Location-Aware Security Services for Wireless Sensor Networks using Network Coding

Erman Ayday, Farshid Delgosha, and Faramarz Fekri
School of Electrical and Computer Eng.
Georgia Institute of Technology
Atlanta, GA 30332-0250, USA
Email: {erman, fekri}@ece.gatech.edu, delgosha@ieee.org

***Abstract*— Security services such as data confidentiality, authenticity, and availability are critical in wireless sensor networks deployed in adversarial environments. Due to the resource constrains of sensor nodes, the existing protocols currently in use in ad-hoc networks cannot be employed in wireless sensor networks. In this paper, we propose a protocol called location-aware network coding security (LNCS) that provides all the aforementioned security services. By dividing the terrain into non-overlapping cells, the nodes take advantage of the location information to derive different location binding keys. An event 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. The key idea in LNCS is that all the nodes involved in the protocol collaborate in every phase. We employ random network coding in our scheme to provide data availability significantly higher than that in other schemes. To generate authentication information, a hash tree is constructed on the generated packets. The packets that fail the authenticity test are considered as bogus and filtered enroute. Every node transmits only a small fraction of the generated packets along the corresponding authentication information to the next cell. The sink is the final entity being able to reconstruct the original message using a few shares of the message. We have provided a comparison between our scheme and previously proposed schemes. The results reveal significant improvement in data availability.**

I. Introduction

Recent advancements in micro-electro-mechanical systems have led to the development of small devices called sensors. A sensor is a low cost and low power device with limited computational power and memory that is equipped with sensing and radio transmission units. Networks of wireless sensors are expected to play key roles in many applications, such as managing energy plants, logistics and inventory, battlefields, and medical monitoring [1]. A typical sensor network is without infrastructure and may include hundreds to several thousands of sensor nodes. Sensor networks are usually connected to the outside world through 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 multi-hops.

This material is based upon work supported by the Army Research Office (ARO) under grant 49586CI.

Security of multi-hop data transfer in wireless sensor networks 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 wireless sensor network are as follows:

Eavesdropping: By listening to the radio channel, the adversary tries to obtain meaningful information.

False Data Injection: In this attack, 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 paper, we propose a new scheme called location-aware network-coding security (LNCS) that provides all the aforementioned security services with moderate communication and computation overhead. The proposed scheme makes extensive use of the 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 the 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 toward 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, in every cell, all the nodes involved in the protocol generate a hash tree of 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 the LNCS [2]. In this type of coding, intermediate nodes process the data by generating random linear combinations of the packets they

receive. This technique is advantageous in the erasure channel model since the redundancy in the data allows the sink to recover the original packets by receiving few encoded packets. The erasure channel also models the packet-drop attack by an adversary. Therefore, random network coding intrinsically provides a countermeasure to data drop.

The rest of this paper is organized as follows. In the rest of this section, we summarize the related work in authentication protocols for wireless sensor networks, highlight the advantages of the LNCS, and present the notation used throughout the paper. In Section II, we briefly review the cryptographic primitives employed in our scheme. The detailed description of the proposed scheme is provided in Section III. In Section IV, we analyze the security of the LNCS. The communication and computation overheads of the LNCS are studied in Section V. In Section VI, we compare the security strength and efficiency of the LNCS with LEDS. Eventually, the concluding remarks are provided in Section VII.

A. Related Work

Interleaved hop-by-hop authentication (IHA) is one of the first works in data authentication for wireless sensor networks [3]. In this scheme, the sensor nodes are organized into clusters. A legitimate report is generated by the collaboration of a minimum number of nodes inside a cluster. Every cluster has a representative that is called the cluster head (CH). The CH is responsible for collecting enough number of message authentication code (MAC) values generated by the collaborating nodes, generating a report, and forwarding it to the sink. The forwarding path from every node to the sink is discovered at the initialization phase.

The authenticity of the report is verified at every hop of the forwarding path to the sink by the aid of the MAC values. For this purpose, authentication chains are discovered and authentication keys are established at the initialization phase of the network operation [4]. 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 these 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 [5]. This scheme is very similar to IHA. The main difference is that associated nodes are not manually determined at the initialization phase. In contrast to IHA, the associated nodes are discovered by a probabilistic approach. In SEF, every node is pre-distributed with the keying materials that are used to establish the authentication keys after the network deployment. The key pre-distribution parameters are selected to guarantee, with a high probability, that any CH is able to establish many authentication keys. The SEF provides data availability similar to IHA. Because of the probabilistic nature of SEF, every node is required to store many keys 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, i.e., 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 [6]. The proposed scheme, called location-based resilient security (LBRS), is conceptually very similar to the SEF. The LBRS localizes the adversarial activities to only the area of the network which is under attack. It 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) [7]. This is a location-aware scheme that provides many security services such as data confidentiality, availability, and authenticity. In LEDS, the 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 the 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 does not seem to exist a practical method to implement this technique. The most logical realization is a voting system that has a high communication overhead and its management introduces a high computational complexity.

