# LR-Seluge: Loss-Resilient and Secure Code Dissemination in Wireless Sensor Networks

Rui Zhang and Yanchao Zhang
*School of Electrical, Computer, and Energy Engineering*
*Arizona State University*
*Tempe, AZ 85287*
{*ruizhang,yczhang*}@*asu.edu*

***Abstract*—Code dissemination in wireless sensor networks refers to the process of disseminating a new code image via wireless links to all sensor nodes after they are deployed. It is desirable and often necessary due to the need for, e.g., removing program bugs and adding new functionalities in a multi-task sensor network. A sound code dissemination scheme need be both loss-resilient and attack-resilient, which are crucial for sensor networks deployed in lossy and hostile environments. To the best of our knowledge, no existing scheme simultaneously satisfies both requirements. This paper fills this gap with the design and evaluation of LR-Seluge, a novel loss-resilient and secure code dissemination scheme. The efficacy and efficiency of LR-Seluge are confirmed by both theoretical analysis and extensive simulation results. In particular, LR-Seluge can reduce up to 40% communication overhead in lossy environments with the same level of attack resilience in contrast to existing schemes.**

***Keywords*-Secure; loss-resilient; code dissemination; sensor networks;**

## I. Introduction

Code dissemination [1] or *over-the-air reprogramming* [2] in wireless sensor networks refers to the process of disseminating a new code image via wireless links to all sensor nodes after they are deployed. It is desirable and often necessary due to the need for, e.g., removing program bugs and adding new functionalities in a multi-task sensor network [3].

A sound code dissemination scheme faces two critical challenges. First, wireless channels are lossy in nature, especially for sensor networks deployed in remote and harsh environments. Packets may get lost during transmission due to many reasons such as RF interference and environmental factors [4]. This calls for loss-resilient code dissemination schemes like [2], [5], [6] that can withstand high packet losses. Second, sensor networks in hostile environments such as the battlefield may be attacked. In particular, the adversary may exploit the code dissemination mechanism to launch various attacks. For example, the adversary may inject bogus code images to take over the control of the whole sensor network or launch the Denial-of-Service (DoS) attack by transmitting many bogus image packets to deplete the limited energy and/or buffer of sensor nodes. This situation necessitates secure code dissemination schemes such as Seluge [7] which provides code-image integrity and DoS attack resilience through immediate and efficient packet authentication.

To the best of our knowledge, no existing code dissemination scheme satisfies loss resilience and attack resilience at the same time. On the one hand, all existing secure code dissemination schemes such as [7]–[12] are secure versions of Deluge [1], the de facto non-secure code dissemination scheme. Deluge [1] relies on Automatic Repeat Request (ARQ) protocols for reliable broadcast transmissions, in which each local broadcast receiver uses NACKs to request retransmission of lost packets. ARQ protocols, however, are generally not suitable for broadcasting in highly lossy networks due to too many retransmissions and the accompanying high latency [4]. On the other hand, most loss-resilient code dissemination schemes employ rateless erasure codes such as Fountain codes [6] and random linear codes [2] at the sender side to cope with packet losses. The common idea is to encode the code image into an unlimited number of packets such that each local receiver can recover the code image after receiving sufficiently many packets. It is unfortunately infeasible to use the similar methods as in Seluge [7] to realize immediate and efficient authentication of potentially unlimited erasure-coded packets. There is a clear gap between these two lines of research.

In this paper, we fill the this gap by LR-Seluge, a novel loss-resilient and secure code dissemination scheme for sensor networks deployed in hostile and lossy environments. As the first of its kind, LR-Seluge is aimed to strike a good balance between loss resilience and immediate packet authentication by using a limited number of predetermined redundant packets. More specifically, we encode the code image using a fixed-rate erasure code and carefully create chaining relationships between original and encoded packets using lightweight cryptographic hash functions. This design allows sensor nodes not only to recover the original code image from a subset of the encoded packets but also to efficiently authenticate any encoded packet upon its arrival, thus simultaneously achieving sufficiently high loss resilience and attack resilience.

Our main contributions are summarized as follows.

- We notice the lack of a sound code dissemination scheme for sensor networks in lossy and hostile environments with satisfactory loss resilience and attack resilience and fill this void by proposing LR-Seluge.
- We theoretically analyze and compare the performance of LR-Seluge with and Seluge [7], a representative secure code dissemination scheme that ensures code-image integrity and provides very strong resilience to DoS attacks. We further confirm the efficacy and efficiency of LR-Seluge by extensive simulation results. Our results reveal that LR-Seluge can save up to 40% in communication overhead and 40% in code-dissemination latency with the same level of attack resilience in comparison with Seluge.

## II. Background

This section introduces the background knowledge necessary for understanding the design of LR-Seluge.

### A. Deluge

Deluge [1] is the de facto code dissemination paradigm for sensor network, which is also one of the standard components of the TinyOS distribution [13].

Deluge employs a page-by-page dissemination strategy, in which a large code image is divided into smaller fixed-size pages, and each page is further divided into same-size packets. A sensor node requests for a new page only after completely receiving all the packets of the previous page. During the code dissemination process, each node works in one of the following three states: *maintenance*, in which node periodically advertises the version of its code image and the number of pages it has for that version, *receiving*, in which the node actively requests the remaining packets to complete the current page using Selective NACK (SNACK for short) requests, and *transmitting*, in which the node broadcasts all the requested packets of the current page and continuously serves any subsequent request.

Deluge employs various suppression techniques to maximize the effect of overhearing and reduce packet collisions. For example, a node suppresses its own request (or data) packet if overhearing request (or data) packets for a page with the same or smaller indices. For lack of space, we refer readers to [1] for the details about these suppression techniques.

### B. Seluge

Seluge [7] is an exemplary secure code dissemination scheme, which ensures code-image integrity and DoS attack resilience by enabling efficient and immediate packet authentication.

Seluge integrates three different techniques to realize efficient and immediate packet authentication. First, Seluge chains the packets of two adjacent pages with a cryptographic hash function. For example, the hash image of the $j$th packet of page $i$ is embedded into the $j$th packet of page $i-1$. Since Seluge adopts the same page-by-page dissemination strategy from Deluge, any packet of page $i$ can be authenticated upon arrival because all the packets of page $i-1$ must have been received. Second, to authenticate the first page, Seluge introduces a special *hash page* formed by concatenating all the hash images of packets of the first page. A Merkle hash tree is built on top of the hash page, and the base station digitally signs the root of the Merkle hash tree to ensure its integrity. In this way, each packet of the hash page can be authenticated by computing the hash images along the path to the tree root. Finally, to prevent the adversary from transmitting a large number of signature packets to force sensor nodes to perform computationally expensive signature verifications, Seluge let the base station attach a weak authenticator to the signature packet, which is a *message specific puzzle* [14]. Only if the weak authenticator is valid do sensor nodes verify the signature. Seluge does not work well in lossy environments due to its dependence on Deluge, as to be shown later.

