Data Authenticity and Availability in Multihop Wireless Sensor Networks

ERMAN AYDAY, Georgia Institute of Technology
FARSHID DELGOSHA, New York Institute of Technology
FARAMARZ FEKRI, Georgia Institute of Technology

Security services such as data confidentiality, authenticity, and availability are critical in wireless sensor networks (WSNs) deployed in adversarial environments. Due to the resource constrain's of sensor nodes, the existing protocols currently in use in adhoc networks cannot be employed in WSNs. In this article, we propose a protocol called location-aware network-coding security (LNCS) that provides all the aforementioned security services. By dividing the terrain into nonoverlapping cells, the nodes take advantage of the location information to derive different location-binding keys. The key idea in LNCS is that all the nodes involved in the protocol collaborate in every phase. We employ random network coding in order to provide data availability significantly higher than that in other schemes. A hash tree-based authentication mechanism is utilized to filter the bogus packets enroute. We provide a comparison between our scheme and previously proposed schemes. The results reveal significant improvement in data availability while maintaining the same level of data confidentiality and authenticity.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms: Security, Reliability

Additional Key Words and Phrases: Data authentication, data availability, wireless sensor networks, pollution attack, network coding

ACM Reference Format:
Ayday, E., Delgosha, F., and Fekri, F. 2012. Data authenticity and availability in multihop wireless sensor networks. ACM Trans. Sensor Netw. 8, 2, Article 10 (March 2012), 26 pages.
DOI = 10.1145/2140522.2140523 http://doi.acm.org/10.1145/2140522.2140523

1. INTRODUCTION

Wireless sensor networks (WSNs) have gained so much attention among the research community because of their numerous applications, such as managing energy plants, logistics and inventory, battlefields, and medical monitoring [Arampatzis et al. 2005]. A sensor is a low-cost, battery-powered device with limited computational power and memory that is equipped with sensing and radio transmission units. A WSN is an unattended network without any infrastructure that consists of sensors that communicate wirelessly. WSNs are usually connected to the outside world through

This material is based on work supported by the Army Research Office (ARO) under grant 49586CI.
F. Delgosha was a graduate student at Georgia Institute of Technology at the time of this work.
Authors' addresses: E. Ayday and F. Fekri, Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA; emails: {erman,faramarz.fekri}@ece.gatech.edu; F. Delgosha, Department of Electrical and Computer Engineering and Computer Science, New York Institute of Technology, Old Westbury, NY; email: fdelgosh@nyit.edu

DOI 10.1145/2140522.2140523 http://doi.acm.org/10.1145/2140522.2140523

a computationally powerful center called the sink that is also responsible for data collection and data fusion.

Triggered by an event in the field or upon a sink query, the nodes close to the center of stimulus collaboratively generate a report and send it back to the sink. Considering the wide scattering of the nodes in the field, the center of stimulus is usually distanced from the sink, rendering single-hop communication with the sink impossible. Therefore, the generated report is forwarded to the sink through multihops.

Security of multihop data transfer in WSNs becomes very important, especially for the networks deployed in hostile environments. Constraints of sensor nodes and the lack of infrastructure in such networks poses new challenges in designing security services. In an adversarial environment, the major attacks on a WSN are as follows.

—*Eavesdropping*. By listening to the radio channel, the adversary tries to obtain meaningful information.
—*False Data Injection*. An insider node attempts to cause false alarms or to consume the energy of the forwarding sensors by injecting false data.
—*Data Drop*. An insider node drops a legitimate report on the forwarding path toward the sink.
—*Noise Injection*. The legitimate reports are modified by injecting noise. Thus, the sink is unable to regenerate the original message.

In this article, we reintroduce the location-aware network-coding security (LNCS) of Ayday et al. [2007] with mathematical details. The proposed scheme provides all the previously mentioned security services with moderate communication and computation overhead. LNCS makes extensive use of node collaboration and data redundancy to provide data authenticity and availability. To achieve this goal, we assume that the node scattering is dense enough, such that a single event in the field is sensed by more than one sensor node, and a message broadcast is received by multiple nodes in the proximity. Every step of the proposed scheme is carried out by multiple nodes involved in the protocol, and all of them generate the same output. Hence, a few malicious nodes can be detected, and the bogus packets generated by them are dropped.

To evenly distribute the load of report generation and forwarding and also enhance node collaboration, we partition the terrain into non-overlapping cells of the same shape and area. A report generated at the event cell is forwarded towards the sink on the shortest path in a cell-by-cell fashion. The advantages of this technique are localizing adversarial activities and providing a robust and simple routing and authentication mechanism. To provide an authentication mechanism, all the nodes involved in the protocol in every cell generate a hash tree on the same packets. Every node broadcasts only a few packets, along with the corresponding authentication information. The nodes in the next forwarding cell check the authenticity of all the received packets and drop bogus ones.

Linear network coding is an essential component of LNCS [Li et al. 2003]. In this type of coding, intermediate nodes process the data by generating random linear combinations of the packets they receive. The objectives of using this novel idea of random network coding in LNCS are twofold. First, every node generates correlated data by calculating random linear combinations of the received packets. Hence, the availability of the data at the receiver is guaranteed with a high probability. The second advantage is the feasibility of implementing LNCS in the real case scenario in which the communication media between the sensors is usually modeled as the erasure channel, since the redundancy in the data allows the sink to recover the original message packets by receiving few encoded packets. The erasure channel also models the packet-drop attack by an adversary. Therefore, linear network coding intrinsically provides a counter measure against the data-drop attack. However, this idealistic feature comes with the

cost of the bogus-packet propagation (due to the pollution attack), since only one bogus packet in a linear combination infects the generates packets. We will analyze this problem in detail.

The main contributions of our scheme are summarized in the following.

(1) In the proposed scheme, data is forwarded toward the sink using multiple paths and authenticated by multiple nodes, using a collaboration between these nodes. Data authentication is performed without overhearing nodes and voting systems. Such mechanisms, employed by some other schemes, suffer from extensive communication overhead. Moreover, in the proposed scheme, the bogus packets are identified and dropped by the legitimate nodes in the next cell.
(2) We employ linear network coding in our scheme to generate redundant information that facilitates recovery of the packets erased by the channel or dropped by malicious nodes. This kind of coding significantly improves data availability, compared to all other schemes.
(3) Contrary to previous schemes, our proposed scheme does not require a trustworthy cluster head (CH) that is responsible for generating the report and forwarding it to the next cell. We emphasize that the existence of a trustworthy CH cannot be guaranteed, and that a malicious CH completely breaks down the security of the protocol.

The rest of this article is organized as follows. In the rest of this section, we summarize the related work and present the notation used throughout. In Section 2, we briefly review the cryptographic primitives employed in our scheme. In Section 3, we discuss and motivate the use of linear network coding to provide security and high data availability and give a detailed analysis of the major problems due to the use of linear network coding. In Section 4, we introduce LNCS that uses linear network coding and resolves these problems by novel authentication and routing techniques tailored for linear network coding. In Section 5, we analyze the security of LNCS. The communication and computation overheads of LNCS are studied in Section 6. In Section 7, we compare the security strengths and efficiency of LNCS with another scheme. The concluding remarks are provided in Section 8.

1.1. Related Work

Interleaved hop-by-hop authentication (IHA) is one of the first works in data authentication for WSNs [Zhu et al. 2004]. In IHA, the authenticity of the report is verified at every hop of the forwarding path to the sink, using message authentication codes (MACs). For this purpose, both authentication chains are discovered and authentication keys are established at the initialization phase of the network operation [Zhang 2005]. A report with even one unverified MAC is regarded as bogus and dropped enroute. Therefore, a malicious node injecting noise to the network always causes the messages to be dropped. The other drawback of IHA is the association maintenance that introduces high communication overhead. Another approach to data authentication is the statistical enroute filtering (SEF) proposed in Ye et al. [2005]. In SEF, every node is predistributed with the keying material that is used to establish the authentication keys after the network deployment. The key predistribution parameters are selected to guarantee, with high probability, that any CH is able to establish many authentication keys. Because of the probabilistic nature of SEF, every node is required to store many keys in order to guarantee the existence of a minimum number of authentication keys. Therefore, two other drawbacks of SEF are the requirement for large storage memory and the possibility of revealing many authentication keys by compromising only a few nodes. Both previous schemes have a threshold property, that is, an adversary has to compromise a minimum number of authentication keys to forge a report. To achieve

graceful performance degradation to an increasing number of compromised keys, the location-binding keys and location-based key assignment are employed in Yang et al. [2005]. The proposed scheme, called location-based resilient security (LBRS), localizes the adversarial activities to only the area of the network which is under attack. LBRS inherits the disadvantages of the SEF, except the performance degradation behavior. One of the most recent authentication schemes is the location-aware end-to-end data security (LEDS) [Ren et al. 2006]. This is a location-aware scheme that provides many security services, such as data confidentiality, availability, and authenticity. In LEDS, data confidentiality is achieved by using symmetric cryptography and linear secret sharing. To check the authenticity of the data, a legitimate report carries many MACs that are verified by the nodes in the intermediate cells. For data availability, the overhearing nodes in every forwarding cell collaborate to inform the next cell, in case a legitimate report is dropped by a malicious node. Although overhearing nodes theoretically provide data availability, there doesn't seem to exist a practical method for implementing this technique. The most logical implementation is a voting system that has a high communication overhead, whose management introduces a high computational complexity.

As we have discussed, liner network coding comes with the cost of the bogus-packet propagation problem caused by pollution attack. Solutions to the packet pollution attack can be classified into two categories: (1) cryptographic approaches [Charles et al. 2007; Gkantsidis and Rodriguez 2006; Krohn et al. 2004; Yu et al. 2008; Zhao et al. 2007] and (2) information theoretical approaches [Jaggi et al. 2007; Ho et al. 2004]. Cryptographic approaches are based on either homomorphic hash functions [Gkantsidis and Rodriguez 2006; Krohn et al. 2004] or homomorphic signatures [Charles et al. 2007; Yu et al. 2008; Zhao et al. 2007; Boneh et al. 2009]. The use of homomorphic hash functions (or signatures) is a common method for checking the integrity of the packets encoded using network coding. However, it is computationally expensive to use such techniques, especially for resource limited devices, such as sensor nodes. Information theoretical approaches either rely on coding redundant information into packets or strong assumptions on the network or the adversary. In Ho et al. [2004], receivers detect the presence of polluted packets by coding redundant information into packets. However, the scheme provides only a partial solution, as it does not specify any mechanisms for recovering from pollution attacks. Therefore, it does not provide data availability. In Jaggi et al. [2007], a mechanism is developed on strong assumptions, such as limiting the capabilities of the attackers or assuming a secret channel between the source and the sink. Hence, using information theoretic approaches does not provide data availability in the presence of attackers. In our previous work [Delgosha et al. 2006], we proposed a technique for encountering the pollution attack problem by adding a MAC to each message packet at the source and checking the integrity of each message packet at some special cells (called the *checkpoints*) on the way to the sink. We analyze the feasibility of this approach in Section 3.1.