B. Outline of Our Scheme

In this paper, we propose location-aware network-coding security (LNCS) that provides data confidentiality, authenticity, and availability for wireless sensor networks. The proposed scheme makes extensive use of node collaboration to reduce the effect of adversarial activities. To enhance node collaboration and localize the effect of malicious nodes, we divide the terrain into non-overlapping cells with equal shapes. The sensor nodes are densely and uniformly at random deployed in the field. Prior to the network deployment, every node is loaded with a master secret key and a unique ID. We assume that a short period of time after the network deployment, the entire network is secure during which every node obtains the location of its cell and two keys (a cell and a node key) using the location and preloaded information. After the initialization phase, all the nodes delete the secret master key from their memories. Sink is the only entity with the ability of deriving the secret keys of all nodes.

An event in the field is sensed by multiple nodes because of the dense deployment of the sensor nodes. To generate a report, the nodes close to the center of stimulus broadcast their own sensor readings to the neighbors involved in the protocol. (All the inter-cell communications are secured using the cell key.) After the completion of information exchange, all the nodes in the event cell, involved in the protocol, have access to the same set of packets. These nodes aggregate the packets using a secure aggregation function such as median. In the next step, every collaborating node generates a share of the aggregated data using a threshold secret-sharing algorithm and encrypts that using its unique node key. To linearly encode the encrypted shares, these nodes generate the same coefficients matrix using a pseudo-random function. The encoding is performed by multiplying the coefficients matrix to the vector of secret shares. The involved nodes generate a hash tree on the encoded packets and the coefficients matrix. Eventually, every node broadcasts only a few encoded packets, the corresponding rows of the coefficients matrix, and the authentication information to the next cell closest to the sink. Every node tags its broadcast with the IDs of its own and its cell. The report is routed in a cell-by-cell fashion on the shortest path toward the sink.

Upon receiving the packets by the nodes in a forwarding cell, these nodes verify the authenticity of the received packets and the coefficients matrix, and drop bogus ones. Similar to the event cell, these nodes generate a common coefficients matrix, encode the authentic packets, and generate a hash tree on the encoded packets and the coefficients matrix. The result is forwarded to the next forwarding cell. Sink is the final check point that verifies the authenticity of the packets. We note that only a fraction of the nodes in every cell take part in the protocol. The remaining nodes remain inactive.

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

1) In contrary to previous schemes, our proposed scheme do 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 a malicious CH completely breaks down the security of the protocol.
2) In the proposed scheme, data authentication is performed without overhearing nodes and voting systems. Such mechanisms, employed by some other schemes, suffer from extensive communication overhead.
3) We employ random network coding in our scheme to generate redundant information that facilitate 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.

C. Notation

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 two is denoted by $\mathbb{F}$. Since the order of the field is fixed throughout the paper, we have dropped it in our notation. For any $n, k \in \mathbb{N}$, the set of all $n \times k$ matrices with entries from $\mathbb{F}$ is denoted by $\mathrm{M}_{n,k}(\mathbb{F})$. For the case $n = k$, we use the short notation $\mathrm{M}_n(\mathbb{F})$. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\dagger$. For any matrix $\mathbf{A} \in \mathrm{M}_{n,k}(\mathbb{F})$ and any set of indices $I \subseteq [\,n\,]$, the symbol $\mathbf{A}(I)$ represents a sub-matrix 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.

In order 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 the 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

II. Cryptographic Primitives

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

A. 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 [8]. The pooled shares of specific subsets of users allow the reconstruction of the original secret. We employ a (T, t) threshold secret-sharing algorithm. Such an algorithm generates T shares such that any combination of at least $t \leq T$ shares suffices to reconstruct the original secret. We suggest Shamir's algorithm that generates T distinct shares using the following secret-share generator.

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

Here, k is a secret key and $(M \gg i)$ denotes cyclically shifting M to the right 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.

B. Pseudo-random Function

A pseudo-random 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 [9]. 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 $\mathcal{K} = \mathbb{F}$ 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 $F_k(\cdot) := F(k, \cdot)$. Pseudo-random functions can be implemented using the output feedback mode of block ciphers [8]. In this paper, we employ a family of pseudorandom functions with $\mathcal{K} = \mathbb{F}$, $\mathcal{D} = \mathbb{N} \cup \{\, 0 \,\}$, and $\mathcal{R} = \mathbb{F}$ such that each of them has a unform distribution on the range $\mathcal{R}$.

