Low-overhead authentication method for reprogramming protocol based on rateless codes in wireless sensor networks

Lina Yang[1,2], Shining Li[1], Yu Zhang[1] and Gang Liu[3]

[1]School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an, China

[2]School of Cryptographic Engineering, Information Engineering University, Zhengzhou, China

[3]School of Information Science and Engineering, Henan University of Technology, Zhengzhou, China

Email: yanglina@mail.nwpu.edu.cn, {lishining, zhangyu}@nwpu.edu.cn, liu2002gang@163.com

***Abstract*—Over-the-air reprogramming is a key service in wireless sensor networks. The service can disseminate a new code image to every sensor node in the network. For security reasons, every code image must be authenticated to prevent an attacker from installing its code to the network. In this paper, an authentication method named Hierarchical Hash Tree (HHT) is proposed to reduce the overheads of Sreluge, which is a reprogramming protocol based on rateless codes. HHT is a composed structure including two layers of Merkle Tree. The pages from code image are used to construct small hash trees in bottom. For reducing communication overhead, the roots of bottom trees are aggregated into root fingerprints, which are used to build top tree. Then, the security is analysed for proposed method and mathematical analysis are provided for the HHT overheads. Furthermore, we implement pages authentication using HHT in Sreluge. Experimental results show that our method can cut pages authentication overhead by at least half that of Sreluge for more than 3-KByte code image. And dissemination completion time of our method is about 60% that of Sreluge.**

***Keywords*—over-the-air reprogramming; wireless sensor networks; Hierarchical Hash Tree; root fingerprints; authentication overhead**

I. INTRODUCTION

Many applications exploit wireless sensor networks (WSNs) for long term data gathering, ranging from environmental sensing, industry monitoring to military operations, etc. In fact, the requirements for the network may change in time, or existing applications need to be modified or patched. Given the difficulty of physically redeploying sensor nodes, this requires new code image (a binary file obtained from newly compiled program) to be securely and remotely disseminated to all the nodes in the network, which is referred to as over-the-air reprogramming (OAP).

Recently, rateless codes are used in OAP because they have no limitation on the rate and are simultaneously near optimal for every erasure channel, such as Rateless Deluge [1] and Synapse [2]. The advantage of the reprogramming protocols using rateless codes is that they are resilient to packet loss. But an attacker still can pollute encoded packets, thereby executing the kind of denial-of-service attack known as pollution attack. Several security schemes [3]–[6] can provide authentication services to prevent an attacker from installing its code in WSN, but they work on original data packets. So, these schemes cannot ensure the security of the reprogramming protocols based on rateless codes as data packets are transmitted as encoded packets in the protocols.

For reprogramming protocols based on rateless codes, several security schemes [7]–[10] have been proposed to resist pollution attack. One of these schemes referred to as Sreluge [10] is a secure extension to Rateless Deluge. Sreluge employs a neighbor classification approach with time series forecasting and digital signature scheme based on Merkle Tree (MT) to resist pollution attack. This combinatorial technique can detect polluters with zero false negative rate and a negligible false positive rate. In this technique, nodes need to authenticate obtained pages after decoding encoded packets. When code image has more pages, the depth of MT is so long that pages authentication overhead would get very large, where the overhead is $O\left(N \log_2 N\right)$. The goal of this paper is to propose a method to reduce pages authentication overhead for Sreluge and its implementation is able to ensure the security of OAP.

Our contributions are as follows.

(a) We propose an authentication method referred to as Hierarchical Hash Tree that is a data structure composed of two layers of MT to reduce pages authentication overhead for Sreluge protocol.
(b) We analyse the security of proposed method to ensure code image securely disseminated to all the nodes in WSNs.
(c) We analyse theoretically the overheads of HHT. Analysis results show that HHT authentication overhead can be reduced to $O(N)$, in which N is the number of divided pages from code image.
(d) We implement pages authentication using HHT in Sreluge. Experimental results show that our method can cut pages authentication overhead by at least half that of Sreluge for more than 3-KByte code image.

The remainder of this paper is organized as follows. Section II discusses related work. Section III gives system model which this proposal is based on. Section IV describes our method in detail. Section V analyse the security of code image.