Recently, two techniques are proposed for combating the pollution attack by using efficient and practical solutions [Dong et al. 2009; Yu et al. 2009]. Dong et al. [2009] propose a scheme (referred to as DART) that uses time-based authentication, along with random linear transformations to defend against pollution attacks. The security of the DART relies on time asymmetry, which may introduce additional delays to the scheme. Further, malicious nodes may flood the network to prevent the legitimate nodes from receiving checksums, and hence, to force them to drop legitimate packets. Yu et al. [2009] propose a scheme for securing XOR network coding against pollution attacks that claims to work for normal network coding, as well. The proposed scheme exploits probabilistic key predistribution, as well as MACs. It can filter polluted messages in a few hops with high probability. However, the authors assume restricted capabilities

for malicious nodes. Moreover, the proposed scheme only detects the polluted packets; it does not provide data availability at the sink.

In Ayday et al. [2007], we proposed LNCS that simultaneously provides data integrity, entity authentication, and data availability. This scheme utilizes node collaboration to provide data and entity authentication. For data availability, the LNCS employs random network coding, in which intermediate nodes on the forwarding path linearly combine the incoming packets using random coefficients. Further, we show that the LNCS is robust against pollution attacks. In this article, we explain and analyze the LNCS in complete detail.

1.2. Notations

The set of positive integers is represented by $\mathbb{N}$. For all $n \in \mathbb{N}$, we define $[n] := \{x \in \mathbb{N},\ x \leq n\}$. A Galois field of characteristic q is denoted by $\mathbb{F}_q$. For any $n, k \in \mathbb{N}$, the set of all $n \times k$ matrices with entries from $\mathbb{F}_q$ is denoted by $\mathsf{M}_{n,k}(\mathbb{F}_q)$. For the case $n = k$, we use the short notation $\mathsf{M}_n(\mathbb{F}_q)$. For any $r \in \mathbb{N}$, the notation $\mathsf{M}^r_{n,k}(\mathbb{F}_q)$ is used for the set of all sparse matrices $\mathbf{C} \in \mathsf{M}_{n,k}(\mathbb{F}_q)$, such that (1) every row of $\mathbf{C}$ has at most r nonzero entries such that $\lim_{k\to\infty} r/k = 0$, and (2) none of the columns of $\mathbf{C}$ is entirely zero. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\dagger$. For any matrix $\mathbf{A} \in \mathsf{M}_{n,k}(\mathbb{F}_q)$ and any set of indices $I \subseteq [n]$, the symbol $\mathbf{A}(I)$ represents a submatrix of $\mathbf{A}$ generated by removing any row of $\mathbf{A}$ with index outside I. The notation $x \| y$ implies the concatenation of x and y as bit strings.

To facilitate future references, frequently used notations are listed below with their meanings.

—n. Total number of nodes in the network
—N. Average number of nodes in every cell
—T_0. Number of involved nodes in the event cell
—T. Number of involved nodes in the intermediate cells
—T'. $(= T_0 + \tau)$ Total number of packets generated after network coding
—$\hat{T}$. Number of legitimate packets after report authentication
—t. Minimum number of shares required to reconstruct the message
—Δ_i. The ith cell on the forwarding path
—$\mathcal{V}_i$. Set of the involved nodes in the cell Δ_i
—$\mathbf{e}_i$. Packet vector generated at the cell Δ_i
—$\mathbf{C}_i$. Coefficients matrix generated at the cell Δ_i
—x. Number of malicious nodes in the entire network
—p_{nc}. Fraction of captured nodes in the entire network

2. CRYPTOGRAPHIC PRIMITIVES

In this section, we briefly introduce the cryptographic primitives that we employ throughout the article.

2.1. Secret-Sharing Algorithm

The idea of secret sharing is to start with a secret, divide it into pieces called shares, and distribute them amongst a set of users [Menezes et al. 1997]. The pooled shares of specific subsets of users allow the reconstruction of the original secret. We employ a $(\tilde{T}, t)$ threshold secret-sharing algorithm. Such an algorithm generates $\tilde{T}$ shares such that any combination of at least $t \leq \tilde{T}$ shares suffices to reconstruct the original secret. We suggest Shamir's algorithm that generates $\tilde{T}$ distinct shares, using the following

secret-share generator.

$$\begin{aligned} \mathsf{SSG}_k \ : \ \mathbb{F}_q \ &\longrightarrow \ \mathbb{F}_q, \\ M \ &\longmapsto \ M + \sum_{i=1}^{t-1} (M \gg i)\, k^i. \end{aligned} \tag{1}$$

Here, k is a secret key and $(M \gg i)$ denotes cyclically shifting the message M to the right by i bits. Any combination of t shares generated using distinct secret keys can be used to construct a system of linearly-independent equations, from which the original secret M is uniquely obtained.

2.2. Pseudorandom Function

A pseudorandom function is a family of functions with the property that the input-output behavior of a random instance of the family is computationally indistinguishable from that of a random function [Bellare and Rogaway 2005]. The indistinguishability is measured in terms of the ability of a computationally limited adversary to distinguish the output sequence from a completely randomly generated sequence. A function family is a map $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$, where $\mathcal{K}$ is the set of possible keys, $\mathcal{D}$ is the domain, and $\mathcal{R}$ is the range. For simplicity, we assume $\mathcal{K} = \mathbb{F}_q$, although this is not a necessary condition. For any $k \in \mathcal{K}$, the instance of the family $F_k : \mathcal{D} \to \mathcal{R}$ is defined as $F_k(\cdot) := F(k, \cdot)$. Pseudorandom functions can be implemented using the output feedback mode of block ciphers [Menezes et al. 1997]. In this article, we employ a family of pseudorandom functions with $\mathcal{K} = \mathbb{F}_q$, $\mathcal{D} = \mathbb{Z}_{\geq 0}$ and $\mathcal{R} = \mathbb{F}_q$, such that each of them has a uniform distribution on the range $\mathcal{R}$.

2.3. Hash Tree

Hash trees have many applications in theoretical cryptographic constructions, such as data authentication and commitment schemes [Merkle 1987; Szydlo 2004]. A hash tree on Υ data values $e_1, \ldots, e_\Upsilon$ is a binary tree on leaves $H(e_1), \ldots, H(e_\Upsilon)$, where $H(\cdot)$ is a one-way hash function.

Every arbitrary leaf is assigned a unique *authentication path* that consists of all the values of all nodes that are siblings of the nodes on the unique path, from the root of the tree to that leaf. We note that an authentication path excludes the values of the leaf itself and the root. Therefore, the length of all authentication paths is at most $\lceil \log_2 \Upsilon \rceil$, where $\lceil \cdot \rceil$ is the ceiling function. The authentication path of every leaf is used to verify the authenticity of the corresponding data value. Let $\mathsf{AuthPath}(i; e_1, \ldots, e_\Upsilon)$ be an algorithm that calculates the authentication path of the ith leaf $H(e_i)$. An optimal algorithm is presented in Szydlo [2004], for this purpose, that generates the authentication paths in both time and space $O(\log_2 \Upsilon)$. For every $i \in [\Upsilon]$, the data value e_i is authentic if $r = \mathsf{Auth}(e_i, \mathsf{AuthPath}(i; e_1, \ldots, e_\Upsilon))$, where Auth is an algorithm that takes any leaf value, along with its corresponding authentication path, to generate the root of the tree.

A hash tree for the data values $e_1, \ldots, e_6$ is shown in Figure 1. Here, $h_i = H(e_i)$ for all $i \in [6]$, $h_{12} = H(h_1 \| h_2)$, $h_{34} = H(h_3 \| h_4)$, $h_{56} = H(h_5 \| h_6)$, $h_{1\sim4} = H(h_{12} \| h_{34})$, and eventually the root value is $r = H(h_{1\sim4} \| h_{56})$. The authentication path for the data value e_3 is the sequence h_4, h_{12}, h_{56}. This data value is authentic if $r = H(H(h_{12} \| H(H(e_3) \| h_4)) \| h_{56})$.

3. SECURITY VIA LINEAR NETWORK CODING

The principle behind network coding is to let intermediate nodes encode packets, as well as the source. Network coding is performed in two major steps: (1) computing the minimum cost subgraph and (2) coding and sending data over the subgraph. Linear

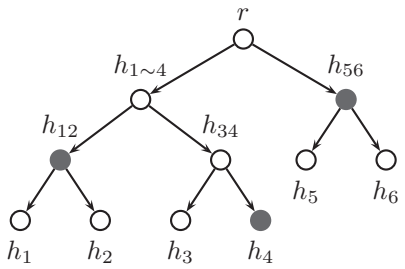


Fig. 1. Hash tree for data values $e_1, \ldots, e_6$.

network coding has recently gained attention due to its interesting properties, such as high throughput and data availability in erasure channels [Ho et al. 2003]. In this section, we explain and motivate the use of linear network coding for WSNs, based on our previous work [Delgosha et al. 2006] and discuss the advantages and disadvantages of this technique. Furthermore, we give a detailed analysis of the bogus-packet propagation problem (due to the pollution attack) caused by linear network coding. Then, in the next section, we will explain our proposed, more advanced scheme in detail.

To illustrate the use of linear network coding, we employ sparse random coding—a subclass of linear network coding—to lower the computational complexity. We assume that the terrain is divided into non-overlapping cells with equal shapes. The sensor nodes are densely and uniformly deployed in the field at random. We assume $\Delta_0, \Delta_1, \ldots, \Delta_\lambda, \Delta_{\lambda+1}$ is a typical sequence of report forwarding cells, starting at the event cell Δ_0 and ending at the sink in $\Delta_{\lambda+1}$. In each intermediate cell, only a fraction of the sensor nodes is involved in the protocol for a particular flow. For each intermediate cell Δ_i, we denote this fraction by the set $\mathcal{V}_i = \{ v_1^i, \ldots, v_T^i \}$. Further, we assume that every node inside a cell can directly communicate with any node in a neighboring cell. This requirement is necessary, since data is transmitted in a cell-by-cell fashion toward the sink. Moreover, we assume that every node knows the location of the cell it resides in, as well as two secret keys (a cell and a node key) obtained by using the location and preloaded information.

An event in the field is sensed by multiple nodes in the event cell Δ_0 because of the dense deployment of the sensor nodes. A CH, selected at the event cell, is responsible for generating a report about the event. To generate a report, the CH broadcasts its own reading inside the event cell (all the inner-cell communications are secured using the cell key). After this information exchange, Ω nodes in Δ_0 endorse the message. For this purpose, every involved node generates a share of the CH's reading using an (Ω, t) threshold secret-sharing algorithm, encrypts that using its unique node key, and sends it back to the CH.