C. Hash Tree

Hash trees have many applications in theoretical cryptographic constructions such as *data authentication* and *commitment schemes* [10], [11]. A hash tree on T data values $e_1, \ldots, e_T$ is a binary tree with T leaves. Let $\mathcal{U}$ be the set of all nodes of the tree. Every interior node $u \in \mathcal{U}$ has a left child u_L and a right child u_R. By labeling each left child with a "0" and each right child with a "1", the digits along the path from the root identify each node uniquely. The tree is equipped with a one-way hash function H and a function $\phi : \mathcal{U} \to \mathbb{F}$ that iteratively assigns a value to every node of the tree. The assignment procedure starts at the leaves of the tree by assigning them arbitrary values of $\mathbb{F}$ that are somehow related to the data values. For every interior node $u \in \mathcal{U}$ with the left and right children u_L and u_R, respectively, the value assigned to u is

$$\phi(u) := H\big(\phi(u_L) \,\|\, \phi(u_R)\big). \tag{2}$$

Assuming that $u_1^l, \ldots, u_T^l$ are the leaves of the tree, we suggest the assignment function with the leaf values $\phi\left(u_i^l\right) := H(e_i)$.

Every arbitrary leaf u_i^l of the tree is assigned with 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 the leaf u_i^l. We note that an authentication path excludes the value of the leaf itself and the root. Therefore, the length of all authentication paths is at most $\lceil \log_2 T \rceil$ 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 $\mathsf{AuthPath}\,(i; e_1, \ldots, e_T)$ be an algorithm that calculates the authentication path of the ith leaf u_i^l. An optimal algorithm is presented in [11] for this purpose that generates the authentication paths in both time and space $O(\log_2 T)$. For every $i \in [T]$, the data value e_i is authentic if $r = \mathsf{Auth}\big(e_i, \mathsf{AuthPath}\,(i; e_1, \ldots, e_T)\big)$ 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 $r = H(H(h_{12} \| H(H(e_3) \| h_4)) \| h_{56})$.

III. Location-Aware Network Coding Security

Our proposed scheme takes advantage of the location information to enhance the collaboration of the sensor nodes. We divide the terrain into non-overlapping cells of equal shape and area. The sensor nodes are randomly deployed in the field. If the node distribution is uniform, we expect almost 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. Throughout the paper, 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 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 intermediate cells. As we will explain later, this distinction provides robustness in designing the network for required 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 IDs.

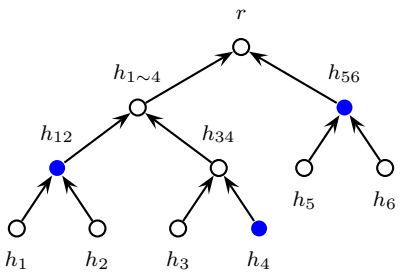


Fig. 1. Hash tree for four data values.

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.

1. 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 $c_i \leftarrow 1$.
3. 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 ID and broadcasts it. Otherwise, it sets $c_i \leftarrow c_i + 1$ and defers its transmission.
4. If the value of the last broadcast is $< g$, then return to 3.
5. Other nodes that never get access to the medium, set their IDs to zero.

In the rest of this section, we explain different steps of the proposed protocol.

A. 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. In addition, descriptions of the following algorithms are loaded in the memory of every sensor node: a secret-key block cipher Enc_k, a secret-share generator SSG_k as in (1), a collusion-resistant hash function H, and a pseudo-random function F_k. Each one of these algorithms is a function $\mathbb{F} \to \mathbb{F}$ and $k \in \mathbb{F}$ is a secret key except the pseudorandom function $F_k : \mathbb{N} \cup \{0\} \to \mathbb{F}$. For the ID assignment, Algorithm 1 with $G = g = n$ is employed.

B. 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 [12], 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*

$$k_\Delta := H(K \| x_c \| y_c) \tag{3}$$

and a *node key*

$$k_u := H(K \| x_c \| y_c \| u)\,. \tag{4}$$

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

C. 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

$$k_{\Delta_0} \leftarrow H(k_{\Delta_0})\,. \tag{5}$$

(The reason for this update is provided at the end of this subsection.) Then, they run Algorithm 1 with $G = N$ and $g = T_0$ to select a subset $\mathcal{V}_0$ consisting of T_0 nodes. 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 $v_i^0 \in \mathcal{V}_0$ broadcasts its own sensor reading $M_i \in \mathbb{F}$ to other nodes in the set $\mathcal{V}_0$. (Communications within every cell are secured using the cell key.) Upon the completion of the information exchange, every node in $\mathcal{V}_0$ aggregates the T_0 measurements using a resilient aggregation function $\mathcal{A}$. As suggested in [13], *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 $M \in \mathbb{F}$ be the aggregation value, i.e.,

$$M := \mathcal{A}\left(M_1, \ldots, M_{T_0}\right). \tag{6}$$

The advantage of using a resilient aggregation function is that the effect of bogus packets is only limited to the malicious nodes generating them.

