint should be imposed on packet arrival time. Each packet can be authenticated immediately when it has been received by the nodes.

SeDrip: The Protocol for Generic WSNs

Overview of SeDrip

In a brute force approach, each disseminated data can be digitally signed to ensure its authenticity. However, such an approach incurs too much computational loads for sensor nodes. Recently an efficiently secure extension of Drip, SeDrip, is proposed [10]. In its simplest form SeDrip ensures data authenticity and integrity. The essential idea of SeDrip is to make use of Merkle hash tree to significantly reduce the number of public key operations.

In the initialization phase of SeDrip the base station sets up the public and private keys based on elliptic curve cryptography (ECC), and distributes the public parameters to all sensor nodes in the network. Then before the base station disseminates n data items: $d_i = \{round, key_i, version_i, data_i\}$, $i = 1, 2, \ldots, n$, a Merkle hash tree is first built based on the n data items. We illustrate this process by considering $n = 4$ as an example. First, packet P_i, $i = 1, 2, \ldots, 4$, composed of data item $d_i(= \{round, key_i, version_i, data_i\})$ and the header information is created. Then hash values of packets $e_i = h(P_i)$ are used as the leaves of the Merkle hash tree, as shown in Fig. 1. Here $h(.)$ denotes a public one-way cryptographic hash function (e.g. SHA-1). The internal nodes in the next level are obtained as the hash values of two adjacent leave nodes. That is, referring to Fig. 1, $e_{1-2} = h(e_1||e_2)$ and $e_{1-4} = h(e_{1-2}||e_{3-4})$. This process continues until the root node is formed. Then P_i is appended with the siblings of the nodes in the path from e_i to the root node. For example, P_1 is appended with e_2, and e_{3-4}. Moreover, the base station generates a signature packet P_0 composed of the root data item $d_0 = \{round, key_0, version, e_{1-4}\}$, the signature $SIG_{SK}(h(d_0))$ and header information.

To disseminate the n data items, the base station first broadcasts P_0, advertising a new round of dissemination. This packet enables each sensor node to immediately verify the received data items upon their arrival, based on the nodes from the Merkle hash tree included in P_i. For example, given that e_{1-4} in P_0 has already been authenticated, a node can immediately verify d_1 contained in P_1 by checking if $h(h(h(d_1)||e_2)||e_{3-4}) = e_{1-4}$. Thus it can be seen that authentication of n data items mainly involves efficient hash operations, and only one public key operation.

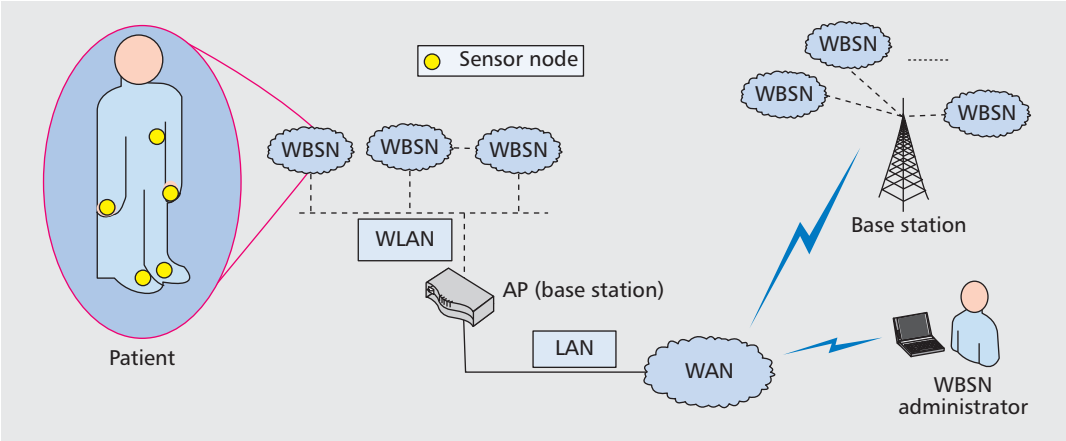

Figure 2. System architecture of a WBSN for mobile health monitoring.

With the assumption that the base station is trustworthy and cannot be compromised by adversaries, the private key signing the root of the Merkle hash tree is only known to the base station itself. Since all sensor nodes have the public key distributed by the base station, they can check the integrity of the data items contained in the received packets.