### C. Erasure Code

A $k$-$n$-$k'$ erasure code transforms $k$ packets into $n \geq k$ encoded packets of the same length such that the original $k$ packets can be recovered from any $k'$ encoded packets. Typical erasure codes include Reed-Solomon codes, Tornado codes, Raptor codes, Online codes, and LT codes, for which a good survey can be found in [15].

## III. Network/Adversary Models and Design Goals

### A. Network Model

As in Seluge [7], we consider a WSN consisting of a base station and many sensor nodes densely deployed in hostile environments such as the battlefield. The base station has a new code image $\mathcal{M}$ to be disseminated to all the sensor nodes via wireless links. We assume that the base station has abundant resources in computation and communication and is safeguarded from attacks. We also assume that the base station has a public/private key pair. In contrast, sensor nodes are more constrained in storage, energy, computation, and communication capabilities. We, however, adopt the same assumption in [7] that a sensor node can verify a few digital signatures, e.g., one signature verification per code image. It is worth noting that technical advances have rendered it quite feasible to execute once-daunting public-key operations on sensor nodes. For example, it takes 1.12s for a Tmote Sky mote to verify an ECDSA signature [16]. Such public-key operations will be minimized in LR-Seluge, as to be shown later. Moreover, we assume lossy and unreliable wireless channels such that packets may get lost due to many reasons such as environmental factors and accidental or malicious RF interference [4].

## B. Adversary Model

We also adopt the adversary model from [7]. In particular, we consider a computationally bounded adversary consisting of both *external* and *internal* attackers. External attackers do not belong to the target WSN, but they are capable of overhearing packet transmissions, injecting bogus packets, and replaying intercepted packets. Internal attackers are compromised yet undetected sensor nodes which are fully controlled by the adversary. We, however, follow the conventional assumption that non-compromised sensor nodes are always the majority.

Despite the many attacks the adversary can launch, this paper has the same target as Seluge [7] and focuses on defeating the following two attacks on code dissemination.

- The adversary may inject forged code images to take control of the sensor network.
- The adversary may send many fake packets or replay intercepted packets to force sensor nodes into wasteful packet processing so as to quickly deplete their limited energy and/or memory buffer.

## C. Design Objectives

In view of the two attacks outlined above, LR-Seluge is designed with the following goals in mind.

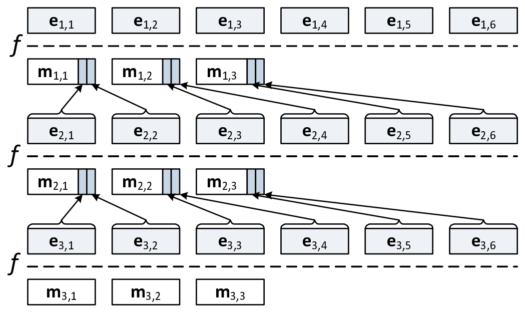
- **Code-image integrity**: Every sensor node is guaranteed to receive an authentic code image unless the node is isolated from the rest of the network.
- **DoS attack resilience**: Any forged packets should be immediately detected upon their arrivals, and packet authentication should be efficient.
- **Loss resilience**: LR-Seluge should enable more efficient code dissemination in the presence of severe packet losses than prior work such as Seluge [7].

## IV. LR-SELUGE DESIGN

## A. Overview of LR-Seluge

LR-Seluge is largely inspired by two observations. First, loss-resilient broadcasting can be achieved by using erasure codes at the sender side, i.e., introducing redundant packets, as demonstrated in [2], [4]–[6]. Second, immediate and efficient packet authentication can be realized if all the packets to be transmitted are predetermined. Consider Seluge [7] as an example, in which all the image packets are preprocessed to create some chaining relationships with cryptographic hash functions to enable immediate and efficient packet authentication. Such preprocessing can be done only if all the packets can be determined prior to transmission.

The key idea of LR-Seluge is to introduce a limited number of predetermined redundant packets to increase loss resilience and also achieve immediate packet authentication by carefully creating chaining relationships between encoded packets and original packets. More specifically, LR-Seluge differs from existing loss-resilient code dissemination schemes [2], [4]–[6] in that it uses a fixed-rate erasure code instead of rateless ones to encode the image packets. To enable immediate packet authentication, LR-Seluge creates some chaining relationships between the original packets of one page and the encoded packets of the next page. In this way, once a node receives sufficient encoded packets to recover one page, it also recovers all the hash images of the next page at the same time. This design differs from Seluge [7] in that there is no one-to-one correspondence between the packets of adjacent pages. Furthermore, LR-Seluge employs a simple but effective scheduling algorithm that allows each sender to transmit much fewer packets to ensure that every one-hop neighbor receives sufficient encoded packets to recover a code page.



e1,1 e1,2 e1,3 e1,4 e1,5 e1,6
f
m1,1 m1,2 m1,3
e2,1 e2,2 e2,3 e2,4 e2,5 e2,6
f
m2,1 m2,2 m2,3
e3,1 e3,2 e3,3 e3,4 e3,5 e3,6
f
m3,1 m3,2 m3,3

Figure 1. An example of code-image preprocessing for pages $\mathcal{M}_1$ to $\mathcal{M}_g$, where $g = 3, k = 3, n = 6$.

In what follows, we detail the operations of LR-Seluge, including *initialization*, *code-image preprocessing*, *efficient loss-resilient code dissemination*, and *authenticated packet transmission*.

## B. Initialization

Before the network deployment, the network owner preloads each sensor node with the following information.

- The same instance of a $k$-$n$-$k'$ erasure code $f(\cdot)$;
- The same instance of a $k_0$-$n_0$-$k'_0$ erasure code $f_0(\cdot)$;
- The public key of the base station;
- A public cryptographic hash function $H(\cdot)$.

With $f(\cdot)$ or $f_0(\cdot)$, every node can generate the same $n$ or $n_0$ encoded packets from the same $k$ or $k_0$ input packets.

## C. Code-Image Preprocessing

Assume that the base station has a code image $\mathcal{M}$ for dissemination to all sensor nodes. As in [1], [10], the base station partitions the original image $\mathcal{M}$ into $g$ pages of fixed size, denoted by $\{\mathcal{M}_i\}_{i=1}^{g}$. Each page $\mathcal{M}_i$ is then further divided into $k$ original blocks of equal length, i.e., $\mathcal{M}_i = \{\mathbf{m}_{i,j}\}_{j=1}^{k}$, $\forall i \in [1, g]$. We will use the terms "Page $i$" and "Page $\mathcal{M}_i$" interchangeably hereafter. Starting from page $\mathcal{M}_g$, the base station constructs the packets for each page in the reverse order as the pages are transmitted. An example is given in Fig. 1, where $g = 3, k = 3, n = 6$.