978-1-4799-0959-9/14/$31.00 ©2014 IEEE

Section VI makes theoretical analysis for the performance of Hierarchical Hash Tree. Section VII presents simulation results of our method. Finally, Section VIII concludes this paper.

II. Related Work

Secure OAP is critical in WSNs deployed in adversarial environments. Early reprogramming protocols use digital signature schemes to ensure data integrity and authenticity in the process of code dissemination. The technique common to these schemes is the use of hash function that binds a later packet to an earlier packet which ultimately leads to a digital signature signed by the base station, to facilitate authentication. Sluice [3] constructs a Hash chain and signs the head of the chain. Seluge [4] constructs a Merkle Tree and signs the tree root. These security schemes are designed based on the original data packets, while in network coding-based reprogramming protocols, data packets are transmitted as encoded packets.

Pollution attack is the main security threat for reprogramming protocols based on network coding. Several security schemes have been proposed to combat pollution attacks by using efficient solutions. The use of homomorphic hash functions (or signatures) is a common method for checking the integrity of encoded packets. However, it is computationally expensive to use such technique for sensor nodes. Dong et al. [7] propose a scheme referred to as DART that uses time-based authentication to resist pollution attack. The security of DART relies on time asymmetry, which may introduce additional delays to the scheme. Further, malicious nodes prevent legitimate nodes from receiving checksums, and hence force them to abandon legitimate packets. Yu et al.'s scheme [8] detects pollution by attaching multiple authentication tags to a message, thereby causing considerable overhead. Bohli et al. [9] propose a secure approach to address the problem of authenticating FC-coded data packets in Synapse. This approach can reduce decoding overhead but has high communication complexity. Sreluge [10] deals with the problem of authenticating encoded data packets using random linear codes in Rateless Deluge. In this method, pages authentication overhead grows quickly along with the increase of the pages number.

III. System Model

A. Assumptions and Threat Model

A WSN consists of a large number of resource-constrained sensor nodes and the base station that is the source of code images. The base station is a powerful node (e.g. a laptop) with sufficient energy supply. Sensor nodes are equipped with an IEEE 802.15.4 compliant radio transceiver, where the maximum packet payload size is 102 bytes [4]. Due to resource constraints, a typical sensor node can perform only a minimal number of public-key operations. We assume that the base station has a pair of keys (K_p, K_s) by running some key-generation algorithm, where K_p is the public key and K_s is the private key. Moreover, K_p is preconfigured to all sensor nodes. We assume Sreluge as the underlying reprogramming protocol.

The base station is the only trusted entity in the network and cannot be captured. Sensor nodes are not trusted. An attacker may introduce its own nodes into a network, or he (or she) may capture existing nodes in the network, as sensor nodes are generally not attack-resistant. However, an attacker cannot compromise an unlimited number of sensor nodes without being detected. The attacker attempts to launch pollution attack by injecting false messages or modifying maliciously some encoded packets in the network.

TABLE I. The notations used in the paper

notations	it denotes
$Hash(B)$	Hash of B
$A \parallel B$	A is concatenated with B
$Page_i$	the ith divided page from code image
$H_{1,i}$	hash value of the ith page as the leaf of a bottom tree in HHT
$H_{2,i}$	hash value of the ith root fingerprint as the leaf of top tree in HHT
$HHT(h, \theta)$	the structure of HHT
HT_{bottom}	bottom trees in HHT
HT_{top}	top tree in HHT
h	the height of HT_{bottom}
H	the height of HT_{top}
g_i	the ith tree root of HT_{bottom}
rf_i	the ith root fingerprint
θ	the number of g_i included in root fingerprint
N	the number of divided pages from code image
s	the data packet size in bytes
s_{hash}	the size of hash value in bytes

B. Sreluge Overview

Sreluge [10] is an over-the-air reprogramming protocol based on random linear codes. In Sreluge, a new code image is divided into pages, and a page is divided into packets. We assume that a page consists of m packets $X_1, X_2, ..., X_m$ and the packet size is s bytes. To send a page, the packets in a page are encoded as $Y_1, Y_2, ..., Y_n (n > m)$ according to

$$\begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,m} \\ \beta_{2,1} & \cdots & \beta_{2,m} \\ \vdots & \vdots & \vdots \\ \beta_{n,1} & \cdots & \beta_{n,m} \end{bmatrix} \begin{bmatrix} X_{1,k} \\ X_{2,k} \\ \vdots \\ X_{m,k} \end{bmatrix} = \begin{bmatrix} Y_{1,k} \\ Y_{2,k} \\ \vdots \\ Y_{n,k} \end{bmatrix} \quad (1)$$