There are additional mechanisms in SeDrip that enable it to be DoS resistant and further improve the authentication efficiency. Readers are referred to [10] for details. Note that since the only difference between Drip, DIP, and DHV is the approaches taken to search for different data items among sensor nodes to achieve high message efficiency, the security extension in SeDrip is also applicable to DIP and DHV.

SeSmall: The Protocol for WBSNs

A WBSN is a special type of WSN that is used to monitor a patient's physiological parameters by body sensor nodes that are worn by and/or implanted on a patient. It is operated by the WBSN administrator (e.g. the patient himself, the patient's relative, the e-Health service provider, or the medical practitioner) through the base station, as shown in Fig. 2. A salient feature of WBSNs is that their sensor nodes, especially the implanted ones, have much less memory, lower processor speed, transmission speed, and energy supply than that of common WSNs. Thus secure small data dissemination protocols developed for generic WSNs may not be suitable for WBSNs. In this section we propose *SeSmall*, which is an enhancement of SeDrip.

The basic idea of SeSmall is that the authentication information of the ith data packet will be used to authenticate the $(i + 1)$th data packet. In this way the base station needs to sign just the first packet and then the properties of this single signature will "propagate" to the remaining data packets of the same round of dissemination through the chaining mechanism. Moreover, in SeSmall only one hash value of a packet is included in each packet. Compared to SeDrip, which includes D (the tree depth) hash values in each packet, SeSmall incurs much less communication overhead.

SeSmall consists of three phases: system initialization, packet pre-processing, and packet verification. In the system initialization phase the base station creates its public and private keys. Only the public parameters are loaded on each node before deployment. In the packet pre-processing phase, if the base station wants to disseminate data, it will need to construct the packets and then send them to the network.

System Initialization Phase

At this stage, in order to set up an ECC the base station picks a one-way cryptographic hash function $h(.)$ such as SHA-1, and derives a private key x and the corresponding public key y. After that the related public parameters are preloaded in each node of the network.

In order to provide security protection, we extend the 3-tuple (*key*, *version*, *data*) of the existing small data dissemination protocols into a 5-tuple (*round*, *seq*, *key*, *version*, *data*) to present a data item, where *round* refers to which round of data dissemination this data item belongs to (the higher the round, the newer the data dissemination), *seq* indicates the sequence number of this packet in this round, and the latter three elements bear the same meaning as existing protocols. Similar to the Drip implementation, *key* and *version* are two bytes and four bytes long, respectively. Additionally, the lengths of *round* and *seq* should be set to avoid any ambiguity due to wrap around in the number space. In our implementation, the lengths of *seq* and *round* are set to one and four bytes, respectively.

Packet Pre-Processing Phase

After system initialization, if the base station hopes to disseminate n data items: $d_i = \{round, seq_i, key_i, version_i, data_i\}$, $i = 1, 2, \ldots, n$, it uses the hash chain method to construct the packets of the respective data as follows.

Besides the packet header, a packet, say P_i, is composed of both d_i and cryptographic hash value $H_{i+1} = h(P_{i+1})$, which is used to verify the next packet. Therefore, each packet has a format of

$$P_i = \begin{cases} d_i \| H_{i+1} & \text{if } i = 1, \ldots, n-1 \\ d_i & \text{if } i = n. \end{cases}$$

Note that since the only difference between Drip, DIP, and DHV is the approaches taken to search for different data items among sensor nodes to achieve high message efficiency, the security extension in SeDrip is also applicable to DIP and DHV.

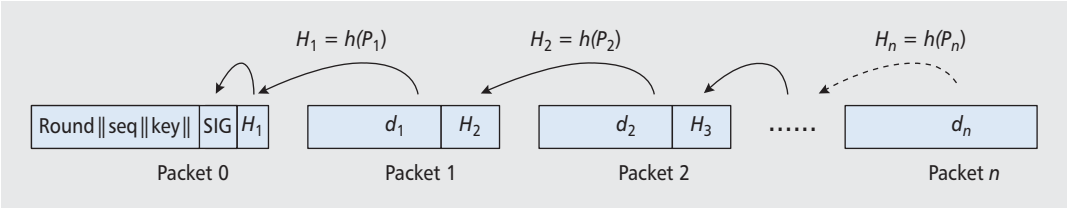

Figure 3. Hash chain method.