*1) Page $g$:* For page $\mathcal{M}_g$, the base station applies $f$ on $\{\mathbf{m}_{g,j}\}_{j=1}^{k}$ to generate $n$ encoded blocks as

$$f(\mathbf{m}_{g,1}, \cdots, \mathbf{m}_{g,k}) = (\mathbf{e}_{g,1}, \cdots, \mathbf{e}_{g,n}) \ . \qquad (1)$$

The $n$ packets of page $\mathcal{M}_g$ are then constructed as $P_{g,j} := \langle \nu, g, j, \mathbf{e}_{g,j} \rangle, \forall j \in [1, n]$, where $\nu$ and $g$ denote the code version and the page number, respectively.

*2) Pages $g-1$ to 1:* After constructing $\{P_{g,j}\}_{j=1}^{n}$, the base station proceeds to construct $\{P_{g-1,j}\}_{j=1}^{n}$ for page $\mathcal{M}_{g-1}$. The basic idea is to append the hash images of $\{P_{g,j}\}_{j=1}^{n}$ to the original $k$ blocks of page $\mathcal{M}_{g-1}$ and then apply $f$ to generate the $n$ packets of $\mathcal{M}_{g-1}$. In particular, the base station computes $h_{g,j} = H(P_{g,j}), \forall j \in [1, n]$, and then splits $h_{g,1}|| \cdots ||h_{g,n}$ into $k$ slices of equal length, denoted by $\{\mathbf{h}_{g,j}\}_{j=1}^{k}$. The $n$ encoded blocks of page $\mathcal{M}_{g-1}$ are then generated as

$$f(\mathbf{m}'_{g-1,j}, \cdots, \mathbf{m}'_{g-1,k}) = (\mathbf{e}_{g-1,1}, \cdots, \mathbf{e}_{g-1,n}) \ , \qquad (2)$$

where $\mathbf{m}'_{g-1,j} := \mathbf{m}_{g-1,j}||\mathbf{h}_{l,j}$. The $n$ packets of page $\mathcal{M}_{g-1}$ to be transmitted are finally generated as $P_{g-1,j} := \langle \nu, g-1, j, \mathbf{e}_{g-1,j} \rangle$, $\forall j \in [1, n]$.

By repeating the above process, the base station can also iteratively construct the packets for pages $g-2$ to 1, i.e., $\{P_{i,j}\}_{j=1}^{n}, \forall i \in [1, g-2]$.

*3) Page 0 and signature packet:* We use a similar approach as in Seluge [7] to authenticate page $\mathcal{M}_1$ by purposefully introducing a hash page $\mathcal{M}_0$ as the concatenation of the $n$ hash images of page $\mathcal{M}_1$, i.e., $\mathcal{M}_0 := h_{1,1}|| \cdots ||h_{1,n}$. In general, $\mathcal{M}_0$ is much shorter than other pages but still cannot fit into one packet. Therefore, the base station first splits $\mathcal{M}_0$ into $k_0$ blocks of equal length, denoted by $\{\mathbf{m}_{0,i}\}_{i=1}^{k_0}$, and then applies the erasure code $f_0$ to generate $n_0$ encoded blocks as follows,

$$f_0(\mathbf{m}_{0,1}, \cdots, \mathbf{m}_{0,k_0}) = (\mathbf{e}_{0,1}, \cdots, \mathbf{e}_{0,n_0}) \ , \qquad (3)$$

where $n_0 = 2^d$ for some integer $d$.

The base station then builds a Merkle hash tree [17] of depth $d$ on top of $\{\mathbf{e}_{0,j}\}_{j=1}^{n_0}$, as illustrated in Fig. 2. In particular, the base station computes $\mathbf{v}_j = H(\mathbf{e}_{0,j})$, $j \in [1, n_0]$, and builds a binary tree by computing each internal node as the hash of its adjacent children nodes. For example, $\mathbf{v}_{3-4} = H(\mathbf{v}_3||\mathbf{v}_4)$ and $\mathbf{v}_{1-4} = H(\mathbf{v}_{1-2}||\mathbf{v}_{3-4})$ in Fig. 2.

Given the Merkle hash tree, the base station constructs one packet for each $\mathbf{e}_{0,j}$, which consists of $\mathbf{e}_{0,j}$ itself and its authentication information, i.e., all the siblings of the nodes in the path from $\mathbf{v}_{0,j}$ to the root of the Merkle hash tree. For example, $P_{0,2} := \langle \nu, 0, 2, \mathbf{e}_{0,2}, \mathbf{v}_1, \mathbf{v}_{3-4}, \mathbf{v}_{5-8} \rangle$ in Fig. 2.

The whole code image is authenticated by the base station signing the root of the Merkle hash tree using its private key. To mitigate the possible DoS attack in which the adversary injects many signature packets to deceive sensor nodes into continuous relatively expensive signature verifications, a weak authenticator like a message specific puzzle in Seluge [7] can be attached to the signature packet as a defense.

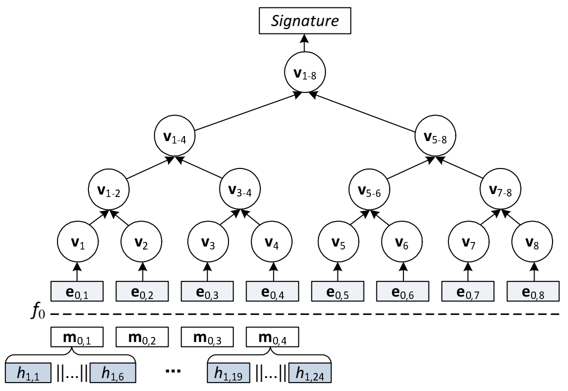


Figure 2. An example of construction Merkle hash tree over page $\mathcal{M}_0$, where $n = 24, k_0 = 4, n_0 = 8$.

## D. Efficient Loss-Resilient Code Dissemination

This subsection details the code dissemination process of LR-Seluge which efficiently copes with packet losses. To ease the illustration, we defer the illustration of LR-Seluge's packet authentication mechanisms to Section IV-E. As in Deluge and Seluge, every node in LR-Seluge works in one of three states at any time: MAINTAIN, RX and TX. LR-Seluge differs from Deluge and Seluge mainly in the TX state due to the use of erasure codes. For self-containment, below we briefly discuss the operations in the MAINTAIN and RX states and then detail the operations in the TX state.

*1) MAINTAIN:* Every node in the MAINTAIN state monitors all its one-hop neighbors to ensure that they all possess the same number of pages of the latest code image. For this purpose, every node periodically broadcasts advertisements, each consisting of the sender ID, the code version number, and the largest page number in possession (which implies all previous pages are also in possession). Here a page is said to be possessed if the sender has received at least $k'$ or $k'_0$ out of $n$ of $n_0$ encoded packets of the page and successfully decoded it.