In (1), $X_{i,k}$ denotes the kth byte of X_i and $\beta_{i,j}$ is pseudorandom number in Galois field $GF\left(2^8\right)$. The parameters $\beta_{i,j}$ of the encoding can be easily adjusted so that the rows $\beta_{i,1}, \beta_{i,2}, ..., \beta_{i,m}$ are linearly independent with high probability. Thus, any node that receives m of the Y_i can solve the linear equations to obtain X_i by means of Gaussian elimination.

Sreluge uses a pollution detection engine to identify the polluters among the neighbors, which is holden in this paper. In the process of pollution detection, obtained pages after decoding encoded packets need to be authenticated. Sreluge adopts digital signature scheme based on MT to authenticate these pages. MT is first constructed over hash values of the pages, then the MT root is signed using Elliptic Curve Digital Signature Algorithm (ECDSA) [11].

IV. Proposed Authentication Method

In this section, we describe our authentication method named Hierarchical Hash Tree in detail. The notations used in this paper are provided in Table I.

A. Hierarchical Hash Tree

In the run-up to the reprogramming, the base station partitions code image into N pages, then Hierarchical Hash Tree is constructed based on the pages.

Definition 1: The structure $HHT(h, \theta)$ of Hierarchical Hash Tree. Assuming N is the pages number, Hierarchical Hash Tree consists of two layers of Merkle Tree, bottom trees (HT_{bottom}) with the height h are constructed using the hash values of the pages as the leaves, the obtained HT_{bottom} roots are defined as g_i (i = 1,2,$\cdots$, $\lceil \frac{N}{2^h} \rceil$), the g_i are aggregated in θ blocks into root fingerprints rf_i ($i = 1, \cdots, \lceil \frac{N}{2^h \theta} \rceil$), where a root fingerprint is placed in a data packet to be transmitted together. Then, top tree (HT_{top}) is constructed using the hash values of rf_i as the leaves, and the HT_{top} root is denoted as HHT_root.

In $HHT(h, \theta)$, the range of h is $\{h | 1 \leq h \leq \lceil \log_2 N \rceil, h \in Z+\}$. The θ relates to the data packet size (s) and the size of hash value (s_{hash}). The range of θ is $\left\{\theta \left| 2 \leq \theta \leq \left\lfloor \frac{s}{s_{hash}} \right\rfloor , \theta \in Z+ \right.\right\}$.

According to Definition 1, Hierarchical Hash Tree is constructed as follows:

Step 1 $\boldsymbol{HT_{bottom}}$. Hash operations are done for the pages, then we have got the values $H_{1,i} = Hash(Page_i)$ (i = 1, 2 ,$\cdots$, N). MT is a complete binary tree [12], so HT_{bottom} are built using $H_{1,i}$ as the leaves of the tree according to

$$W = Hash(l \parallel r) \tag{2}$$

In (2), W represents parent node, l represents the left child node, and r represents the right child node. Each parent node is equal to the hash of the concatenation for the hash values stored in the children's nodes, such as g_1 = $Hash($ $H_{1,1}$ $\parallel$ $H_{1,2}$), and it proceeds recursively until reaching the top of the HT_{bottom}.

Step 2 **Root fingerprints**. For reducing communication overhead, root fingerprints are obtained by the HT_{bottom} roots aggregated in θ blocks. Thus, for $i = 1, \cdots, \lceil \frac{N}{2^h \theta} \rceil$, we have $rf_i = g_{\theta(i-1)+1} \parallel g_{\theta(i-1)+2} \parallel \cdots \parallel g_{\theta i}$.