To encode a message vector $\mathbf{d} \in \mathbb{F}_q^\Omega$, the CH generates a sparse matrix[1] $\mathbf{C}_0 \in \mathsf{M}_{\tilde{s},\Omega}^{\tilde{r}}\left(\mathbb{F}_q\right)$ and generates the vector $\mathbf{e}_0 \in \mathbb{F}_q^{\tilde{s}}$ as

$$\mathbf{e}_0 = \mathbf{C}_0\mathbf{d}, \tag{2}$$

where $\tilde{r}$ and $\tilde{s}$ are functions of Ω to ensure decodability at the sink. We note that none of the columns of the sparse matrix $\mathbf{C}_0$ is entirely zero. Therefore, all entries of the message vector appear in the resultant linear combinations. By this process, the CH

[1]To generate such a matrix, the CH picks $\tilde{r}$ entries from every row, randomly chooses their values from $\mathbb{F}_q$, and sets the rest to zero. At the end, if there are any all-zero columns, it makes necessary changes to some rows.

generates $\tilde{s}$ data packets. After the encoding, the vector $\mathbf{e}_0$, along with the coefficients matrix $\mathbf{C}_0$, is transmitted to the next cell Δ_1, as in network coding.

In cell Δ_1, every involved node $v_j^1 \in \mathcal{V}_1$, after receiving $\mathbf{e}_0$ and $\mathbf{C}_0$, randomly generates a sparse matrix[2] $\hat{\mathbf{C}}_{1,j} \in \mathsf{M}_{s,\tilde{s}}^r\left(\mathbb{F}_q\right)$ with $\tilde{s}$ columns, in which r is the number of nonzero entries in every row of the matrix. Then, the node v_j^1 forms the vector $\mathbf{e}_{1,j} \in \mathbb{F}_q^s$ and the matrix $\mathbf{C}_{1,j} \in \mathsf{M}_{s,\Omega}\left(F_q\right)$ as

$$\mathbf{e}_{1,j} = \hat{\mathbf{C}}_{1,j}\mathbf{e}_0, \qquad \mathbf{C}_{1,j} = \hat{\mathbf{C}}_{1,j}\mathbf{C}_0, \tag{3}$$

and transmits the pair $\left(\mathbf{e}_{1,j}, \mathbf{C}_{1,j}\right)$ to the next intermediate cell Δ_2. Therefore, every node in Δ_2 receives the vector $\mathbf{e}_1$ and the matrix $\mathbf{C}_1$, defined as

$$\mathbf{e}_1 = \begin{bmatrix} \mathbf{e}_{1,1} \\ \vdots \\ \mathbf{e}_{1,T} \end{bmatrix} \in \mathbb{F}_q^{sT}, \qquad \mathbf{C}_1 = \begin{bmatrix} \mathbf{C}_{1,1} \\ \vdots \\ \mathbf{C}_{1,T} \end{bmatrix} \in \mathsf{M}_{sT,\Omega}\left(\mathbb{F}_q\right). \tag{4}$$

In general, at every intermediate cell Δ_i, each involved node carries on random network coding in a similar fashion. The only difference is that for all $i \geq 2$, every involved node $v_j^i \in \mathcal{V}_i$ generates a sparse matrix $\hat{\mathbf{C}}_{i,j} \in \mathsf{M}_{s,sT}^r\left(\mathbb{F}_q\right)$ with sT columns, rather than $\tilde{s}$ columns ($sT \geq \tilde{s}$). The report is routed in a cell-by-cell fashion on the shortest path toward the sink.

The decoding at the sink requires Gaussian elimination to solve a linear system for Ω unknowns. Therefore, the sink requires a sufficient number of independent equations to be able to decode for the message. To provide the sufficient properties for decodability and reduce the encoding complexity, we use the analysis in Luby [2002] for LT codes—a class of rateless erasure codes. According to this analysis, for a decodability probability of $1 - \delta$ (where $\delta \in [0, 1]$ is a design parameter), it is sufficient to hold the following equations for the sparsity degree $\tilde{r}$ and the number of equations $\tilde{s}$ generated by the source.

$$\tilde{r} = O\left(\ln\left(\Omega/\delta\right)\right), \tag{5a}$$

$$\tilde{s} = \Omega + O(\sqrt{\Omega}\,\ln^2(\Omega/\delta)). \tag{5b}$$

These choices make the decoding possible with a nearly minimal number of encoding symbols.[3]

As we have previously mentioned, linear network coding is advantageous in data-drop attacks by an adversary, since the redundancy in the data allows the sink to recover the original message packets by receiving sufficient encoded packets. Therefore, linear network coding intrinsically provides a countermeasure against the data-drop attack and provides high data availability at the sink, as well as high throughput. However, this idealistic feature comes with the cost of the bogus-packet propagation problem caused by the pollution attack. In pollution attacks, the attacker node injects corrupted packets into the network. Since each forwarding node combines received packets to form new coded packets, such attacks can cause an epidemic effect in which the corrupted packets from one affected honest node further affect other honest nodes. As a result, by injecting even a few corrupted packets, the attacker can degrade the performance significantly.

[2]This matrix is generated the same way as $\mathbf{C}_0$.

[3]To avoid confusion, we denote r and s by $\tilde{r}$ and $\tilde{s}$, respectively, when they are functions of Ω, as in Equation (5). The intermediate nodes may fix r and s, independent of Ω.

As we discussed in Section 1.1, all previous schemes on securing network coding against the pollution attack either have vulnerabilities to certain attacks or are impractical for implementation. In our previous work [Delgosha et al. 2006], we proposed an approach for encountering the pollution attack problem by adding a MAC to each message packet at the source and checking the integrity of each message packet at some special cells (*checkpoints*) on the way to the sink. To analyze the feasibility of using checkpoints, we uniformly distribute checkpoints on the forwarding path that are responsible for decoding, cleansing, and authenticating the data. The checkpoints can be defined as the cells on the forwarding path that are uniformly spaced (with every two consecutive checkpoints $\psi \in \mathbb{N}$ cells apart from each other). Thus, in the forwarding sequence $\Delta_0, \Delta_1, \Delta_2, \ldots$, the cells $\Delta_{j\psi}$ are the checkpoints for all $j \in \mathbb{N}$. The task of a checkpoint is data cleansing; the active nodes in the checkpoint decode the received packets (to obtain the message packets), verify the MACs, and start random coding anew. The distance between consecutive checkpoints should be selected such that the probability of decodability at the checkpoints is arbitrarily high, even in the presence of malicious nodes injecting noise or bogus packets. In the next section, we analyze this distance to see the feasibility of using checkpoints for packet authentication.

3.1. Bogus-Packet Propagation and Data Availability

As discussed, the use of checkpoints is a potential solution for combating pollution attacks. By positioning the checkpoints far from each other, data may not be decodable because of the rapid propagation of bogus packets. On the other hand, if the checkpoints are too close to each other, computational complexity would be high. Therefore, in this section, we theoretically study the maximum distance between checkpoints to examine the feasibility of using them for packet authentication.

LEMMA 3.1. *Let $i \in \mathbb{N}$ be an arbitrary integer. Assume the nodes in the cell Δ_i receive bogus packets from Δ_{i-1} with probability P_{i-1}. The probability that a node in Δ_i generates a bogus packet is*

$$P_i = 1 - (1 - P_{i-1})^r .$$

PROOF. A linear combination generated by a node in Δ_i will be bogus, even if one of the packets in the linear combination is bogus. Considering that every node generates a linear combination of r packets in its memory, the probability of generating a non-bogus packet is $(1 - P_{i-1})^r$. □

To determine the number of hops, ψ, between any two consecutive checkpoints, it suffices to obtain the distance of the first checkpoint Δ_ψ to the event cell, since all checkpoints are equally distanced. By Lemma 3.1, the probability that one of the packets received by an arbitrary node in Δ_ψ is bogus is

$$P_{\psi-1} = 1 - (1 - P_1)^{(\psi-2)r} , \tag{6}$$

where P_1 is the probability of generating a bogus packet by a malicious node in the cell Δ_1. Every node in Δ_ψ receives sT packets from the nodes in the previous cell ($\Delta_{\psi-1}$). Since any received packet in Δ_ψ is bogus with probability $P_{\psi-1}$, the total number of bogus packets has a binomial distribution. Furthermore, a node in Δ_ψ requires at least Ω legitimate packets to decode for the message packets. Therefore, the probability that any node in Δ_ψ is unable to decode is

$$P_{undec} = \sum_{i=0}^{\Omega-1} \binom{sT}{i} P_{\psi-1}^{sT-i} (1 - P_{\psi-1})^i . \tag{7}$$

Assuming the maximum tolerable probability of undecodability is $\epsilon \in (0,\ 1]$, we must have

$$P_{undec} \leq \epsilon. \tag{8}$$

We denote the CDF of the binomial distribution $B(n,\ p)$ by $Q(p;\ n, x)$, that is,

$$Q(p;\ n, x) := \sum_{i=0}^{x} \binom{n}{i} p^i (1-p)^{n-i}. \tag{9}$$

Using this notation, Equation (8) translates into $Q(1 - P_{\psi-1};\ sT, \Omega - 1) \leq \epsilon$. Since Q is a monotonically decreasing function in p, we have

$$P_{\psi-1} \leq 1 - Q^{-1}(\epsilon;\ sT, \Omega - 1).$$

After combining this result with Equation (6), we obtain

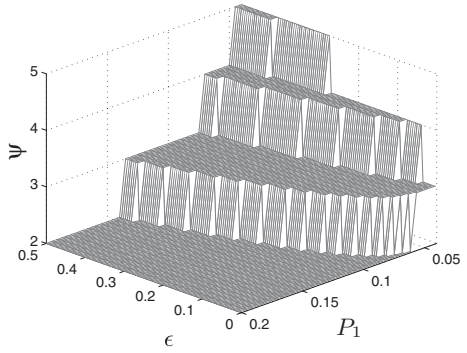
$$\psi \leq 2 + \left\lfloor \frac{\ln Q^{-1}(\epsilon;\ sT, \Omega - 1)}{r \ln(1 - P_1)} \right\rfloor. \tag{10}$$

In Figure 2, the maximum tolerable value of ψ, given by, Equation (10), is plotted versus P_1 and ϵ for different values of sT, Ω, and r. As these curves show, by fixing the parameters sT and Ω, the distance between two checkpoints ψ increases by decreasing r. This is because for small values of r, every packet generated by an intermediate node is the linear combination of only a few received packets. Hence, the probability of generating bogus packets is low, and we can place the checkpoints farther apart from each other. Further, when sT and r are fixed, ψ increases with decreasing Ω, since for small values of Ω, only a few packets are required to decode the information at the checkpoints. Therefore, the probability of decodability is higher, and the checkpoints can be farther from each other. On the other hand, when there is malicious activity in the network, the checkpoints should be located very close to each other to guarantee the decodability of the encoded packets. As the probability of generating a bogus packet by a malicious node in cell Δ_1 (P_1) increases, we end up with no spacing between the checkpoints (i.e., every cell becomes a checkpoint). This observation introduces both high computational overhead and extra delay. Hence, we introduce a novel algorithm called location-aware network-coding security (LNCS) that provides data confidentiality, data authenticity, and high data availability using the advantages of linear network coding while introducing moderate communication and computation overhead. In the next section, we describe LNCS in detail.