The next step is report endorsement in which, for all $i \in [\,T_0\,]$, the node v_i^0 calculates the encrypted share

$$d_i = \mathsf{Enc}_{k_i}\big(\mathsf{SSG}_{k_i}\left(M\right)\big) \tag{7}$$

where $k_i = k_{v_i^0}$, as in (4), is the unique secret key of this node that is 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.

To encode the generated shares, every node in $\mathcal{V}_0$ generates the coefficients matrix $\mathbf{C}_0 = \left[\, c_{ij}^0 \,\right] \in \mathrm{M}_{T',T_0}\left(\mathbb{F}\right)$ as follows

$$c_{ij}^0 := F_{k_{\Delta_0}}\left(i \| j\right) \tag{8}$$

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

$$T' := T_0 + \tau, \quad \tau \geq 0. \tag{9}$$

Since F is a pseudorandom function with uniform output distribution, the entries of the matrix $\mathbf{C}_0$ are uniformly at random chosen from $\mathbb{F}$. Hence, the matrix $\mathbf{C}_0$ is invertible with a high probability.

The encoding process is performed by all the nodes in $\mathcal{V}_0$ as follows

$$\begin{aligned} \mathbf{e}_0 &:= \mathbf{C}_0\, \mathbf{d} \in \mathbb{F}^{T'} \\ &= \left[\, e_1^0, \ldots, e_{T'}^0 \,\right]^\dagger \end{aligned} \tag{10}$$

where $\mathbf{d} := \left[\, d_1, \ldots, d_{T_0} \,\right]^\dagger \in \mathbb{F}^{T_0}$. We note that the nodes in $\mathcal{V}_0$ generate more than T packets to compensate for the packets lost or corrupted by noise (due to the medium or adversarial activity) and allow decoding at the sink.

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 $v_i^0 \in \mathcal{V}_0$ generates the sequence of authentication paths $\mathbf{a}_{i,1}^0, \ldots, \mathbf{a}_{i,|I_i|}^0$ where

$$\mathbf{a}_{i,j}^0 := \mathsf{AuthPath}\left(j;\, f_1^0, \ldots, f_{T'}^0\right), \quad \forall j \in I_i. \tag{11}$$

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

$$f_i^0 := e_i^0 \| c_{i1}^0 \| \cdots \| c_{iT_0}^0 \tag{12}$$

is the concatenation of the ith packet with only the corresponding row 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.

Eventually, the node v_i^0 broadcasts the packets

$$\mathcal{P}_i^0 := \Big(\, \mathbf{e}_0\left(I_i\right), \mathbf{C}_0\left(I_i\right), \mathbf{a}_{i,1}^0, \ldots, \mathbf{a}_{i,|I_i|}^0, v_i^0, \Delta_0 \,\Big) \tag{13}$$

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

$$\mathcal{P}_0 := \Big(\, \mathbf{e}_0, \mathbf{C}_0, \mathbf{a}_{1,1}^0, \ldots, \mathbf{a}_{T,|I_T|}^0, \mathcal{V}_0, \Delta_0 \,\Big). \tag{14}$$

Upon detecting the reception of the report by the nodes in Δ_1, all the N nodes update their cell key as $k_{\Delta_1} \leftarrow H\left(k_{\Delta_1}\right)$ 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.

D. 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 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 $\lceil T'/T \rceil$.

In order 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

$$\mathfrak{R}_0 := \mathrm{mode}\Big\{\mathsf{Auth}\left(f_i^0, \mathbf{a}_i^0\right) \,:\, \forall i \in [\,T'\,]\Big\} \tag{15}$$