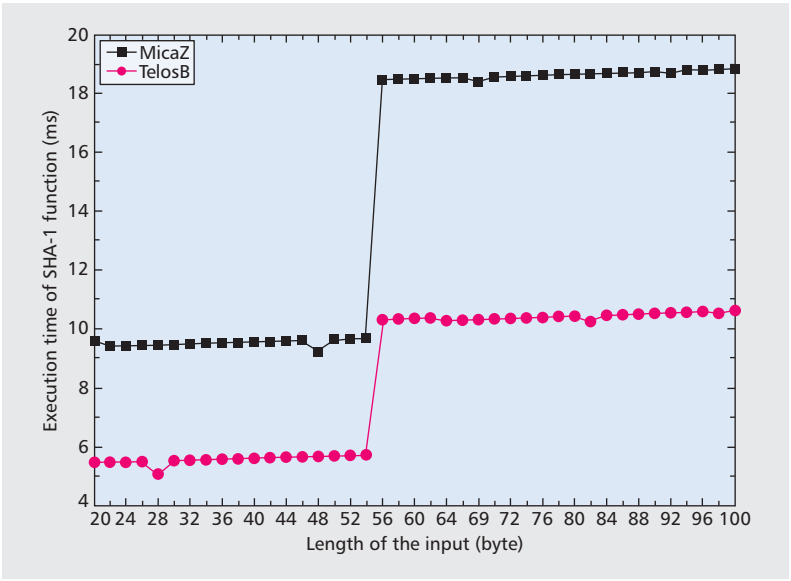

Figure 4. The execution times of SHA-1 hash function on MicaZ and TelosB motes.

Here each cryptographic hash H_i is calculated over the full packet i, not just the data portion d_i, thereby establishing a chain of hashes, as shown in Fig. 3.

Also, the base station signs the entire chain of hashes by signing the first hash H_1 with its private key x through running an ECDSA sign operation. More specifically, the *advertisement packet* P_0 includes the signature SIG of message $\{round||seq||key||H_1\}$. That is,

$$P_0 = round||seq_0||SIG||H_1,$$

in which SIG represents $SIG_k(h(round||seq_0|H_1))$, where $SIG_k(M)$ denotes the signature on message M with the key k. As in SeDrip, P_0 is used to advertise a new round of data dissemination. New data items are transmitted to the sensor nodes as a sequence of packets $(P_0, \ldots, P_n)$.

Packet Verification Phase

Upon receiving a packet (from any one-hop neighboring node or the base station), each sensor node, say S_j, first checks the *seq* field of the packet:

1. If this is an advertisement packet P_0, node S_j first checks whether the *round* in this packet is newer than its current round. If yes, node S_j uses the public key y of the base station to run an ECDSA verify operation to authenticate H_1. If the result is positive, node S_j updates its stored $<round, H_1>$ by the corresponding values in P_0; otherwise, node Sj simply discards the packet.
2. Otherwise (i.e. it is a *data packet* P_i, $i = 1, 2, \ldots, n$), node S_j can easily verify the authenticity and integrity of P_i with the previously received and verified H_i of the same round based on the one-way property of hash functions. If the result is positive and this is a new version, node S_j will update the data according to the *key* of this packet, and replace H_i of its stored $<round, H_i>$ with H_{i+1} for subsequent checking.

Implementation and Performance Evaluation

We have implemented SeSmall to evaluate its performance. The base station side programs are C programs using OpenSSL running on a 1.8-GHz laptop PC (with 2 GB RAM) under Ubuntu 11.04 environment. Also, the node side programs are written in nesC and run on TelosB motes. 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 TelosB motes run TinyOS [8] 2.x. Additionally, the key size of ECC is set to 160 bits and SHA-1 is used. All experiments on PCs (resp., sensor nodes) were repeated one million times (resp., one thousand times) for each measurement in order to obtain accurate average results.