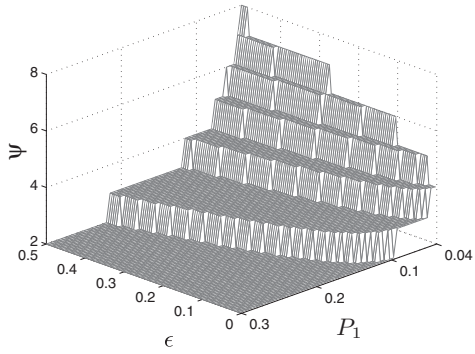
4. LOCATION-AWARE NETWORK-CODING SECURITY

Our proposed scheme takes advantage of location information to enhance the collaboration of sensor nodes. We divide the terrain into non-overlapping cells of equal shape and area. Sensor nodes are randomly deployed in the field. If the node distribution is uniform, we expect almost an equal number of nodes in every cell. Let n be the total number of sensor nodes in the network and N be the average number of nodes in every cell.

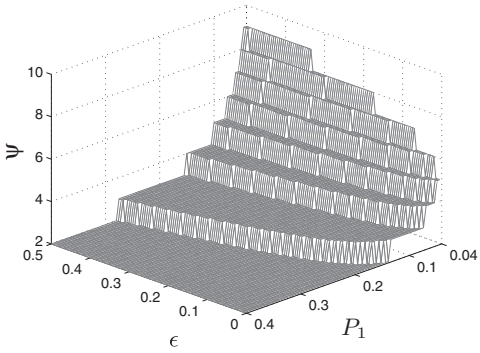
An event detected in a cell is endorsed by the collaboration of many nodes within that cell. Next, it is forwarded toward the sink in a cell-by-cell fashion. Our protocol provides a geographical routing mechanism that chooses the shortest path to the sink. Similar to Section 3, we assume $\Delta_0, \Delta_1, \ldots, \Delta_\lambda, \Delta_{\lambda+1}$ is a typical sequence of report forwarding cells starting at the event cell Δ_0 and ending at the sink $\Delta_{\lambda+1}$. In every cell, only a fraction of the nodes are involved in the protocol. For every cell Δ_i, we denote this fraction by the set $\mathcal{V}_i := \{v_1^i, \ldots, v_{T_i}^i\}$, where $T_i \leq N$ is the size of the set. In addition, for simplicity, we assume $T_i =: T$ for all $i \in [\lambda]$, but T_0 is not necessarily



(a) $sT = 20, \Omega = 10, r = 5$.



(b) $sT = 20, \Omega = 10, r = 3$.



(c) $sT = 20, \Omega = 8, r = 3$.

Fig. 2. Variations of the distance ψ between the checkpoints.

equal to T. In other words, the number of involved nodes T_0 in the event cell is not necessarily the same as that in the intermediate cells. As we will explain later, this distinction provides robustness in designing the scheme for required probabilities of data authenticity and availability. We employ Algorithm 1 with $G = N$ and $g = T_i$ to randomly select the set $\mathcal{V}_i$ that consists of nodes with nonzero tags.

ALGORITHM 1: Tag

Input: Total number of nodes G and the number of nodes $g \leq G$ to be tagged
Output: An ID in $\{0, 1, \ldots, g\}$ for all G nodes

▷ Let $u_1, \ldots, u_G$ be the nodes and $\gamma \geq G$ a fixed integer.

For all $i \in [G]$, the node u_i runs a timer initially set to a random value $t_i \in [\gamma]$. Moreover, it sets its counter $c_i \leftarrow 1$. ;

For all $i \in [G]$, the node u_i listens to the medium when its timer fires. If there is no transmission, it considers the value of c_i as its tag and broadcasts it. Otherwise, it sets $c_i \leftarrow c_i + 1$ and defers its transmission. ;

If the value of the last broadcast is $< g$, then return to 2. ;

Other nodes that never get access to the medium, set their tags to zero.

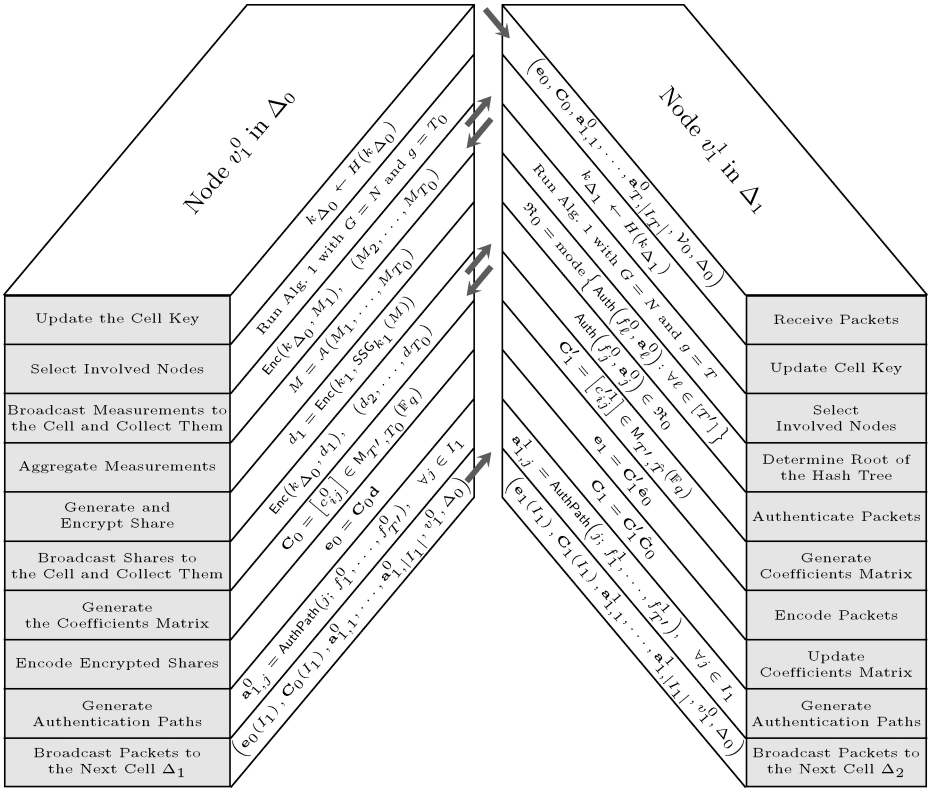


Fig. 3. Steps of LNCS at nodes v_1^0 and v_1^1.

All steps of LNCS at two nodes v_1^0 (in the event cell Δ_0) and v_1^1 (in cell Δ_1) located in two consecutive cells are illustrated in Figure 3. In the rest of this section, we explain the different steps of this scheme.

4.1. Setup

This phase takes place prior to the network deployment, during which every sensor node is loaded with a unique ID $u \in [n]$ and a secret master key K. The entire field

is virtually partitioned into non-overlapping cells of equal areas. For this purpose, we consider cells of hexagonal shape, based on the observation that sensors usually employ omnidirectional antennas [Perrig and Tygar 2003]. Hence, similar to mobile communication systems, a honeycomb-like structure of communication cells provides the most efficient coverage [Stüber 2001]. The advantage of using hexagons over squares is that the deployment field can be covered with a smaller number of cells.

If the communication range of every sensor node is R, we design hexagonal cells with the maximal lateral dimension $R/2$. With this choice, every node inside a cell can directly communicate with another node in a neighbor cell. This requirement is necessary, as the data is transmitted in a cell-by-cell manner to the sink. To minimize energy consumption, the report generated in a cell is routed to the sink on the shortest cell path, called the *forwarding path*. Assuming that every node knows the location of the sink, the use of the cellular structure makes the routing discovery a trivial task.

In the setup phase, the following algorithms are loaded in the memory of every sensor node: a symmetric block cipher Enc_k, a secret-share generator SSG_k, as in Equation (1), a collusion-resistant hash function H, and a pseudorandom function F_k. Each one of these algorithms is a function $\mathbb{F}_q \to \mathbb{F}_q$, and $k \in \mathbb{F}_q$ is a secret key, except the pseudorandom function $F_k : \mathbb{Z}_{\geq 0} \to \mathbb{F}_q$.

4.2. Secure Initialization

The initialization is a short period of time after the network deployment, during which we assume there is no adversarial activity. This assumption is practical, as it has been made by many other sensor-network protocols.

Assume an arbitrary node u that resides in the cell Δ. Using a localization scheme, such as the one in Lazos et al. [2005], the node u obtains the location (x_c, y_c) of the center of Δ. The location information is used to derive a *cell key*, such that

$$k_\Delta := H\left(K \| x_c \| y_c\right), \tag{11}$$

and a *node key*, such that

$$k_u := H\left(K \| x_c \| y_c \| u\right). \tag{12}$$

These keys are used to secure inner-cell communications and report endorsement. At the end of the initialization step, all nodes in the network delete the master key K from their memories.

4.3. Report Generation

Triggered by an event or upon a sink query, all the N nodes within the event cell Δ_0 first update their cell key as

$$k_{\Delta_0} \leftarrow H\left(k_{\Delta_0}\right). \tag{13}$$

(The reason for this update is provided at the end of this section). Then, they run Algorithm 1 with $G = N$ and $g = T_0$ to select a subset of T_0 nodes with the ID set $\mathcal{V}_0$. The nodes tagged zero by this algorithm do not belong to this subset. Hence, they do not participate in the protocol and remain inactive until the next session. Every node $v_i^0 \in \mathcal{V}_0$ performs the following steps in the specified order.

(1) It broadcasts its own sensor reading $M_i \in \mathbb{F}_q$ to other nodes in the set $\mathcal{V}_0$. (Communications within every cell are secured using the cell key.)
(2) Upon the completion of the information exchange, v_i^0 aggregates the T_0 measurements using a resilient aggregation function $\mathcal{A}$. As suggested in Wagner [2004], *median* is a resilient aggregation function that is a good replacement for the mean value (which is shown to be insecure) when the data distribution is symmetric. Let

$M \in \mathbb{F}_q$ be the aggregated value, that is,

$$M := \mathcal{A}\left(M_1, \ldots, M_{T_0}\right). \tag{14}$$

The aggregated message M may also include a time stamp to avoid message-replay attacks.