If a node detects that any neighboring node has either a newer code image or more pages of the same code image, it requests the missing pages from that neighbor with an SNACK request. For example, assume that node $v$ overhears an advertisement from node $u$ indicating that node $u$ has more pages, node $v$ switches to the RX state and begins requesting the missing pages from node $u$ which will switch to the TX state upon an SNACK request from node $v$. In addition, to enable fast code propagation while limiting the number of advertisement packets, every node adjusts the advertisement frequency using Trickle [18], a protocol for maintaining code updates in WSNs.

*2) RX:* A node in the RX state keeps sending SNACK requests to corresponding neighboring nodes which possess a missing page until receiving enough packets to decode the

(a) Node $u$'s tracking table at some time $T$

| Node | $P_{i,1}$ | $P_{i,2}$ | $P_{i,3}$ | $P_{i,4}$ | $\mathbf{d}$ |
|---|---|---|---|---|---|
| $v_1$ | 0 | 1 | 0 | 1 | 1 |
| $v_2$ | 1 | 1 | 0 | 1 | 2 |
| $v_3$ | 1 | 1 | 1 | 1 | 3 |
| Pop. | 2 | 3 | 1 | 3 | |

(b) After transmitting packet $P_{i,2}$

| Node | $P_{i,1}$ | $P_{i,2}$ | $P_{i,3}$ | $P_{i,4}$ | $\mathbf{d}$ |
|---|---|---|---|---|---|
| $v_2$ | 1 | 0 | 0 | 1 | 1 |
| $v_3$ | 1 | 0 | 1 | 1 | 2 |
| Pop. | 2 | 0 | 1 | 2 | |

(c) After transmitting packet $P_{i,4}$

| Node | $P_{i,1}$ | $P_{i,2}$ | $P_{i,3}$ | $P_{i,4}$ | $\mathbf{d}$ |
|---|---|---|---|---|---|
| $v_3$ | 1 | 0 | 1 | 0 | 1 |
| Pop. | 1 | 0 | 1 | 0 | |

Table I
AN EXAMPLE OF NODE OPERATION IN *TX* STATE, WHERE $k = k_0 = 3, n = 4$.

missing page. A SNACK request includes a bit-vector of $n$ bits with every bit indicating whether the corresponding packet is desired. For every received packet, the requesting node first authenticates it using the methods in Section IV-E and stores only the packet passing the authentication. Once $k'$ or $k'_0$ out of $n$ or $n_0$ authenticated packets are received, the requesting node can erasure-decode the missing page and then returns to the MAINTAIN state.

*3) TX:* A node, say $u$, switches to the TX state after receiving an SNACK request for a page it has. The operations of LR-Seluge in the TX state differs from Deluge and Seluge mainly in the following two aspects.

First, to serve an SNACK request, a node need erasure-encode the requested page with the hash images of the next page's packets. Consider node $u$ as an example. Assume that $u$ has received at least $k'$ authenticated packets of page $\mathcal{M}_i$ and erasure-decoded them to recover $\mathcal{M}_i$ and the appended hash images of page $\mathcal{M}_{i+1}$'s encoded packets, i.e., $h_{i+1,1}||\cdots||h_{i+1,n}$. Suppose that $u$ receives an SNACK request from node $v$ requesting $\mathcal{M}_i$. As the base station does in Section IV-C, node $u$ applies the same erasure code $f$ to $\mathcal{M}_i$ appended by $h_{i+1,1}||\cdots||h_{i+1,n}$ and obtains the $n$ encoded packets. Node $u$ can then broadcast the encoded packets corresponding to the bit vector in the SNACK request. Since node $v$ has received page $\mathcal{M}_{i-1}$ and thus the hash images of $\mathcal{M}_i$'s $n$ encoded packets, it can immediately authenticate the packets from $u$ using the method in Section IV-E.

Second, a suitable scheduling algorithm is needed for nodes in the TX state to reduce the number of transmissions for reducing communication overhead. In particular, different packets of the same page may be needed by different neighbors of the node having that page due to random packet losses. It is thus desirable for the sender to transmit the smallest subset of the $n$ encoded packets to simultaneously satisfy the requests from all its neighbors. This scheduling requirement is not found in existing code dissemination schemes. In particular, a node in Deluge and Seluge simply transmits packets corresponding to the union of bit vectors in SNACK packets, and a node in the schemes [2], [5], [6] based on rateless erasure codes always transmits a fresh encoded packet for an SNACK request.

We propose an effective *greedy round-robin* scheduling algorithm for nodes in the TX state. The basic idea is for a node to transmit the packet desired by the highest number of neighbors in a round-robin fashion. More specifically, every node in the TX state, say node $u$, maintains a so-called *tracking table* with every table entry corresponding to one neighbor from which an SNACK was received. The tracking entry for a node, say $v$, consists of the following fields.

- The node ID $v$;
- A bit vector of length $n$ indicating which packets have been received by $v$ from $u$'s current point of view;
- The *distance* of node $v$, denoted by $\mathbf{d}_v$ and referring to the number of additional packets $v$ needs to recover the requested page.

Initially, the tracking table is empty. Upon receiving an SNACK request from a node, say $v$, node $u$ first checks if there is an entry for node $v$. If not, node $u$ creates an entry for node $v$, copies the bit vector from the SNACK request, and sets the distance $\mathbf{d}_v$ as the additional number of packets needed by $v$. For example, if all bits in the bit vector of the SNACK request are ones, then $\mathbf{d}_v = k'$. In general, since node $v$ needs at most $k'$ packets for decoding the requested page, we have $\mathbf{d}_v = q + k' - n$, where $q$ is the number of ones in the bit vector. If there is an entry for node $v$, node $u$ updates the entry according to the SNACK request.

To illustrate the scheduling algorithm, assume that node $u$ is handling the SNACK requests from its neighbors for page $\mathcal{M}_i$ which is erasure-encoded into packets $\{P_{i,j}\}_{j=1}^n$. We also define the *popularity* of a packet as the number of nodes requesting it. Also assume that there are $z$ entries in node $u$'s tracking table. The $z$ bit vectors form a $z \times n$ bitmap, in which the total number of ones in the $j$th column indicates the popularity of packet $P_{i,j}$. The first packet, say $P_{i,x}$, sent by $u$ is the packet with the highest popularity and also the lowest packet index in case that there are multiple packets of the highest popularity. After sending $P_{i,x}$, node $u$ updates the tracking table by setting all the bits in the $x$th column to zero and decreasing the distances of the nodes needing $P_{i,x}$ by one. Note that if $P_{i,x}$ failed to reach some nodes, these nodes may request it again in a later SNACK packet. If some nodes' distances reach zero, their entries are deleted. The next packet is selected as the one with the highest popularity and the index equal to $\min\{x+1, \cdots, n, n+1, \cdots, n+x-1\} \mod n$, i.e., the first packet to the right of $P_{i,x}$ with the highest popularity. This process continues until $u$'s tracking table is empty.