Step 3 $\boldsymbol{HT_{top}}$. Hash operations are done for root fingerprints, then we have got the values $H_{2,i} = Hash(rf_i)$ (i = 1, 2, $\cdots$, $\lceil \frac{N}{2^h \theta} \rceil$). HT_{top} is built using $H_{2,i}$ as the leaves of the tree according to (2), such as $HHT_$root $= Hash\,(H_{2,1} \parallel H_{2,2})$, and it proceeds recursively until reaching the top of the HT_{top}.

After HHT constructed, its authentication paths can be obtained. HHT authentication paths are transmitted in index packets. The size of index packet relates to the packet payload size of node platform and the size of hash value. Fig. 1 illustrates an example of $HHT(1, 4)$ using the height $h = 1$ of bottom trees and aggregation degree $\theta = 4$. Fig. 2 shows authentication paths of $HHT(1, 4)$. The verificaton of root fingerprints must precede that of all pages. For example, in order to authenticate $Page_1$, root fingerprint rf_1 containing g_1, g_2, g_3 and g_4 must be verified first using $H_{2,2}$. The authentication path of each page contains one hash since the height of HT_{bottom} is one, similarly the authentication path of each root fingerprint contains one hash as the height of HT_{top} is one.

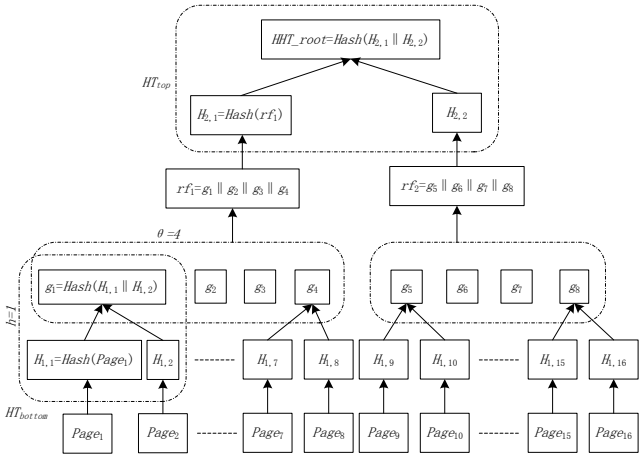

Fig. 1. Hierarchical Hash Tree using the height $h = 1$ of bottom trees and aggregation degree $\theta = 4$.

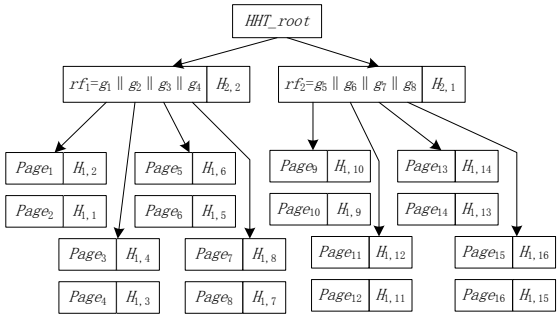

Fig. 2. Authentication paths of $HHT(1, 4)$.

B. Digital Signature for the HHT Root

The base station needs to sign the HHT root in order that the root can be transmitted securely to all the nodes in WSNs. Elliptic Curve Cryptography (ECC) has been the good choice among various Public Key Cryptography (PKC) options due to its fast computation, small key size, and compact signatures. TinyECC can provide an open software package for PKC operations based on ECC that can be flexibly configured and integrated into WSNs. Moreover, it takes 2 seconds to generate ECDSA signature using TinyECC, and a MicaZ mote can implement a 160-bit ECDSA signature verification operation in about 2.4 seconds using TinyECC [11]. Thus, the base station signs the HHT root with the private key K_s using ECDSA module from TinyECC, which is similar to that of Sreluge.

C. Dissemination of Code Image

When a new code image needs to be disseminated, the base station broadcasts first a command (CMD) packet globally to announce the availability of new image. The CMD packet includes a dissemination command, meta-information about new code image, and authentication information for ensuring data packets. The meta-information contains version number, the size of code image, etc. The authentication information includes the HHT root and the signature for the root. After broadcasting CMD packet, the base station propagates HHT

authentication paths as index packets. Then, it transmits encoded packets for each page by page-by-page propagation.