(3) The node v_i^0 generates and encrypts its share as

$$d_i = \mathsf{Enc}_{k_i}\left(\mathsf{SSG}_{k_i}(M)\right), \tag{15}$$

where $k_i = k_{v_i^0}$ (as in Equation (12)) is the unique secret key of this node, derivable only by the sink. Using a (T_0, t) secret-sharing scheme allows the sink to reconstruct the message M if up to $T_0 - t$ nodes in $\mathcal{V}_0$ are malicious.

(4) The node v_i^0 broadcasts its encrypted share d_i, so all nodes in $\mathcal{V}_0$ have access to the vector $\mathbf{d} := [d_1, \ldots, d_{T_0}]^\dagger \in \mathbb{F}_q^{T_0}$.

(5) Prior to encoding, v_i^0 generates the coefficients matrix $\mathbf{C}_0 = [c_{ij}^0] \in \mathsf{M}_{T',T_0}\left(\mathbb{F}_q\right)$ as

$$c_{ij}^0 := F_{k_{\Delta_0}}(i \| j), \tag{16}$$

where k_{Δ_0} is used as a seed known by all the nodes in $\mathcal{V}_0$ and

$$T' := T_0 + \tau, \quad \tau \geq 0. \tag{17}$$

Since F in Equation (16) is a pseudorandom function with uniform output distribution, the entries of the matrix $\mathbf{C}_0$ are chosen uniformly at random from $\mathbb{F}_q$. Hence, the matrix $\mathbf{C}_0$ is invertible with a high probability. Moreover, since all nodes in $\mathcal{V}_0$ use the same seed, they all derive the same matrix $\mathbf{C}_0$.

(6) The node v_i^0 encodes the vector $\mathbf{d}$

$$\begin{aligned} \mathbf{e}_0 &:= \mathbf{C}_0\,\mathbf{d} \in \mathbb{F}_q^{T'}, \\ &= \left[e_1^0, \ldots, e_{T'}^0\right]^\dagger. \end{aligned} \tag{18}$$

We note that v_i^0 generates more than T_0 packets to compensate for the packets lost or corrupted by noise (due to the medium or adversarial activity) and allow decoding at the sink.

(7) The final step of report generation is constructing the hash tree. To evenly distribute the load of handling this step, we split the packet vector $\mathbf{e}_0$ and the rows of the coefficients matrix $\mathbf{C}_0$ into T_0 groups of almost equal sizes. Let $I_1, \ldots, I_{T_0} \subset [T']$ be a uniform partition of the set $[T']$. For all $i \in [T_0]$, the node v_i^0 generates the sequence of authentication paths $\mathbf{a}_{i,1}^0, \ldots, \mathbf{a}_{i,|I_i|}^0$, where

$$\mathbf{a}_{i,j}^0 := \mathsf{AuthPath}\left(j;\ f_1^0, \ldots, f_{T'}^0\right), \quad \forall j \in I_i. \tag{19}$$

Here, for all $\ell \in [T']$,

$$f_\ell^0 := e_\ell^0 \| c_{\ell 1}^0 \| \cdots \| c_{\ell T_0}^0 \tag{20}$$

is the concatenation of the ith packet with only the corresponding rows of the coefficients matrix. We note that both the generated packets and the entries of the coefficients matrix are involved in the hash tree to prevent an adversary from tampering with any one of them.

(8) Eventually, the node v_i^0 broadcasts the packets as

$$\mathcal{P}_i^0 := \left(\mathbf{e}_0(I_i), \mathbf{C}_0(I_i), \mathbf{a}_{i,1}^0, \ldots, \mathbf{a}_{i,|I_i|}^0, v_i^0, \Delta_0\right) \tag{21}$$

to the next forwarding cell. We note that v_i^0 does not transmit the whole packet vector $\mathbf{e}_0$ and the coefficients matrix $\mathbf{C}_0$; it only transmits the rows determined by the index set I_i.

In summary, the following packets are forwarded from the cell Δ_0 to Δ_1.

$$\mathcal{P}_0 := \left(\mathbf{e}_0, \mathbf{C}_0, \mathbf{a}^0_{1,1}, \ldots, \mathbf{a}^0_{T,|I_T|}, \mathcal{V}_0, \Delta_0\right). \tag{22}$$

Upon detecting the reception of the report by the nodes in Δ_1, all the N nodes update their cell key as $k_{\Delta_1} \leftarrow H(k_{\Delta_1})$ and proceed to authenticate the received packets. Updating the cell key adds to the security of the inner-cell communications. In addition, it changes the random selection of the coefficients matrix $\mathbf{C}_0$ prior to every session, since the cell key is used as a seed to generate this matrix.

4.4. Report Authentication and Filtering

Every nonmalicious node in $\mathcal{V}_0$ transmits approximately T'/T_0 packets from the vector $\mathbf{e}_0$. One possible attack is consuming the energy of the nodes in the forwarding cells. To launch such an attack, a malicious node in $\mathcal{V}_0$ may transmit many more than T'/T_0 packets using the IDs of other nodes in $\mathcal{V}_0$. To prevent this attack, the nodes in $\mathcal{V}_1$ accept, at most, $\lceil T'/T_0 \rceil$ packets all tagged with the same ID. This threshold for other forwarding cells is $\lceil T'/T \rceil$.

To authenticate packets received from Δ_0, the nodes in $\mathcal{V}_1$ require the root of the hash tree. Since it is not transmitted, they assume it is within the set

$$\mathfrak{R}_0 := \text{mode}\left\{\mathsf{Auth}\left(f^0_\ell, \mathbf{a}^0_\ell\right) \quad \forall \ell \in \left[T'\right]\right\}, \tag{23}$$

where f^0_ℓ is given in Equation (20), $\mathbf{a}^0_\ell$ is the authentication path of the packet e^0_ℓ, and *mode* is the statistic that, from a list of data values, returns the ones with the highest repetition. We note that every member of $\mathfrak{R}_0$ is repeated exactly $\rho^0_p \leq T'$ times that represents the number of possible authentic packets. For all $i \in [T]$ and $j \in [T']$, the node v^1_i verifies the authenticity of the packet e^0_j through the test $\mathsf{Auth}(f^0_j, \mathbf{a}^0_j) \in \mathfrak{R}_0$, where f^0_j is related to e^0_j, as in Equation (20). If the packet e^0_j fails the membership test, it is considered bogus; otherwise, it is authentic. Let $\rho^0_v \leq T_0$ be the number of nodes in $\mathcal{V}_0$ that have generated all authentic packets. To proceed to the next step of report forwarding, the number of legitimate packets has to be at least T_0, and the number of nonmalicious nodes has to be at least ζT_0, where $0 \leq \zeta \leq 0.5$ (this threshold is ζT for other intermediate forwarding cells). The possible cases are as follows.

(1) $\rho^0_p \geq T_0$. In this case, any node in the intermediate cell is able to decode the data. Therefore, nodes in $\mathcal{V}_1$ proceed to the report forwarding phase as explained in the following section.
(2) $\rho^0_p < T_0$. Based on the value of ρ^0_v, there are two possible cases.
   (a) $\rho^0_v \geq \zeta T_0$. This case may happen when malicious nodes, in contradiction to their objective, generate some legitimate packets. Taking advantage of the situation, the nodes in $\mathcal{V}_1$ ask for the retransmission of information from the legitimate nodes in the previous cell Δ_0 and discard all packets transmitted by the nodes detected as malicious.
   (b) $\rho^0_v < \zeta T_0$. The report is dropped.

This process is also illustrated in Figure 4. We note that the result of the test $\rho^0_p \lessgtr T_0$ stimulates the necessity for the test $\rho^0_v \lessgtr \zeta T_0$. If $\rho^0_p \geq T_0$, then the data is decodable in the intermediate cell. Thus, there is no need to check the number of nonmalicious nodes.

Setting $\zeta = 0.5$ implies that the majority of the nodes in the previous forwarding cell have to be nonmalicious to continue report forwarding. In this case, the set $\mathfrak{R}_0$ has at most one element, that is, there could be only one authentic message. Nevertheless, for

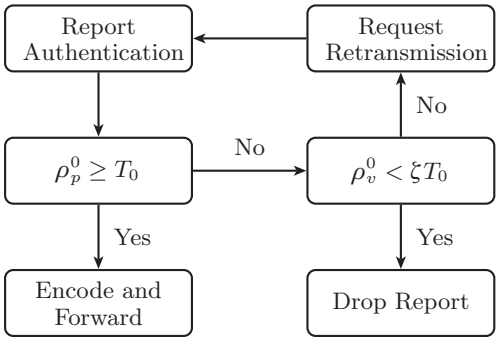


Fig. 4. Flowchart of the report authentication.

$\zeta < 0.5$, the set $\mathfrak{R}_0$ may have more than one element. The implication of this scenario is that there are different reports, each generated by the same number of nodes, but only one of them is authentic. The intermediate nodes cannot determine which report is authentic, since making this decision requires reconstructing the original message from its shares, and the keys used to encrypt the shares are unavailable to the intermediate nodes. As we will explain in Section 5.3, data availability is inversely related to the value of ζ; for small values of ζ, the probability of data drop due to malicious activities of captured nodes is low. However, as we will see in Section 6.2, the payoff for increasing data availability is increasing communication overhead.

4.5. Report Forwarding

Let $J \subseteq [T']$ with $|J| = \hat{T} \leq T'$ be indices of authentic packets after the filtering phase. The nodes in $\mathcal{V}_1$ have access to the common packet vector $\hat{\mathbf{e}}_0 := \mathbf{e}_0(J) \in \mathbb{F}_q^{\hat{T}}$ and coefficients matrix $\hat{\mathbf{C}}_0 := \mathbf{C}_0(J) \in \mathsf{M}_{\hat{T},T_0}(\mathbb{F}_q)$. To encode the authentic packets, the nodes in $\mathcal{V}_1$ generate the coefficients matrix $\mathbf{C}'_1 = [c'^1_{ij}] \in \mathsf{M}_{T',\hat{T}}(\mathbb{F}_q)$ such that:

$$c'^1_{ij} := F_{k_{\Delta_1}}(i \| j). \tag{24}$$

We note that, similar to the event cell, the cell key k_{Δ_1} (known by all the nodes in Δ_1) is used as a seed to randomly generate the matrix $\mathbf{C}'_1$. The next step is performing the network coding and updating the coefficients matrix. For all $i \in [T]$, the node v_i^1 calculates the packet vector, such that

$$\begin{aligned} \mathbf{e}_1 &:= \mathbf{C}'_1 \hat{\mathbf{e}}_0 \in \mathbb{F}_q^{T'}, \\ &= \left[e^1_1, \ldots, e^1_{T'}\right]^\dagger, \end{aligned} \tag{25}$$

and updates the coefficients matrix as

$$\begin{aligned} \mathbf{C}_1 &:= \mathbf{C}'_1 \hat{\mathbf{C}}_0, \\ &= \left[c^1_{ij}\right] \in \mathsf{M}_{T',T_0}\left(\mathbb{F}_q\right). \end{aligned} \tag{26}$$