Table I gives an example where $k = k' = 3, n = 4$. Assume that at some point, node $u$'s tracking table has three entries for nodes $v_1$, $v_2$ and $v_3$, as shown in Table (a). The popularity of packets $P_{i,1}$ to $P_{i,4}$ are $2, 3, 1,$ and $3,$

respectively. Node $u$ will first transmit $P_{i,2}$ and then updates the tracking table to Table (b). Note that node $v_1$ has been removed from the tracking table because its distance reaches zero. Subsequently, node $u$ chooses $P_{i,4}$ to transmit because it is the first packet of the highest popularity on $P_{i,2}$'s right side. After $P_{i,4}$ is transmitted, the tracking table is updated again to Table (c). Finally, packet $P_{i,1}$ is transmitted, after which the tracking table becomes empty.

### E. Authenticated Code Dissemination

This subsection details how LR-Seluge realizes authenticated code dissemination. LR-Seluge adopts the page-by-page dissemination approach from Deluge and Seluge in which a node can only request a new page if all previous pages have been completely received and recovered. This page-by-page strategy together with LR-Seluge's packet construction enables immediate packet authentication to ensure code-image integrity and also prevents the DoS attack that targets exhausting the receivers' energy or buffers.

In particular, the base station initiates the dissemination process by broadcasting the signature packet. On receiving the signature packet, every sensor node, say $u$, verifies the signature to authenticate the root of the Merkle hash tree, e.g., $\mathbf{v}_{1-8}$ in Fig. 2. If the verification succeeds, node $u$ begins to send SNACK packets requesting the packets of page $\mathcal{M}_0$. Every packet in page $\mathcal{M}_0$ can be immediately authenticated upon its arrival. For example, for packet $P_{0,1}$ in Fig. 2, node $u$ can verify its authenticity by checking if the following equation holds,

$$\mathbf{v}_{1-8} = H(H(H(H(\mathbf{e}_{0,1})||\mathbf{v}_2)||\mathbf{v}_{3-4})||\mathbf{v}_{5-8}) .$$

If so, it stores the packet and otherwise drops it.

Once node $u$ has collected $k'_0$ authenticated packets of page $\mathcal{M}_0$, say $\{P_{0,j_x}\}_{x=1}^{k'_0}$, it can erasure-decode $\mathcal{M}_0$ as

$$f_0^{-1}(\mathbf{e}_{0,j_1}, \cdots, \mathbf{e}_{0,j_{k'_0}}) = (\mathbf{m}_{0,1}, \cdots, \mathbf{m}_{0,k_0}) . \quad (4)$$

Recall that $\mathcal{M}_0$ contains all the hash images of the packets of page $\mathcal{M}_1$. Therefore, node $u$ can subsequently authenticate all the packets of page $\mathcal{M}_1$ upon their arrivals by a simple hash verification. Similarly, once at least $k'$ authenticated packets of $\mathcal{M}_1$ have been collected, node $u$ can decode $\mathcal{M}_1$ to get all the hash images of the packets of page $\mathcal{M}_2$ whereby to immediately authenticate all the packets of page $\mathcal{M}_3$. In short, the page-by-page strategy guarantees that whenever node $u$ requests a new page from a neighboring node, all the information needed for authenticating the new page is available at that time. Therefore, any data packet can be immediately authenticated upon their arrivals.

In addition, LR-Seluge adopts the same mechanisms in Seluge, i.e., cluster key and message specific puzzle, to authenticate advertisement and SNACK packets and to effectively filter out forged signatures of the root of the Merkle hash tree, respectively. Therefore, LR-Seluge inherits the same level of resilience to DoS attacks that exploit Deluge's epidemic propagation and suppression mechanisms.

It is worth noticing that LR-Seluge and all existing secure code dissemination schemes [7]–[12] are vulnerable to a special kind of *denial of receipt* attack in which a compromised sensor node denies it has received any data packets but keeps sending SNACK packets to a victim node in order to deplete its energy. In particular, a victim node need transmit $k'$ data packets on receiving a SNACK packet with all bits set to one. It is fundamentally difficult to verify whether a particular packet has been received by certain nodes in lossy environment.

To mitigate the impact of this attack, a possible solution is to replace cluster key by a local authentication scheme like LEAP [19] to simultaneously authenticate and identify the source of any SNACK packet. In addition, each node in TX state maintains a counter for the number of SNACK packets from each neighbor. For any page, if the number of data packets requested by a neighboring node, say $v$, exceeds some certain threshold, the node that serving the page, say $u$, can simply ignore future SNACK packets from $v$, under the assumption that either $v$ is launching the denial of receipt attack or the channel between $u$ and $v$ is too bad so that $v$ should request data packets from its other neighbors. Due to the space limitation, we leave the detailed investigation as our future work.

## V. Performance Analysis

Section IV-E discusses how LR-Seluge realizes code-image integrity and DoS resilience. In this section, we analyze the communication and computation overhead of LR-Seluge.

### A. Communication Overhead

The communication costs of Seluge and LR-Seluge both comprise the transmissions of data, advertisement, and SNACK packets, among which data-packet transmissions account for the most. To enable theoretical tractability, we here analyze the number of data-packet transmissions under Seluge and LR-Seluge, respectively, under a one-hop scenario. This is a meaningful simplification, as the performance of Seluge and LR-Seluge largely depends on hop-by-hop local broadcasting. The impact of advertisement and SNACK packets and also the communication costs of Seluge and LR-Seluge in multi-hop scenarios will be demonstrated using simulations in Section VI.

We consider a one-hop scenario consisting of $N$ receivers and a local sender at the center. Our main goal is to demonstrate the impact of employing erasure codes in lossy environments. So we adopt a similar model as in [20] in which every packet to node $i$ gets lost with probability $p_i$ and packet losses at different nodes are uncorrelated.

We then have the following theorems regarding the number of data packet transmissions in Seluge and LR-Seluge, respectively, whose proofs are available in our technical report [21]. It is worth mentioning that Theorem 2 is obtained by analyzing a variation of LR-Seluge, called ACK-based LR-Seluge. In ACK-based LR-Seluge, instead of using SNACK packets, each receiver returns an ACK packet only after receiving $k'$ packets of the current page. Since the local sender is not aware of which packets are missing at each node, it repeatedly transmits the $n$ erasure-coded packets until receiving an ACK from every neighboring node. ACK-based LR-Seluge is apparently less efficient than LR-Seluge, as the sender may unnecessarily transmit some packets which have reached all the receivers. Therefore, its communication cost is an upper bound of LR-Seluge.

**THEOREM** *1: With Seluge, the expected number of data-packet transmissions needed to transmit a page of $k$ packets to all $N$ nodes is given by*

$$k\sum_{t=1}^{\infty} t \cdot \big(\prod_{i=1}^{N}(1-p_i^t) - \prod_{i=1}^{N}(1-p_i^{t-1})\big) . \quad (5)$$

**THEOREM** *2: With LR-Seluge employing a $k$-$n$-$k'$ erasure code, the expected number of data-packet transmissions needed for all $N$ nodes to receive at least $k'$ packets to recover the original page of $k$ packets is bounded by*

$$n\sum_{r=1}^{\infty} r \cdot (\mathsf{Pr}(\mathbf{R} \le r) - \mathsf{Pr}(\mathbf{R} \le r-1)) , \quad (6)$$

*where* $\mathsf{Pr}(\mathbf{R} \le r) = \prod_{i=1}^{N}\sum_{j=k'}^{n}\binom{n}{j}(1-p_i^r)^j p_i^{r(n-j)}$.