V. Security of Code Image

A. Page Authentication

Upon receiving a CMD packet, each node decrypts first the signature in the packet using the public key K_p to authenticate the HHT root HHT_root. The root allows the node to authenticate root fingerprints upon receipt, using the values in index packets. For example, if HHT_root has been authenticated in the CMD packet, upon receiving the index packet $\{rf_1, H_{2,2}\}$, a node can verify whether $Hash\left(Hash\left(rf_1\right) \parallel H_{2,2}\right) \stackrel{?}{=} HHT\_root$. If yes, the received index packet is accepted; otherwise, it must be a forged packet and should be discarded. To continue the above instance, the receipt of an authenticated rf_1, where $rf_1 = g_1 \parallel g_2 \parallel g_3 \parallel g_4$, implies the correct receipt of g_1, g_2, g_3 and g_4. Since root fingerprints include all the HT_{bottom} roots, successful receipt of them allows the node to further authenticate primary page data.

After the m encoded packets from a page received by a node, the node decodes these packets to recover the primary page data by means of Gaussian elimination. Then, the node can verify whether the page is correct according to corresponding index packet. For example, after the index packet $\{Page_1, H_{1,2}\}$ received, a node can verify whether $Hash\left(Hash\left(Page_1\right) \parallel H_{1,2}\right) \stackrel{?}{=} g_1$ upon obtaining $Page_1$ by decoding. If yes, the page is accepted and should be transmitted again after recoded; otherwise, it must be a forged page and should be discarded.

B. Security Analysis

The security objective for proposed method is to prevent an attacker from substituting the encoded packet(s) with its own packet(s). The security proof is reduced to the security proof of the digital signature scheme and HHT, which are existentially unforgeable under known-message attacks. Usually, a signature scheme is required to satisfy a stronger security notion: existentially unforgeable under chosen-message attacks. An attacker cannot choose the messages to be signed as the base station is trusted in the threat model, so an attacker can at most only launch known-message attacks.

In this paper, the HHT root is signed by the base station using ECDSA. ECDSA has been proven existentially unforgeable under chosen message attacks in a generic group model by Brown [13]. While the generic group model used in Brown's proof is invalid for supersingular elliptic curves, the model is valid for the curves defined in the standard. Since HHT consists of two layers of Merkle Tree, root fingerprint and each page can be authenticated based on the security of Merkle Tree. Merkle Tree has been proven existentially unforgeable under chosen message attacks [14]. Therefore, it can be concluded here that if both ECDSA and Merkle Tree are secure, then the component for pre-dissemination process of code image and the component for pages authentication of proposed scheme are provably secure.

VI. Performance Analysis for HHT

A. Computation Overhead

HHT computation overhead includes construction overhead and authentication overhead. HHT is constructed by the base station in the run-up to code dissemination, and HHT needs to be verified in each node. Authentication overhead should be as little as possible because computing capability of nodes is relatively weak.

1) Construction Overhead: The hash operation time required for constructing HHT is defined as hht_hash_num. It is the product between the HT_{bottom} number and the sum of hash operations on each height in HT_{bottom} plus the sum of hash operations on each height in HT_{top}. Thus, hht_hash_num is:

$$\begin{aligned} hht\_hash\_num &= \frac{N}{2^h}\sum_{i=0}^{h} 2^i + \sum_{i=0}^{H} 2^i \\ &= \frac{N}{2^h}\left(2^{h+1}-1\right) + \left(2^{H+1}-1\right) \\ &= (2N-1) - 2^H(\theta-2), \end{aligned} \tag{3}$$

where $H = \left\lceil \log_2 \frac{N}{2^h\theta} \right\rceil$.

Likewise, the hash operation time required for constructing MT is defined as mt_hash_num. It is the sum of hash operations on each height in MT. Then, mt_hash_num is:

$$mt\_hash\_num = \sum_{i=0}^{H_{MT}} 2^i = 2^{H_{MT}+1} - 1 \ = 2N-1, \tag{4}$$

Where N is the pages number from code image and H_{MT} is the height of MT.

Definition 2: Construction overhead of hash tree. The hash operation time required for constructing a hash tree is defined as construction overhead of the hash tree.

From (3) and (4), the conclusion can be obtained that construction overhead of HHT is less than that of MT where $2 < \theta \leq \left\lfloor \frac{s}{s_{hash}} \right\rfloor$. Only when $\theta = 2$, will construction overhead of HHT be the same as that of MT.