To evenly distribute the load of generating the authentication information, similar to the event cell, we use a uniform partition $I_1, \ldots, I_T \subset [T']$ of the set $[T']$. Every node v_i^1 generates the sequence of authentication paths $\mathbf{a}^1_{i,1}, \ldots, \mathbf{a}^1_{i,|I_i|}$, where

$$\mathbf{a}^1_{i,j} := \mathsf{AuthPath}\left(j;\ f^1_1, \ldots, f^1_{T'}\right), \quad \forall j \in I_i. \tag{27}$$

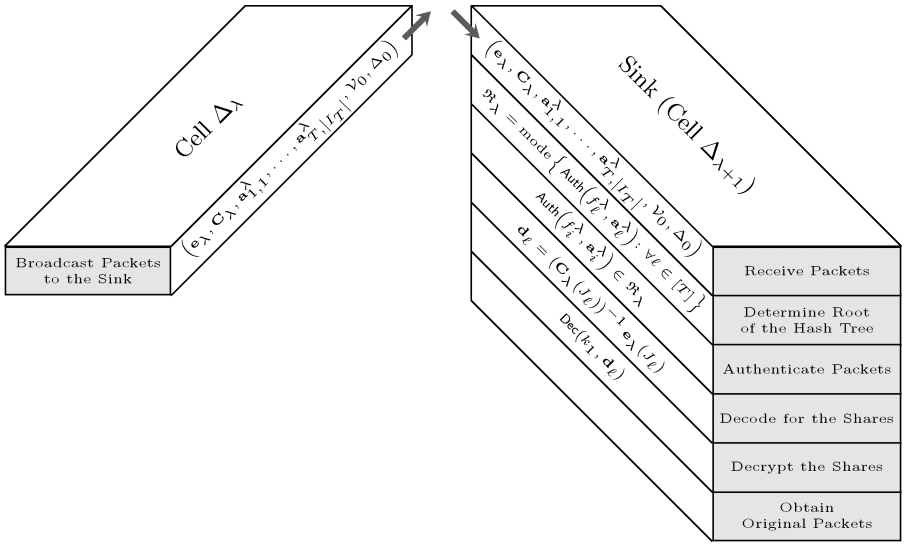


Fig. 5. Final verification and decoding at the sink.

Here, $f_i^1 := e_i^1 \| c_{i1}^1 \| \cdots \| c_{iT_0}^1$ for all $i \in [T']$. Eventually, the node v_i^1 broadcasts the packets

$$\mathcal{P}_i^1 := \left(\mathbf{e}_1(I_i), \mathbf{C}_1(I_i), \mathbf{a}_{i,1}^1, \ldots, \mathbf{a}_{i,|I_i|}^1, v_i^0, \Delta_0\right) \tag{28}$$

to the next forwarding cell. In summary, the following packets are forwarded from the cell Δ_1 to Δ_2.

$$\mathcal{P}_1 := \left(\mathbf{e}_1, \mathbf{C}_1, \mathbf{a}_{1,1}^1, \ldots, \mathbf{a}_{T,|I_T|}^1, \mathcal{V}_0, \Delta_0\right). \tag{29}$$

The message forwarding continues in the same fashion at every cell in the sequence $\Delta_1, \ldots, \Delta_\lambda$. It can be easily shown that for every $i \in \{0, 1, \ldots, \lambda\}$, we have

$$\mathbf{e}_i = \mathbf{C}_i\, \mathbf{d}. \tag{30}$$

4.6. Sink Verification

The final verification at the sink (which is located in the cell $\Delta_{\lambda+1}$) is illustrated in Figure 5. Sink receives the following packets from cell Δ_λ.

$$\mathcal{P}_\lambda := \left(\mathbf{e}_\lambda, \mathbf{C}_\lambda, \mathbf{a}_{1,1}^\lambda, \ldots, \mathbf{a}_{T,|I_T|}^\lambda, \mathcal{V}_0, \Delta_0\right). \tag{31}$$

Let $\mathfrak{R}_\lambda$, as in Equation (23), be the set of possible roots of the hash tree generated at cell $\Delta_{\lambda-1}$. This implies that the packet vector $\mathbf{e}_\lambda$ consists of $\theta := |\mathfrak{R}_\lambda|$ subvectors that are equally likely to be authentic. Let $J_1, \ldots, J_\theta \subset [T']$ be the indices of these subvectors. From Equation (30), we have $\mathbf{e}_\lambda(J_\ell) = \mathbf{C}_\lambda(J_\ell)\, \mathbf{d}_\ell$ for all $\ell \in [\theta]$, where possibly $\mathbf{d}_m = \mathbf{d}$ for only one $m \in [\theta]$. Therefore, for every invertible matrix $\mathbf{C}_\lambda(J_\ell)$, the sink decodes $\mathbf{e}_\lambda(J_\ell)$ as

$$\mathbf{d}_\ell = (\mathbf{C}_\lambda(J_\ell))^{-1}\, \mathbf{e}_\lambda(J_\ell). \tag{32}$$

In the next step, the sink decrypts the shares in every $\mathbf{d}_\ell$, using the secret keys of the nodes in $\mathcal{V}_0$. Then, the sink tries to reconstruct the original message using any t out of the T_0 shares. If the reconstructed message is meaningless, the sink tries a different

set of t shares. After exhausting all possible combinations, the sink repeats the same process for another vector $\mathbf{d}_\ell$. Therefore, the maximum size of the search space is $\binom{T_0}{t}^\theta$.

5. SECURITY EVALUATION OF LNCS

In this section, we evaluate the security of our scheme through analytical measurements of the security services provided: confidentiality, authenticity, and availability. Throughout this section, we assume that there are n nodes in the network, and every cell has approximately N nodes. In addition, we assume that an adversary has randomly captured x nodes in the entire network. Therefore, the fraction of captured nodes is $p_{nc} := x/n$.

5.1. Data Confidentiality

All the communications within an arbitrary cell Δ are secured using the cell key k_Δ. This key is only used in the event cell to block a passive adversary who is only eavesdropping. Capturing a single node in a cell compromises the security of the entire cell. However, it does not affect other cells, since different cells use distinct keys. Even after compromising the security of the event cell, an adversary does not obtain meaningful information, because shares generated at the event cell are encoded using unique keys pairwise between the report-generating nodes and the sink.

The data confidentiality of LNCS is the same as that at LEDS as proposed in Ren et al. [2006]. A cell is compromised when at least one node inside that cell is captured. Therefore, the probability P_{comp} of cell compromise, with respect to data confidentiality, is

$$P_{comp} = 1 - \frac{\binom{n-N}{x}}{\binom{n}{x}}. \tag{33}$$

The curves of this probability are provided in Ren et al. [2006].

5.2. Data Authenticity

One possible attack launched by an adversary is capturing enough nodes in the event cell to forge a report. We note that the shares of an event are generated at the event cell using the secret keys known only to the report endorsing nodes and the sink. Therefore, an adversary is unable to deceive the sink by capturing nodes along the forwarding path.

Since the sink requires at least t consistent packets to reconstruct the data, the adversary has to capture at least t nodes within the event cell. Thus, the probability of data authenticity is given by

$$P_{auth} = \begin{cases} \sum_{\ell=0}^{t-1} p_c(\ell), & x \geq t, \\ 1, & x < t, \end{cases} \tag{34}$$

where $p_c(\ell)$ is the probability that exactly ℓ random nodes in the event cell are captured. This probability is

$$p_c(\ell) = \frac{\binom{N}{\ell}\binom{n-N}{x-\ell}}{\binom{n}{x}}, \qquad \ell = 0, 1, \ldots, N, \tag{35}$$

with the assumption that $\binom{a}{b} = 0$ when either $a < b$ or $b < 0$. The probability of authenticity is plotted in Figure 6 for different values of N and t. In this graph, p_{nc} is the fraction of captured nodes. As these curves show, increasing t improves P_{auth},

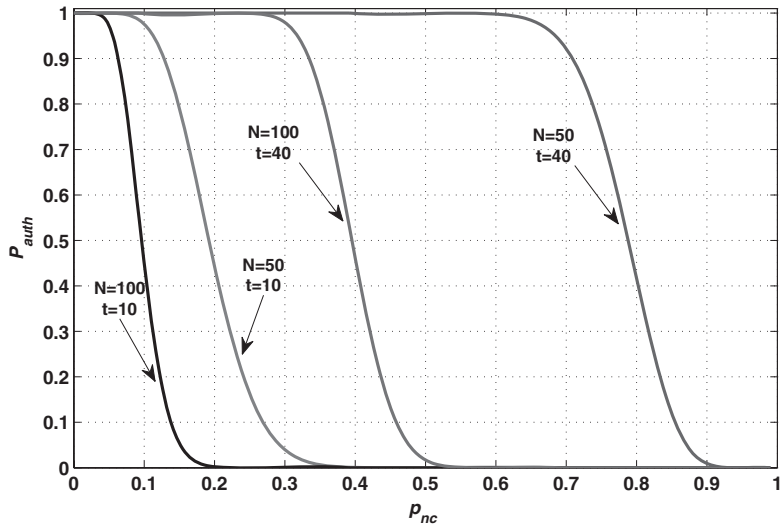


Fig. 6. Probability of authenticity in a network of size $n = 10{,}000$ and cell sizes $N = 50$ and $N = 100$.

since the number of nodes to be captured by an adversary also increases. Another observation is that increasing the cell size degrades the probability of authentication, because the probability that a randomly captured node resides in the cell under study increases. As an example, for $t = 40$, the probability of authenticity is 75% when 36% of the nodes are captured.

5.3. Data Availability

To prevent the sink from receiving a legitimate report, an adversary has to capture a minimum number of involved nodes in an arbitrary forwarding cell Δ_i. As explained in Section 4.4, ζT_{i-1}[4] is the threshold on the number of nonmalicious nodes in $\mathcal{V}_{i-1}$ required by cell Δ_i to forward the message to cell Δ_{i+1}. Therefore, the adversary has to capture at least $T - \zeta T_i + 1$ involved nodes from the set $\mathcal{V}_i$. In light of this observation, the probability of data availability is given by

$$P_{av}^i = \sum_{j=0}^{\lceil T-\zeta T_i \rceil+1} p_{inv}^i(j), \tag{36}$$

where p_{inv}^i is the probability that among the captured nodes in cell Δ_i, exactly j of them are involved. The probability p_{inv}^i is given by the following lemma.

LEMMA 5.1. *The probability that among the nodes captured in cell Δ_i, exactly j of them are involved is*

$$p_{inv}^i(j) = \sum_{\ell=j}^{N-T_i+j} p_c(\ell) \binom{\ell}{j} \left(\frac{T_i}{N}\right)^j \left(1 - \frac{T_i}{N}\right)^{\ell-j}, \tag{37}$$

where $p_c(\ell)$ is given by (35).