Figure 4 shows the execution times of SHA-1 hash function (from TinyECC 2.0) on MicaZ and TelosB motes. The inputs to the hash function are randomly generated, with the length increasing from 20 to 100 bytes in increments of 2 bytes. Each point in the figure is the average of results obtained by repeating the same experiment ten thousand times. For example, the execution times on a MicaZ mote for inputs of 54 bytes, 56 bytes, and 100 bytes are 9.6788 ms, 18.4678 ms, and 18.8208 ms, respectively. Also, the execution times on a TelosB mote for inputs of 54 bytes, 56 bytes, and 100 bytes are 5.7263 ms, 10.3161 ms, and 10.6263 ms, respectively. From Fig. 4 it can be seen that the execution time is piecewise linear. It remains very stable when the byte length is within each of the following intervals: [20, 55] or [56,100]. Thus, it is suggested that in order to reduce the computation cost of SeSmall, the input length to the hash function should be chosen according to the above intervals.

To demonstrate that the use of hash chain is more efficient than public key cryptography for packet authentication, we have compared with the execution time of the 160-bit ECC algorithm from the TinyECC 2.0 library. It has been found that the signature verification time of 20-byte packets are 251 and 692 times longer than SHA-1 hash operation with 54-byte packets on MicaZ and TelosB motes, respectively.

The system initialization and signing a ran-

dom 20-byte message take 1.6080 ms and 0.6348 ms, respectively. As a comparison, even if the base station runs on a 800-MHz laptop PC, signing a 20-byte message only takes 1.83 ms.

To investigate the impact of security protection mechanism on the dissemination delay in multi-hop networks, we compare Drip and SeSmall in an indoor test-bed comprised of 24 TelosB motes organized in a 4 × 6 grid topology. The inter-node spacing is about 35 cm and the transmission power of each node is configured to the lowest power level in order to simulate the multi-hop behavior. In our implementation the base station (i.e. a laptop PC) first sends the signature and data packets through a serial port to the sensor node at the vertex of the grid, which is referred to as the *root node*. Then the root node disseminates these packets by Drip or SeSmall on behalf of the base station. For the implementation of Drip, each packet consists of a packet header and the data item <*version*, *key*, *data*>, where the length of the *data* is two bytes. For SeSmall, the lengths of *round* and *data* in each data item are four bits and two bytes, respectively. A hash function with 20-byte output is used.

Figure 5 shows the measured dissemination delays of Drip and SeSmall for various numbers of disseminated data per round, when the packet delivery rate at the base station is 20 packets/s. The dissemination delay of SeSmall is the time from construction of the data hash chain until the corresponding variables on all sensor nodes are updated. In order to obtain accurate average results, for each experiment we have repeated the dissemination operation 20 times and taken an average over them. It can be seen that as expected, the additional delay incurred by SeSmall is of the order of the verification time of one ECC signature. This is because upon receiving the advertisement packet, each intermediate sensor node forwards the packet to next-hop nodes before verifying the signature. Therefore, the extra delay incurred by SeSmall is mainly due to the verification time of one ECC signature at the last hop of dissemination, irrespective of the number of disseminated data as only one signature verification is executed in each round.

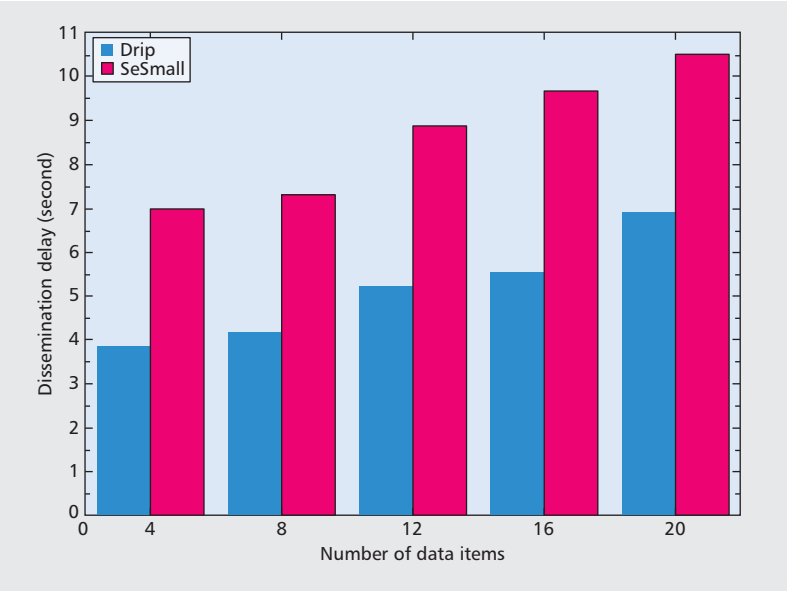

Figure 5. Propagation delay comparison of Drip and SeSmall.