2) Authentication overhead: HHT authentication overhead (O_{AU}) has two components associated, root fingerprint overhead (O_{RF}) and pages authentication overhead (O_{Page}). The O_{RF} is also the authentication overhead of HT_{top}, and it is the product between the number of leaf nodes and the size of the authentication paths for HT_{top}. The O_{Page} is also the authentication overhead of HT_{bottom}, and it is the product between the number of HT_{bottom} and authentication overhead for each HT_{bottom}, which is the product between the number of leaf nodes and the size of the authentication paths for HT_{bottom}. Thus, the O_{AU} is:

$$\begin{aligned} O_{AU} &= 2^H H + 2^h h\left\lceil \frac{N}{2^h} \right\rceil \\ &= \frac{N}{2^h\theta}H + Nh \\ &= O(N). \end{aligned} \tag{5}$$

When $N \gg h$, the O_{AU} grows $O(N)$ in (5). The maximum value of h is $\lceil \log_2 N \rceil$, reducing the HHT to a regular Merkle Tree with authentication overhead of $O\left(N \log_2 N\right)$.

B. Storage overhead

Storage overhead is composed of the public key K_p needed to verify the signature for the HHT root, and authentication paths of HHT required to authenticate code image. We assume that s_p denotes the size of K_p in bits. The size of the public key K_p to be stored is $\frac{s_p}{8}$ bytes. The public key K_p needs to be stored on the nonvolatile EEPROM memory which can keep its data even after a battery change or a reset. Authentication paths of HHT need to be stored on RAM, which include authentication paths of pages, root fingerprints and its authentication paths. The number of hash values contained in authentication paths of pages is equal to the product between the pages number and the height of HT_{bottom}. Likewise, the number of hash values contained in authentication paths of root fingerprints is equal to the product between the number of root fingerprints and the height of HT_{top}. Thereby, storage overhead for HHT in bytes is

$$\left(\left\lceil \frac{N}{2^h\theta} \right\rceil \left\lceil \log_2 \frac{N}{2^h\theta} \right\rceil + \left\lceil \frac{N}{2^h} \right\rceil + Nh\right) s_{hash} \quad (6)$$

C. Communication overhead

Communication overhead is composed of the overheads for the signature and index packets of HHT required to verify code image. Since both the signature and the HHT root are included in CMD packet, one CMD packet need to be broadcast. Index packets of HHT need to be transmitted before encoded packets to be broadcast. The total number of hash values contained in authentication paths of HHT (obtained from (6)) divided by the number of hash values contained in an index packet, so we can get the index packets number required for HHT, which is defined as HHT communication overhead. Given packet loss rate p caused by link quality, then HHT communication overhead is

$$\left(\left(\left\lceil \frac{N}{2^h\theta} \right\rceil \left\lceil \log_2 \frac{N}{2^h\theta} \right\rceil + \left\lceil \frac{N}{2^h} \right\rceil + Nh\right) \div \left\lfloor \frac{s}{s_{hash}} \right\rfloor\right)(1+p)$$

VII. Performance Evaluation

In this section, we make the following setups: each page has data packets of $m = 8$ for code image; the payload size of each data packet is 21 bytes; the number of encoded packets takes $n = 1.5m$ [15]; the 32-bit truncation of SHA-1 is used as hash function. In $HHT(h, \theta)$, We can optimize the data structure to reduce the total overhead by selecting the height h of HT_{bottom} and making θ maximum. The θ is $\lfloor \frac{21}{4} \rfloor = 5$, which is limited to the maximum packet payload size.

A. Comparison with computation overhead

Computation overhead of hash tree consists of construction overhead and authentication overhead. HHT (or MT) is constructed in the base station and needs to be verified in each node.