[4]We recall that $T_i = T$ for all $i \geq 1$.

PROOF. We define random variables X and Y as follows.

—X = Number of captured nodes in Δ_i.
—Y = Number of involved nodes in Δ_i that are captured.

Using conditional probability, we have

$$p_{inv}^{i}(j) = \Pr(Y = j) = \sum_{\ell=0}^{N} \Pr(Y = j | X = \ell) \Pr(X = \ell).$$

It is clear that $\Pr(X = \ell) = p_c(\ell)$. The term $\Pr(Y = j|X = \ell)$ is the probability that j involved nodes in Δ_i are captured, given that the total number of captured nodes in Δ_i is ℓ. Hence, this probability is nonzero when $\ell \geq j$. Moreover, since the total number of uninvolved nodes in Δ_i is $N - T_i$, we must have $\ell - j \leq N - T_i$, in which $\ell - j$ is the number of uninvolved nodes in Δ_i that are captured. Therefore, in summary, we have

$$\Pr(Y = j|X = \ell) = \begin{cases} \binom{\ell}{j} \left(\frac{T_i}{N}\right)^j \left(1 - \frac{T_i}{N}\right)^{\ell - j}, & \ell \geq j \text{ and } \ell \leq N - T_i + j \\ 0, & \text{Otherwise.} \end{cases},$$

This completes the proof. □

Another possible attack is selective forwarding, in which malicious nodes may refuse to forward the report and simply drop it [Karlof and Wagner 2003]. In our proposed scheme, this attack fails when an adversary randomly captures a few nodes within a forwarding cell. The adversary achieves her goal by capturing only involved nodes in a cell.

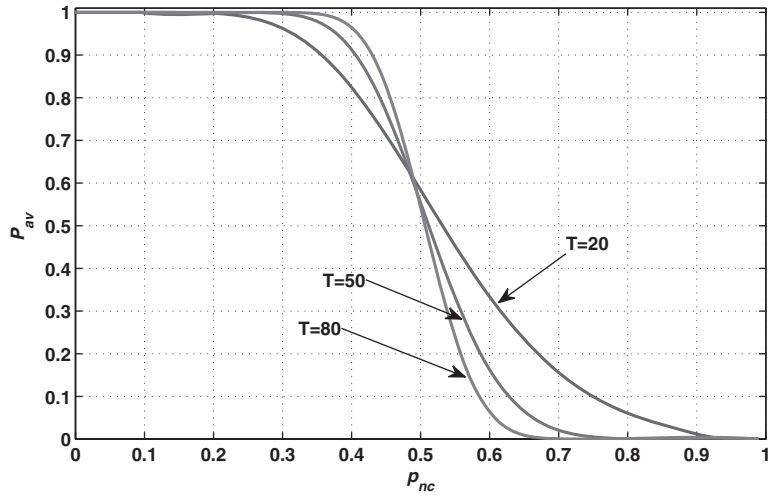
Assuming $T_0 = T$, the probability of data availability P_{av} is plotted in Figure 7 for different values of T and ζ. In all these curves, a general observation is that for small values of p_{nc}, increasing T improves the probability P_{av}, because the adversary has to capture more nodes. However, beyond a specific value of p_{nc}, this effect reverses, that is, increasing T decreases probability P_{av}. This phenomenon becomes clear after recalling that the data is available in a forwarding cell only when this cell has received authentic packets from at least ζT_0 nonmalicious nodes in the previous cell. When there are too many malicious nodes in the network, finding at least ζT nonmalicious nodes becomes difficult for large values of T. Another observation is that decreasing ζ improves the availability, since ζT is the threshold on data availability.

6. PERFORMANCE EVALUATION OF LNCS

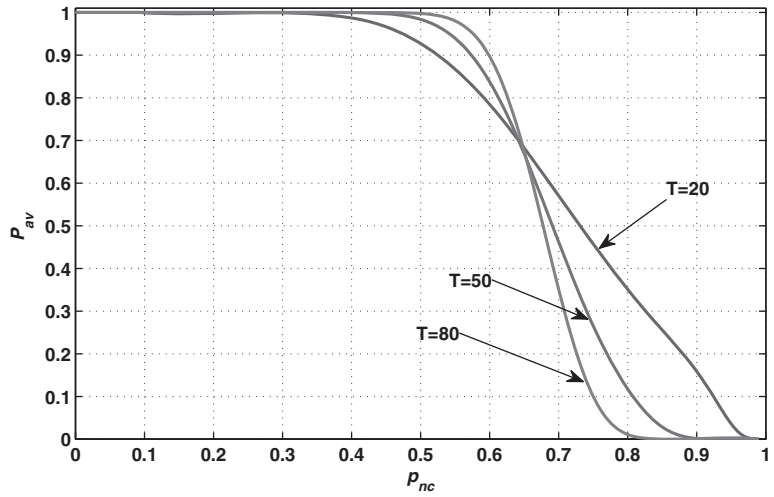
In this section, we evaluate the performance of our scheme in terms of computation and communication overheads per sensor node. Moreover, as explained in Section 4.4, a forwarding cell may request the retransmission of the report from the previous cell. Hence, we calculate the probability of retransmissions considered as communication overhead. Throughout this section, we assume $T' = O(T_0)$, which is a feasible assumption in network coding.

6.1. Computation Overhead

The first phase in our scheme is report generation. The generation of matrix $\mathbf{C}_0$ (as in equation (16)) and the calculation of vector $\mathbf{e}_0$ in Equation (18) are computationally the most expensive calculations in this phase. They both cost $O\left(T_0^2\right)$, that is, the total computational complexity of the report generation. We note that data aggregation in Equation (14) is usually a fast operation. For example, the computational complexity of calculating the median, as suggested in Section 4.3, is $O\left(T_0 \log_2 T_0\right)$ [Cormen et al. 2001].



(a) $\zeta = 1/2$



(b) $\zeta = 1/3$

Fig. 7. Probability of availability in a network with $n = 10{,}000$ and $N = 100$.

The next phase is report authentication and filtering. The only computation performed in this phase is constructing the set $\mathfrak{R}_i$ that consists of the mode of T' data values. This is a relatively cheap operation with complexity $O\left(\log_2 T_0\right)$.

The last phase performed by the sensor nodes is report forwarding. The most expensive computation in this phase is calculating the matrix $\mathbf{C}_i$, as in Equation (26), that costs $T' \hat{T}\, T_0 = O\left(T_0^3\right)$. It is worth noting that the complexity of this calculation decreases when $\mathbf{C}_i$ is sparse, as in Section 3. In that case, the complexity of calculating matrix $\mathbf{C}_i$ would be $O(rsT_0)$. Finally, we conclude that in the worst case (assuming that the matrix $\mathbf{C}_i$ is not sparse), the computational complexity of our scheme is $O\left(T_0^3\right)$ per sensor node.

6.2. Communication Overhead

In this section, we calculate the communication overhead per sensor node, in terms of the number of elements of $\mathbb{F}_q$ transmitted or received, considering the fact that both data transmission and reception consume the same amount of energy.

During the report generation phase, every node in the set $\mathcal{V}_0$ broadcasts its own sensor reading to other nodes in that set. Since the nodes outside this set remain inactive, the communication overhead of this operation is exactly T_0 per node. At the end of the report generation, every node v_i^0 transmits the set of packets $\mathcal{P}_i^0$ in (21) to the next cell. The number of packets in this set is approximately $\frac{T'}{T_0}(1+\log_2 T') + T' = O(T_0)$. Therefore, the communication overhead of report generation is $O(T_0)$ per node.

Every node in a forwarding cell receives a set of packets, as in Equation (22), that approximately consists of $T' + T'T_0 + T\frac{T'}{T_0}\log_2 T' = O(T_0^2)$ packets. In addition, every such node transmits a set of packets, as in Equation (28), that (similar to the report generation phase) consists of $O(T_0)$ packets. Therefore, we conclude that in our scheme, the communication overhead per node is $O(T_0^2)$.

6.3. Retransmission

As explained in Section 4.4, the nodes in a forwarding cell Δ_{i+1} may require the retransmission of information from the previous cell. The retransmission occurs only when the number of nonmalicious nodes detected in the previous cell ρ_v^i is greater than or equal to ζT_i, while the number of authentic packets ρ_p^i is strictly less than T_0. We recall that a nonmalicious node in $\mathcal{V}_i$ generates approximately T'/T_i authentic packets. Therefore, to violate the threshold T_0 on the number of authentic packets ρ_p^i, the adversary has to capture at least

$$\eta_i := \left\lfloor T_i\left(1-\frac{T_0}{T'}\right)\right\rfloor + 1 \tag{38}$$

nodes from $\mathcal{V}_i$. On the other hand, to request retransmission, there has to be at least ζT_i nonmalicious nodes in Δ_i, which implies that the adversary has to capture not more than $T_i(1-\zeta)$ nodes in Δ_i. Considering these facts, retransmission may happen only when $\eta_i < T_i(1-\zeta)$, that is,

$$T_0 > \zeta\,(T_0+\tau) \tag{39}$$

by Equation (17). In this case, the probability of retransmission requested by the nodes in Δ_{i+1} is

$$P_{re}^{i+1} := \sum_{j=\eta_i}^{\lfloor T_i(1-\zeta)\rfloor} p_{inv}^i(j). \tag{40}$$

Here, $p_{inv}^i(j)$, given by Theorem 5.1, is the probability that exactly j involved nodes in the cell Δ_i are captured.

The probability of retransmission for different ratios of overtransmission τ/T is plotted in Figure 8. As the curves in this figure show, increasing the over-transmission decreases the probability of retransmission, which intuitively makes sense. It can also be mathematically explained noting that by increasing τ, threshold η_i in Equation (38) increases as well. Another observation is that when the fraction of captured nodes in the network is high, the probability of retransmission is low. Although practically of less interest, this situation happens when large number of captured nodes causes report drop and breakdown of the protocol.

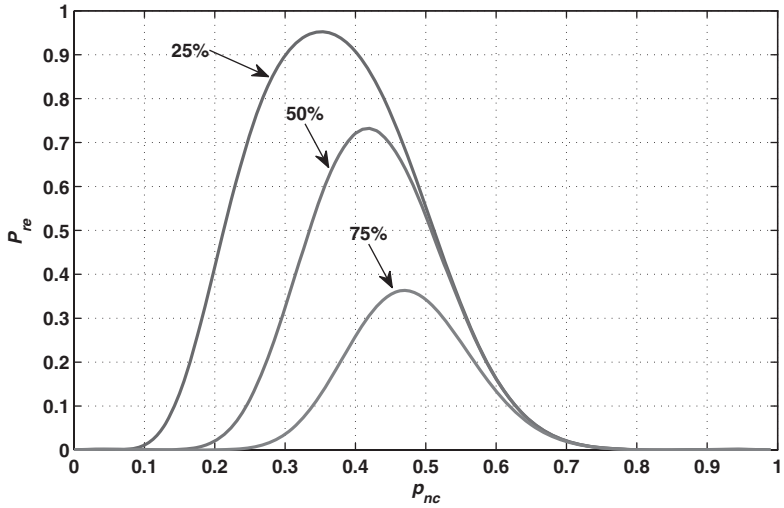


Fig. 8. Probability of retransmission in a network of size $n = 10{,}000$ and other parameters $N = 100$, $T_0 = T = 50$, $\zeta = 0.5$, and $\tau/T \in \{0.25,\ 0.5,\ 0.75\}$.