### B. Computation Overhead

The computation overhead of LR-Seluge comes from packet authentication and page encoding/decoding.

The computation cost incurred by packet authentication is very similar to that of Seluge. First, authenticating a signature packet requires one hash function for the weak authenticator and one signature verification. Second, authenticating a packet of page $\mathcal{M}_0$ requires $d+1$ hash computations, and total $k'_0(d+1)$ hash computations are needed for page $\mathcal{M}_0$. In contrast, authenticating a packet of page $\mathcal{M}_i, 1 \le i \le g$ requires one hash computation, so total $k'$ hash computations are needed for each page $\mathcal{M}_i$.

The computation cost incurred by erasure decoding and encoding depends on the particular erasure code used by LR-Seluge. Theoretically speaking, LR-Seluge can be integrated with any erasure code as long as the parameters $(k, n, k')$ are satisfied. In our evaluation, we adopt a fixed-rate LT code from SYNAPSE [6], in which each encoded packet is the XOR of some original packets. Assuming that $k = 32$, $n = 64$, and $k' = 35$, decoding a page of 32 original packets of 25 bytes requires 5142 XOR operations on average and takes about 462 ms on a Tmote Sky sensor node [6]. Same as Seluge [7], LR-Seluge uses data packets with an effective payload length of 96 bytes. Since the decoding cost of the LT code is linear to the packet length, decoding a page of 32 original packets of 96 bytes requires approximately 19745 XOR operations and takes about 1.8 seconds. Finally, nodes in the TX state need erasure-encode the original page to obtain the other $n-k'$ missing packets, which requires $s(n-k')*96$ XOR operations on average, where $s$ is the average number of original blocks XORed to generate an encoded packet. If $s = 12.06$ as in [6], the encoding cost for a page in LR-Seluge requires approximately 33576 XOR operations and takes about 3.1 seconds on a Tmote Sky sensor node. It is also worth noting that in LR-Seluge, only a few nodes in the TX state need perform encoding.

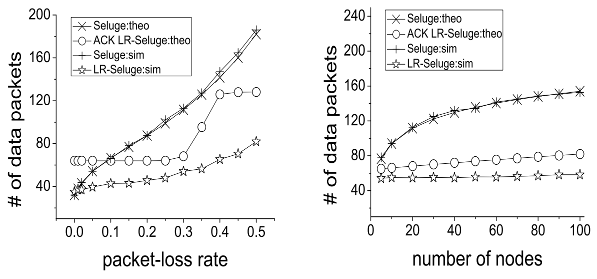
## VI. PERFORMANCE EVALUATION

In this section, we compare LR-Seluge with Seluge using extensive simulations in TOSSIM [22], a discrete event simulator distributed with TinyOS 2.1.1.

Unless stated otherwise, the following simulation configurations are used. For the $k$-$n$-$k'$ erasure code $f(\cdot)$, we have $k = 32$, $n = 64$, and $k' = 35$; for the $k_0$-$n_0$-$k'_0$ erasure code $f_0(\cdot)$, we have $k_0 = 8$, $n_0 = 16$, and $k'_0 = 11$. As in Seluge [7], we set the packet-payload size to 102 bytes and use the 64-bit truncation of SHA-1 as the hash function $H(\cdot)$, and use a gap of 17 ms between two data-packet transmissions. The image version number, page number, and packet number in both Seluge and LR-Seluge packets totally consume 6 bytes, which leaves 96 bytes for the effective packet payload. In addition, except the packets of Page 0, each packet in Seluge and LR-Seluge contains one and $n/k$ hash values, respectively. Therefore, the packets of Seluge and LR-Seluge have $96 - 8 = 88$ bytes and $96 - 8 * n/k = 80$ bytes, respectively, for code-image slices. In our simulations, each page consists of 32 packets for both Seluge and LR-Seluge, and the code image $\mathcal{M}$ is of 20 KB. Under Seluge, $\mathcal{M}$ leads to totally $g = 8$ pages, among which the last page comprises only 9 packets; under LR-Seluge, $\mathcal{M}$ leads to totally $g = 8$ pages as well, all of which comprise 32 packets.

In addition, we set the minimum delays between two advertisement packets and between two SNACK packets to 1 second and 128 ms, respectively, for both Seluge and LR-Seluge. Moreover, we use the delays of 2.5 and 3.5 seconds to emulate the decoding and encoding of a page, respectively, which is clearly in favor of Seluge because the decoding and encoding times are estimated as 1.8 and 3.1 seconds, respectively, in Section V-B.

The performance metrics used in our comparisons include *the total number of data packets*, *the total number of SNACK packets*, *the total number of advertisement packets*, and *the overall dissemination latency* which is defined as the time required to finish disseminating a code image to all the nodes in the network. Since SNACK packets in LR-Seluge are $n - k$ bits longer than those in Seluge, we will also show *the total communication cost* covering data, SNACK and advertisement packets in bytes for fairness.



(a) 20 receivers (b) $p = 0.3$

Figure 3. One-hop one-page scenario.

Each measurement in the following figures is the average over 20 simulation runs, each with a different seed.

## A. Validation of Analytical Results

To validate the analytical results in Section V-A, we first simulate the transmission of one page in Seluge and LR-Seluge in a fully-connected one-hop scenario with one local sender and a varying number of local receivers. To fully control and illustrate the impact of packet losses, we use a similar simulation strategy as in [6], where nodes are placed close enough to eliminate packet transmission errors caused by channel impairments, and packet losses are emulated by each node dropping received data, advertisement, or SNACK packets with the same probability $p$ at the application layer.

Figs. 3(a) and 3(b) show the analytical results of Seluge and ACK-based LR-Seluge and the simulation results of Seluge and LR-Seluge. We can see that the simulation result of Seluge closely matches the analytical result, and the number of data packets transmitted in ACK-based LR-Seluge is always larger than that of LR-Seluge obtained from simulations, which confirms that the number of data-packet transmissions in LR-Seluge is upper bounded by that in ACK-based LR-Seluge. In addition, we can see a significant increase in the number of data packet transmissions in ACK-based LR-Seluge when the packet loss rate increases from 0.3 to 0.4. The reason is that when $p \leq 0.3$ (or $p \geq 0.4$), ACK-based LR-Seluge can finish transmitting one page in one round (or two rounds) with high probability. The figures also confirm that LR-Seluge incurs much fewer data-packet transmissions than Seluge in lossy environments and is less sensitive to the number of receivers as well.