Conclusion and Future Work

Being able to disseminate small data into a network is a useful operational function of WSNs. So far, research on small data dissemination protocols has focused on efficiency, but ignored the security aspects. In this article we have highlighted the security issues and reported a recently proposed solution, SeDrip. We have then enhanced the efficiency of the security functions of SeDrip to make it more suitable for WBSNs.

Nevertheless, work in this research area is just starting since neither standard benchmarks nor reference protocols of secure small data dissemination have been proposed so far. We conclude this article by suggesting the following future research ideas.

First, due to the open nature of wireless channels, messages can be easily intercepted. Clearly, data confidentiality should be taken into account in the design of secure small dissemination protocols.

Second, for large scale WSNs existing small data dissemination protocols are not efficient. This is because they all employ a centralized approach; all data items have to be disseminated by the base station. This kind of approach is vulnerable to the single point of failure. Whenever the base station is unavailable or the connection path is broken, it is impossible to carry out dissemination. Therefore, secure and distributed small data dissemination protocols should be investigated, which could allow users to directly disseminate data through nearby sensor nodes.

Acknowledgment

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.

References

[1] A. Marco *et al.*, "Synchronization of Multihop Wireless Sensor Networks at the Application Layer," *IEEE Wireless Commun.*, vol. 18, no. 1, Feb. 2011, pp. 82–88.
[2] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," *Proc. ACM SenSys*, 2004, pp. 81–94.
[3] D. He *et al.*, "DiCode: DoS-Resistant and Distributed Code Dissemination in Wireless Sensor Networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 5, May 2012, pp. 1946–56.
[4] D. Vukobratovic *et al.*, "Rateless Packet Approach for Data Gathering in Wireless Sensor Networks," *IEEE JSAC*, vol. 28, no. 7, Sept. 2010, pp. 1169–79.
[5] G. Tolle and D. Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks," *Proc. EWSN*, 2005.
[6] K. Lin and P. Levis, "Data Discovery and Dissemination with DIP," *Proc. IPSN*, 2008.
[7] T. Dang *et al.*, "DHV: A Code Consistency Maintenance Protocol for Multi-Hop Wireless Sensor Networks," *Proc. EWSN*, 2009.
[8] TinyOS: An Open-Source OS for the Networked Sensor Regime; available at: http://www.tinyos.net/

[9] P. Levis *et al.*, "Trickle: A Self-Regulating Algorithm for Code Maintenance and Propagation in Wireless Sensor Networks," *Proc. NSDI*, 2004.
[10] D. He *et al.*, "Secure Data Discovery and Dissemination based on Hash Tree for Wireless Sensor Networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, Sept. 2013, pp. 4638–46.

BIOGRAPHIES

DAOJING HE [S'07, M'13] (hedaojinghit@gmail.com) received the B.Eng. (2007) and M. Eng. (2009) degrees from Harbin Institute of Technology (China) and the Ph.D. degree (2012) from Zhejiang University (China), all in Computer Science. He is an associate professor in the School of Computer Science and Engineering, South China University of Technology, P.R. China. His research interests include network and systems security. He is an associate editor or on the editorial board of 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*.

SAMMY CHAN [S'87, M'89] (eeschan@cityu.edu.hk) received his B.E. and M.Eng.Sc. degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and a Ph.D. 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.

MOHSEN GUIZANI [S'85, M'89, SM'99, F'09] (mguizani @ieee.org) is currently a Professor and the Associate Vice President for Graduate Studies at Qatar University, Qatar. He received his B.S. (with distinction) and M.S. degrees in Electrical Engineering; M.S. and Ph.D. degrees in Computer Engineering in 1984, 1986, 1987, and 1990, respectively, from Syracuse University, Syracuse, New York. 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 he is 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 Senior Member of ACM.