7. COMPARISON WITH LEDS

In this section, we compare LNCS with LEDS in terms of security and overhead, since LEDS is the only scheme that provides data availability along with data authenticity. We note that none of the other schemes (IHA, SEF, and LBRS) provides data availability, since data is transmitted on a path consisting of single nodes toward the sink. Therefore, a malicious node on the path may drop the report to prevent its reception by the sink. Further, we do not compare LNCS with other networkcoding security schemes, since all previous schemes on securing network coding against pollution attacks either have vulnerabilities to certain attacks or are impractical for implementation, as we discussed in Section 1.1. In the following, we provide a comparison between LNCS and LEDS.

(1) The transmission of data from one cell to another is performed by a single, trustworthy node in LEDS called CH. The existence of such a node cannot be guaranteed. In LNCS, every node involved in the protocol broadcasts part of the generated report. Thus, in terms of reliability in data transmission, LNCS outperforms LEDS.
(2) To provide collaboration between overhearing nodes in LEDS, an excessive amount of redundant communications between adjacent cells is necessary to collect the votes of nodes in the previous cell on the broadcast message. Moreover, this voting mechanism is not practical and will fail, even in the presence of a few malicious nodes. LNCS does not employ a voting system. Therefore, it does not bear with the communication overhead required for such a system.
(3) In LEDS, nodes in a forwarding cell behave independently. Therefore, malicious nodes cause serious data availability and authenticity problems. For example, a malicious node in LEDS may take the role of the CH and modify the legitimate message. The use of network coding in our scheme significantly improves the data availability.

In Figure 9, we compare LNCS with LEDS in terms of data availability. In this experiment, the number of involved nodes in every cells is 40. In LEDS, all the nodes in every forwarding cell participate in the protocol, so we have assumed there are 40 nodes in every cell, for a fair comparison. We also assume $t = T/2$, thus providing a

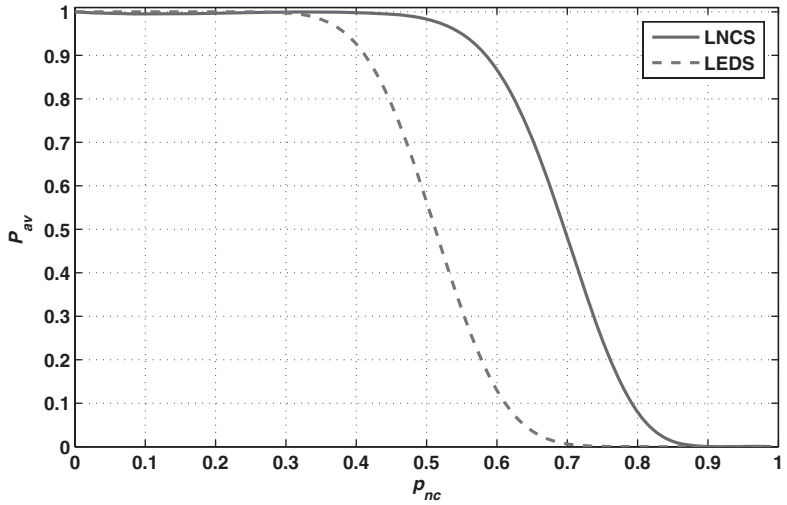


Fig. 9. Comparing data availability between LNCS and LEDS. The network size is $n = 10{,}000$ and other parameters are $N = 50$, $T_0 = T = 40$, and $\zeta = 1/3$. In LEDS, we assume $t = 20$ and the number of nodes per cell is 40.

fair trade-off between data availability and data authenticity. As the figure shows, data availability in LNCS is much higher than that in LEDS. For example, when 50% of the nodes in the entire network are compromised, probabilities of data availability in LNCS and LEDS are 98% and 56%, respectively. The payoff for increasing data availability in LNCS is an increase in communication overhead.

(4) The coefficients matrix used for network coding in LNCS is transmitted from one cell to another. Therefore, in terms of communication overhead, LEDS outperforms LNCS. We note that the communication overhead is the intrinsic drawback of all networks using random network coding.

8. CONCLUSION

In this article, we proposed a package of security services for WSNs in the form of a protocol named location-aware network-coding security (LNCS). As the name implies, the nodes take advantage of location information by dividing the terrain into non-overlapping cells and deriving location binding keys during the secure initialization phase. In LNCS, we have remedied the need for a cluster head that is responsible for report generation and forwarding. A malicious cluster head completely breaks down the security of a protocol. An event detected in the field is sensed by several nodes and aggregated by all of them. Using a secret-sharing algorithm, the aggregated information is divided into several shares that are forwarded toward the sink in a cell-by-cell fashion. To provide data availability, we employ random network coding in our scheme. A comparison with other schemes showed a significant improvement in data availability. As an authentication mechanism, we construct a hash tree on the encoded packets generated at every cell. The packets that fail the authentication test are dropped. Every node in the forwarding cell transmits only a fraction of the generated packets, along with the corresponding authentication information. The sink is the final entity able to reconstruct the original message, using a few shares of the message. A comparison with previous schemes revealed significant improvement in data availability, while maintaining the same level of data confidentiality and authenticity.

REFERENCES

ARAMPATZIS, T., LYGEROS, J., AND MANESIS, S. 2005. A survey of applications of wireless sensors and wireless sensor networks. In *Proceedings of the IEEE International Symposium on Intelligent Control*. 719–724.

AYDAY, E., DELGOSHA, F., AND FEKRI, F. 2007. Location-aware security services for wireless sensor networks using network coding. In *Proceedings of the IEEE Conference Computer Communications (INFOCOM'07)*.

BELLARE, M. AND ROGAWAY, P. 2005. Introduction to modern cryptography. `http://www-cse.ucsd.edu/~mihir/cse207/classnotes.html`.

BONEH, D., FREEMAN, D., KATZ, J., AND WATERS, B. 2009. Signing a linear subspace: Signature schemes for network coding. In *Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography (PKC'09)*.

CHARLES, D., JAIN, K., AND LAUTER, K. 2007. Signatures for network coding. In *Proceedings of the IEEE Conference Information Science and Systems (CISS'06)*. 857–863.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms* 2nd Ed. MIT Press, Cambridge, MA.

DELGOSHA, F., AYDAY, E., CHAN, K., AND FEKRI, F. 2006. Security services in wireless sensor networks using sparse random coding. In *Proceedings of the IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Network (SECON'06)*. 1, 40–49.

DONG, J., CURTMOLA, R., AND NITA-ROTARU, C. 2009. Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec'09)*.

GKANTSIDIS, C. AND RODRIGUEZ, P. 2006. Cooperative security for network coding file distribution. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'06)*.

HO, T., LEONG, B., KOETTER, R., MEDARD, M., EFFROS, M., AND KARGER, D. 2004. Byzantine modification detection in multicast networks using randomized network coding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT'04)*.

HO, T., MÉDARD, M., SHI, J., EFFROS, M., AND KARGER, D. R. 2003. On randomized network coding. In *Proceedings of the Allerton Conference on Communications Control Computing*.

JAGGI, S., LANGBERG, M., KATTI, S., HO, T., KATABI, D., AND MDARD, M. 2007. Resilient network coding in the presence of byzantine adversaries. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'07)*. 616–624.

KARLOF, C. AND WAGNER, D. 2003. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of the International Workshop Sensor Networks Protocols and Applications (SNPA'03)*. 113–127.

KROHN, M. N., FREEDMAN, M. J., AND MAZIÈRES, D. 2004. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proceedings of the IEEE Symposium on Security Privacy (S&P'04)*. 226–240.

LAZOS, L., POOVENDRAN, R., AND ČAPKUN, S. 2005. ROPE: Robust position estimation in wireless sensor networks. In *Proceedings of the International Symposium on Information Processing Sensor Networks (IPSN'05)*. 324–331.

LI, S.-Y. R. ET AL. 2003. Linear network coding. *IEEE Trans. Inf. Theory 49,* 2, 371–381.

LUBY, M. 2002. LT codes. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 271–280.

MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. 1997. *Handbood of Applied Cryptography*. CRC Press.

MERKLE, R. C. 1987. A digital signature based on a conventional encryption function. In *Advanced Cryptology (CRYPTO'87)*, C. Pomerance Ed., Lecture Notes in Computer Science, vol. 293, Springer-Verlag, Berlin, 369–378.

PERRIG, A. AND TYGAR, J. D. 2003. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic, Boston.

REN, K., LOU, W., AND ZHANG, Y. 2006. LEDS: Providing location-aware end-to-end data security in wireless sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'06)*.

STÜBER, G. L. 2001. *Principles of Mobile Communication* 2nd Ed. Kluwer Academic, Boston.

SZYDLO, M. 2004. Merkle tree traversal in log space and time. In *Advanced Cryptology (EUROCRYPT'04)*. C. Cachin and J. Camenisch, Eds., Lecture Notes in Computer Science, vol. 3027, Springer-Verlag, Berlin, 541–554.

WAGNER, D. 2004. Resilient aggregation in sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc Sensor Networks (SASN'04)*. 78–87.

YANG, H., YE, F., YUAN, Y., LU, S., AND ARBAUGH, W. 2005. Toward resilient security in wireless sensor networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Network Computing (MobiHoc'05)*. 34–45.

YE, F., LUO, H., LU, S., AND ZHANG, L. 2005. Statistical en-route filtering of injected false data in sensor networks. *IEEE J. Select Areas Comm. 23,* 4, 839–850.

YU, Z., WEI, Y., RAMKUMAR, B., AND GUAN, Y. 2008. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'08)*.

YU, Z., WEI, Y., RAMKUMAR, B., AND GUAN, Y. 2009. An efficient scheme for securing xor network coding against pollution attacks. In *Proceedings of the IEEE Conference Computer Communications (INFOCOM'09)*.

ZHANG, Y. 2005. The interleaved authentication for filtering false reports in multipath routing based sensor networks. `http://www.cs.pitt.edu/arch/Youtao_Zhang_paper.pdf`.

ZHAO, F., KALKER, T., MEDARD, M., AND HAN, K. 2007. Signatures for content distribution with network coding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT'07)*.

ZHU, S., SETIA, S., JAJODIA, S., AND NING, P. 2004. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy*. 259–271.

Received June 2009; revised January, September 2010; accepted December 2010