## B. The One-Hop Case

For lack of space, hereafter we focus on comparing LR-Seluge and Seluge with simulations which involve disseminating a code image $\mathcal{M}$ of 20 KB.

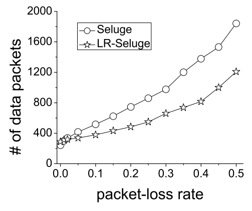
*1) Impact of the packet-loss rate:* Figs. 4(a)∼4(e) show the impact of the packet-loss rate $p$ on LR-Seluge and Seluge, where there are $N = 20$ local receivers. It is not surprising to see that the total communication costs and dissemination latencies of LR-Seluge and Seluge both increase as $p$ increases. In addition, when $p \leq 0.01$, LR-Seluge has a slightly larger communication cost than Seluge for both data and control packets. There are two reasons. First, LR-Seluge has more data packets than Seluge for the same code image due to the use of erasure codes. Second, under LR-Seluge, each node needs $k' > k$ packets to recover each page, so more data-packet transmissions are needed to disseminate one page if there are no or rare packet losses. In contrast, when $p > 0.01$, LR-Seluge outperforms Seluge in all the five performance metrics. For example, when $p = 0.4$, LR-Seluge reduces the total communication cost by $44\%$ and the dissemination latency by $48\%$ in comparison with Seluge. These results clearly demonstrate that LR-Seluge is much more resilient to packet losses than Seluge.

*2) Impact of the node density:* Figs. 5(a)∼5(e) show the impact of the number $N$ of local receivers on LR-Seluge and Seluge, where the packet-loss rate $p = 0.1$. We can see that the communication costs of LR-Seluge and Seluge all increase as $N$ increases. This is understandable because it always requires more data and control packet transmissions to disseminate the same code image under packet losses. However, LR-Seluge is much less sensitive to the increase of $N$, which can be clearly seen in Figs. 5(a)∼5(d). In addition, the dissemination latency of Seluge increases slightly as $N$ increases, while that in LR-Seluge slightly decreases. This could be explained as follows. In Seluge, as $N$ increases, the numbers of SNACK and data packet transmissions increase significantly, which leads to higher the dissemination latency. In contrast, the numbers of SNACK and data packet transmissions increase much slower in LR-Seluge as $N$ increases. In addition, the more nodes that demand the current page, the earlier the first node receives $k'$ packets and thus recovers the current page, and the earlier the SNACK packet is transmitted to request the next page. This leads to the decrease in total dissemination latency.

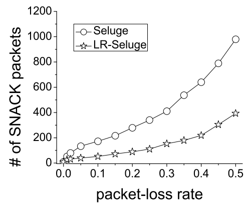
*3) Impact of the erasure-coding rate:* Figs. 6(a)∼6(e) show the impact of the erasure-coding rate $n/k$ on LR-Seluge under different packet-loss rates, where $k$ is fixed to 32. We can see that by introducing a limited number of redundant data packets, the communication cost of LR-Seluge decreases significantly. For example, when $p = 0.1$ and $n = 56$, the total number of SNACK and data packet transmissions decrease by 70.5% and 30%, respectively. As $n/k$ further increases, the communication cost and dissemination latency increase slowly. The reason is that higher erasure-coding rates lead to shorter packet space for code-image slices and thus more packets for the same code image.

## C. The Multi-Hop Case

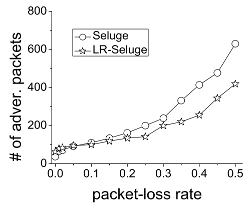
We also simulate LR-Seluge and Seluge in multi-hop networks. In particular, we simulate them under two $15 \times 15$ grid sensor networks using the exemplary topologies specified in *15-15-tight-mica2-grid.txt* (high node density) and *15-15-medium-mica2-grid.txt* (low node density) and RF noise and



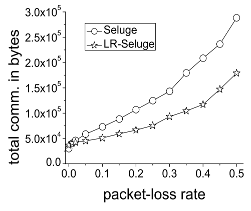
(a) data packet



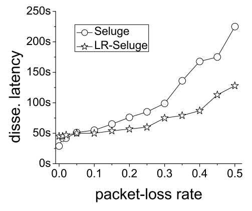
(b) SNACK packet



(c) advertisement packet

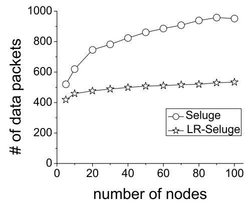


(d) total communication cost

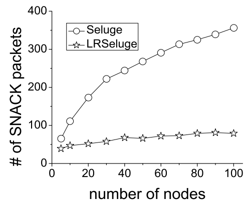


(e) dissemination latency

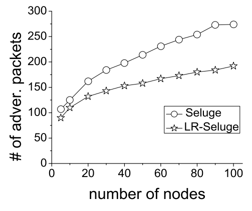
Figure 4. Impact of the packet-loss rate, where there are $N = 20$ local receivers.



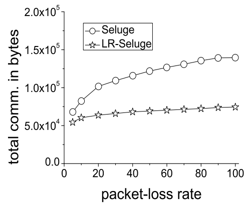
(a) data packet



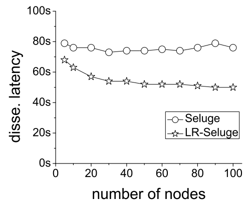
(b) SNACK packet



(c) advertisement packet

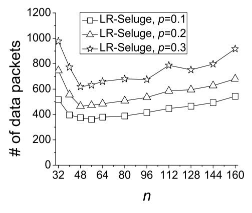


(d) total communication cost

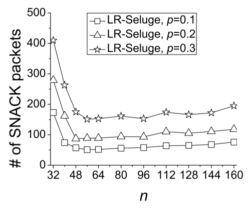


(e) dissemination latency

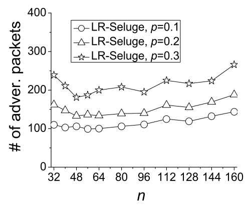
Figure 5. Impact of the node density, where the packet-loss rate $p = 0.1$.



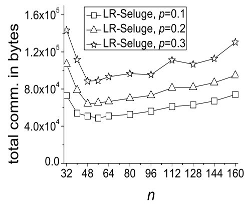
(a) data packet



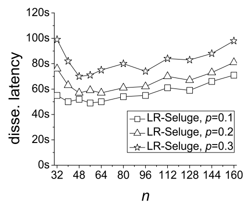
(b) SNACK packet



(c) advertisement packet



(d) total communication cost



(e) dissemination latency

Figure 6. Impact of the erasure-coding rate.