Fig. 3 shows reduction ratio η for construction overhead of HHT compared to that of MT by selecting different height h in $HHT(h, \theta)$. From Fig. 3, it is concluded that construction overhead of HHT is less than that of MT when $\eta > 0$. This is because HHT structure reduces the depth of the tree. And the curves are non-linear since $2^h | N$ is always true and $\log_2 \frac{N}{2^h\theta}$ is not always integer. For example, construction overhead of

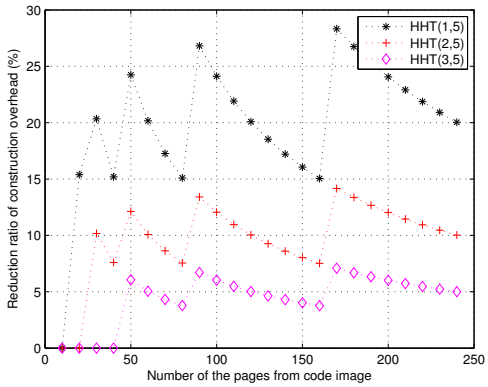

Fig. 3. Reduction ratio for construction overhead of HHT compared to that of MT.

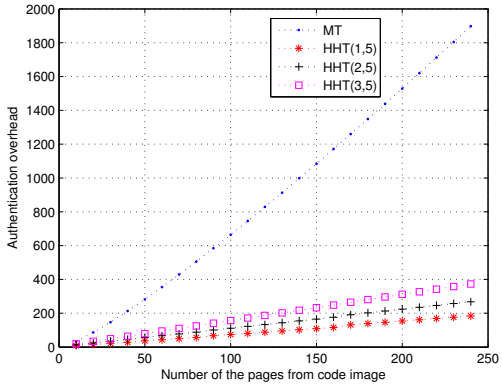

Fig. 4. Authentication overhead of MT and HHT.

$HHT(1, 5)$ is 20% that of MT for 20-KByte code image. When the pages number is less than 20 for $HHT(1, 5)$, 30 for $HHT(2, 5)$, and 50 for $HHT(3, 5)$, we find $\eta = 0$. This means that HHT construction overhead has no reduction here. Moreover, HHT construction overhead varies with the height h.

Fig. 4 shows authentication overhead of MT and HHT with different height h. From Fig. 4, it is concluded that authentication overhead of HHT is less than that of MT no matter what the h is. For example, authentication overhead of $HHT(1, 5)$ is 78% that of MT for 20-KByte code image. The more the size of code image grows, the more HHT authentication overhead reduces.

B. Comparison with communication overhead

We consider only communication overhead caused by hash tree itself. We assume packet loss rate $p = 5\%$. We compare communication overhead of HHT to that of MT, where $HHT(1, 5)$ is selected as the structure of HHT, as are shown in Table II. In Table II, we use the following notations: $CIsize$ denotes the size of code image; CO_MT and CO_HHT denotes communication overhead for MT and HHT respectively. RR denotes reduction ratio for communication overhead of HHT compared to that of MT, that is $\frac{CO\_MT - CO\_HHT}{CO\_MT}$.

TABLE II. COMPARISON OF COMMUNICATION OVERHEAD

pages	CI size KByte	CO_MT	CO_HHT	RR %
30	4.9	32	11	65.6
60	9.8	74	23	68.9
90	14.8	128	36	71.9
120	19.7	170	48	71.8
150	24.6	242	60	75.2
180	29.5	290	72	73.8

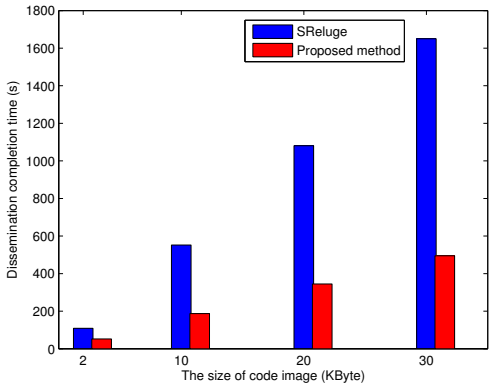

Fig. 5. Comparison of time to complete code dissemination.

From Table II, it is concluded that HHT communication overhead reduces by at least 66% that of MT when the pages number is greater than 30. For example, when the pages number is 120, only 48 index packets need to be transmitted for HHT, but 170 packets for MT. This means that 71.8% packets over HHT need to be transmitted for MT. Moreover, the reduction of HHT communication overhead is more obvious with the increase of the size of code image.

C. Dissemination completion time

Dissemination completion time is defined as the time required for a code image disseminated to every node in a WSN. The time is estimated using TOSSIM in TinyOS 2.x. We simulate a 5×5 grid network, with node spacing of 5m and path loss exponent (rate at which signal decays) of 4.0. In 20% pollution rate, we compute average time by repeating 10 times of dissemination processes using Sreluge and its improvement from the use of HHT, as shown in Fig. 5.

From Fig. 5, it is concluded that dissemination completion time of two protocols grow almost linearly with the increase of the size of code image. Moreover, dissemination completion time using proposed method reduces by at least 52% that of Sreluge. For example, the time is 68% that of Sreluge for 20-KByte code image.

VIII. CONCLUSION

In this paper, we pay special attention to the overheads of Sreluge protocol. To reduce the overheads, we propose an authentication method named Hierarchical Hash Tree. We can optimize Hierarchical Hash Tree by selecting reasonably the parameters h and θ. Moreover, HHT security is analysed and mathematical analysis of HHT overheads are provided. HHT authentication overhead can be reduced to $O(N)$. By experimental analysis, we find that the overheads of our method are less than that of Sreluge and the reductions of the overheads are more obvious to the larger-sized code image.

ACKNOWLEDGEMENT

This work is supported by National Science and Technology Major Project of China (Grant No. 2012ZX03005007), National Natural Science Funds of China (Grant No. 61174056), Natural Science Funds of Shaanxi Province of China (Grant No. 2013JQ8041), and NPU Funds for Fundamental Research (Grant No. JC20110268).

REFERENCES

[1] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes," in *Proc. IEEE/ACM IPSN'08*, 2008, pp. 457–466.

[2] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. F. Harris, and M. Zorzi, "Synapse: A network reprogramming protocol for wireless sensor networks using fountain codes," in *Annu. IEEE SECON'08*, 2008, pp. 188–196.

[3] P. E. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure dissemination of code updates in sensor networks," in *Proc. IEEE ICDCS'06*, 2006, pp. 53–63.

[4] S. Hyun, P. Ning, A. Liu, and W. L. Du, "Seluge: Secure and dos-resistant code dissemination in wireless sensor networks," in *Proc. IEEE/ACM IPSN'08*, 2008, pp. 445–456.

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

[6] E. Ayday and F. Fekri, "A secure broadcasting scheme to provide availability, reliability and authentication for wireless sensor networks," *Elsevier Ad Hoc Netw.*, vol. 10, no. 7, pp. 1278–1290, 2012.

[7] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proc. ACM WiSec'09*, 2009, pp. 111–122.

[8] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing xor network coding against pollution attacks," in *Proc. IEEE INFOCOM'09*, 2009, pp. 406–414.

[9] J.-M. Bohli, A. Hessler, O. Ugus, and D. Westhoff, "Security enhanced multi-hop over the air reprogramming with fountain codes," in *Proc. IEEE LCN'09*, 2009, pp. 850–857.

[10] Y. W. Law, Y. Zhang, J. Jin, M. Palaniswami, and P. Havinga, "Secure Rateless Deluge: Pollution-Resistant Reprogramming and Data Dissemination for Wireless Sensor Networks," *Eurasip J. Wireless Commun. Networking*, pp. 1–21, 2011.

[11] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. IEEE/ACM IPSN'08*, 2008, pp. 245–256.

[12] W. Wong, M. F. Magalhes, and J. Kangasharju, "Piece fingerprinting: Binding content and data blocks together in peer-to-peer networks," in *Proc. IEEE GLOBECOM'10*, 2010, pp. 1–6.

[13] D. R. L. Brown, "Generic groups, collision resistance, and ECDSA," *Designs, Codes, and Cryptography*, vol. 35, no. 1, pp. 119–152, 2005.

[14] L. C. C. Garca, "On the security and the efficiency of the Merkle signature scheme," *Cryptology ePrint Archive: Report*, pp. 192–211, 2005.

[15] J. W. Li, S. N. Li, Y. Zhang, Y. W. Law, X. S. Zhou, and M. Palaniswami, "Analytical model of coding-based reprogramming protocols in lossy wireless sensor networks," in *Proc. IEEE ICC'13*, 2013, pp. 1867–1871.