Table II
PERFORMANCE COMPARISON UNDER NETWORK WITH HIGH DENSITY

| | **LR-Seluge** | **Seluge** | Ratio |
|---|---|---|---|
| Total # of SNACK packets | 1804 | 3629 | 49.71% |
| Total # of data packets | 4040 | 5496 | 73.50% |
| Total # of adver. packets | 1059 | 1678 | 63.11% |
| Total comm. cost in bytes | $6.05 \times 10^5$ | $8.78 \times 10^5$ | 68.91% |
| Dissemination latency (s) | 93 | 146 | 63.70% |

Table III
PERFORMANCE COMPARISON UNDER NETWORK WITH MEDIUM DENSITY

| | **LR-Seluge** | **Seluge** | Ratio |
|---|---|---|---|
| Total # of SNACK packets | 10927 | 20287 | 53.86% |
| Total # of data packets | 38197 | 47373 | 80.63% |
| Total # of adver. packets | 13088 | 18812 | 69.57% |
| Total comm. cost in bytes | $5.55 \times 10^6$ | $7.27 \times 10^6$ | 76.34% |
| Dissemination latency (s) | 1154 | 1534 | 75.23% |

interference from the sample noise trace file *meyer-heavy.txt* of the TinyOS distribution.

Tables II and III compare the performance of LR-Seluge and Seluge under these two topologies, respectively. We can see that LR-Seluge outperforms Seluge for all the performance metrics by significant margins, which coincides with the results under the one-hop scenario. We have also simulated other network topologies generated by the topology tool provided by the TinyOS distribution which is based on theoretical propagation models. In general, the results are very similar to those shown in Tables II and III and thus are omitted here due to the space limitation. In addition, we have simulated the impact of different image sizes in both one-hop and multihop networks and observed similar advantages of LR-Seluge over Seluge.

## VII. RELATED WORK

In addition to Deluge [1] and Seluge [7], the following work is most related to our LR-Seluge scheme.

Sluice [8] aims at authenticated code dissemination based on signature and cryptographic hash functions. It creates a chaining relationship among adjacent pages by embedding the hash image of each page into the previous page and signing only the first page. A scheme similar to Sluice is presented in [9], in which the hash image of each packet is included in the previous packet. Both schemes, however, are vulnerable to DoS attacks in which the adversary keeps sending bogus packets that cannot be immediately authenticated, as pointed out in [7]. A scheme with better DoS resilience is presented in [23] and uses Merkle hash trees to enable immediate authentication of packets upon their arrivals. In addition, Tan *at al.* propose a secure code dissemination scheme based on multiple hash chains [10] and also a code dissemination scheme which preserves the confidentiality of the code image [12]. Most recently, Ugus *et al.* [11] present a ROM-friendly secure code dissemination protocol which significantly reduces the memory requirement. All these previous schemes rely on Deluge and thus do not work well in lossy environments

There are also some loss-resilient code dissemination schemes such as AdapCode [5], Rateless Deluge and ACK-less Deluge [2], and SYNAPSE [6], which do not take security into consideration.

## VIII. Conclusion

This paper presents the design and evaluation of LR-Seluge, the first loss-resilient and secure code dissemination scheme for sensor networks. The performance of LR-Seluge is confirmed by both theoretical analysis and thorough simulation results. Our future work is to evaluate LR-Seluge on a real testbed.

## Acknowledgement

This work was supported in part by the US National Science Foundation under grants CNS-0716302 and CNS-0844972 (CAREER). We would like to thank Dr. Yee Wei Law for helpful discussion and sharing the simulation code.

## References

[1] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *SenSys'04*, Baltimore, MD, Nov. 2004, pp. 81–94.

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

[3] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," UCLA, Los Angeles, CA, Tech. Rep., 2003.

[4] A. Wood and J. Stankovic, "Online coding for reliable data transfer in lossy wireless sensor networks," in *DCOSS'09*, Marina del Rey, CA, June 2009, pp. 159–172.

[5] I.-H. Hou, Y.-E. Tsai, T. Abdelzaher, and I. Gupta, "Adapcode: Adaptive network coding for code updates in wireless sensor networks," in *INFOCOM'08*, Phoenix, AZ, Apr. 2008, pp. 1517–1525.

[6] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. H. III, and M. Zorzi, "SYNAPSE: A network reprogramming protocol for wireless sensor networks using fountain codes," in *IEEE SECON'08*, San Francisco, CA, June 2008, pp. 188–196.

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

[8] P. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure dissemination of code updates in sensor networks," in *ICDCS'06*, Washington, DC, July 2006, pp. 53–62.

[9] P. Dutta, J. Hui, D. Chu, and D. Culler, "Securing the deluge network programming system," in *IPSN'06*, Apr. 2006, pp. 326–333.

[10] H. Tan, S. Jha, D. Ostry, J. Zic, and V. Sivaraman, "Secure multi-hop network programming with multiple one-way key chains," in *WiSec'08*, Mar. 2008, pp. 183–193.

[11] O. Ugus, D. Westhoff, and J.-M. Bohli, "A ROM-friendly secure code update mechanism for WSNs using a stateful-verifier $\tau$-time signature scheme," in *WiSec'09*, Zurich, Switzerland, Mar. 2009, pp. 29–40.

[12] H. Tan, D. Ostry, J. Zic, and S. Jha, "A confidential and DoS-resistant multi-hop code dissemination protocol for wireless sensor networks," in *WiSec'09*, Zurich, Switzerland, Mar. 2009, pp. 245–252.

[13] "TinyOS: An open-source operating system designed for wireless embedded sensor networks," http://www.tinyos.net/.

[14] P. Ning, A. Liu, and W. Du, "Mitigating dos attacks against broadcast authentication in wireless sensor networks," *ACM TOSN*, vol. 4, no. 1, pp. 1–31, Jan. 2008.

[15] M. Mitzenmacher, "Digital fountains: a survey and look forward," in *IEEE ITW'04*, San Antonio, TX, Oct. 2004.

[16] H. Wang, B. Sheng, C. Tan, and Q. Li, "WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes," College of William and Mary, Computer Science, Williamsburg, VA, Tech. Rep. WM-CS-2007-11, 2007.

[17] R. Merkle, "Protocols for public key cryptosystems," in *IEEE S&P'80*, Oakland, CA,USA, Apr. 1980, pp. 122–134.

[18] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *NSDI'04*, San Francisco, CA, Mar. 2004, pp. 15–28.

[19] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *ACM CCS'03*, Washington, DC, Oct. 2003, pp. 62–72.

[20] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Transactions on Vehicular Technology,*, vol. 58, no. 2, pp. 914–925, Feb. 2009.

[21] R. Zhang and Y. Zhang, "LR-Seluge: Loss-resilient and secure code dissemination in wireless sensor networks," Arizona State University, Tech. Rep., 2011, available at http://wins.lab.asu.edu/files/LR-Seluge-tech.pdf.

[22] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire tinyos applications," in *SenSys'03*, Los Angeles, CA, Nov. 2003, pp. 126–137.

[23] J. Deng, R. Han, and S. Mishra, "Secure code distribution in dynamically programmable wireless sensor networks," in *IPSN'06*, Nashville, Tennessee, Apr. 2006, pp. 292–300.