where f_i^0 is given in (12), $\mathbf{a}_i^0$ is the authentication path of the packet e_i^0, 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 $\rho_p^0 \leq T'$ times which represents the number of possible authentic packets. For all $i \in [\,T\,]$ and $j \in [\,T'\,]$, the node v_i^1 verifies the authenticity of the packet e_j^0 through the test $\mathsf{Auth}\left(e_j, \mathbf{a}_j^0\right) \in \mathfrak{R}_0$. If the packet e_j^0 fails the membership test, it is considered as bogus; otherwise, it is authentic. Let $\rho_v^0 \leq T_0$ be the number of nodes in $\mathcal{V}_0$ that have generated all authentic packets. To proceed to the next step, 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) $\rho_p^0 \geq T_0$: 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 subsection.
2) $\rho_p^0 < T_0$: Based on the value of ρ_v^0, there are two possible cases:
   a) $\rho_v^0 \geq \zeta T_0$: The nodes in $\mathcal{V}_1$ ask for the retransmission of information from the previous cell Δ_0 and discard all packets transmitted by the nodes detected as malicious.
   b) $\rho_v^0 < \zeta T_0$: The report is dropped.

We note that the result of the test $\rho_p^0 \lessgtr T_0$ stimulates the necessity for the test $\rho_v^0 \lessgtr \zeta T_0$. If $\rho_p^0 \geq T_0$, 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, i.e., there could be only one authentic message. Nevertheless, for $\zeta < 0.5$, 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 Subsection IV-C, 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 Subsection V-B, the payoff for increasing data availability is increasing the communication overhead.

E. Report Forwarding

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

$$c'^1_{ij} := F_{k_{\Delta_1}}(i \| j)\,. \tag{16}$$

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

$$\begin{aligned} \mathbf{e}_1 &:= \mathbf{C}'_1\, \hat{\mathbf{e}}_0 \in \mathbb{F}^{T'} \\ &= \left[\, e^1_1, \ldots, e^1_{T'} \,\right]^\dagger \end{aligned} \tag{17}$$

and updates the coefficients matrix

$$\begin{aligned} \mathbf{C}_1 &:= \mathbf{C}'_1\, \hat{\mathbf{C}}_0 \\ &= \left[\, c^1_{ij} \,\right] \in \mathrm{M}_{T',T_0}(\mathbb{F})\,. \end{aligned} \tag{18}$$

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

$$\mathbf{a}^1_{i,j} := \mathsf{AuthPath}\left(j;\, f^1_1, \ldots, f^1_{T'}\right), \quad \forall j \in I_i. \tag{19}$$

Here, $f^1_i := e^1_i \| c^1_{i1} \| \cdots \| c^1_{iT_0}$ for all $i \in [\,T'\,]$. Eventually, the node v_i^1 broadcasts the packets

$$\mathcal{P}^1_i := \left(\mathbf{e}_1(I_i), \mathbf{C}_1(I_i), \mathbf{a}^1_{i,1}, \ldots, \mathbf{a}^1_{i,|I_i|}, v^0_i, \Delta_0 \right) \tag{20}$$

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

$$\mathcal{P}_1 := \left(\mathbf{e}_1, \mathbf{C}_1, \mathbf{a}^1_{1,1}, \ldots, \mathbf{a}^1_{T,|I_T|}, \mathcal{V}_0, \Delta_0 \right). \tag{21}$$

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

$$\mathbf{e}_i = \mathbf{C}_i\, \mathbf{d}. \tag{22}$$

F. Sink Verification

The final verification point, the sink, receives the following packets

$$\mathcal{P}_\lambda := \left(\mathbf{e}_\lambda, \mathbf{C}_\lambda, \mathbf{a}^\lambda_{1,1}, \ldots, \mathbf{a}^\lambda_{T,|I_T|}, \mathcal{V}_0, \Delta_0 \right). \tag{23}$$

Let $\mathfrak{R}_\lambda$, as in (15), be the set of possible roots of the hash tree generated at the cell $\Delta_{\lambda-1}$. This implies that the packet vector $\mathbf{e}_\lambda$ consists of $\theta := |\mathfrak{R}_\lambda|$ sub-vectors that are equally likely to be authentic. Let $J_1, \ldots, J_\theta \subset [\,T'\,]$ be the indices of these sub-vectors. From (22), we have $\mathbf{e}_\lambda(J_\ell) = \mathbf{C}_\lambda(J_\ell)\ \mathbf{d}_\ell$ for all $\ell \in [\,\theta\,]$ where possibly $\mathbf{d}_\ell = \mathbf{d}$ for only one $\ell \in [\,\theta\,]$. Therefore, for every invertible matrix $\mathbf{C}_\lambda(J_\ell)$, the sink decodes $\mathbf{e}_\lambda(J_\ell)$ as

$$\mathbf{d}_\ell = \left(\mathbf{C}_\lambda(J_\ell)\right)^{-1} \mathbf{e}_\lambda(J_\ell)\,. \tag{24}$$

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 $\binom{T_0}{t}^\theta$.

IV. Security Evaluation of the 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 probability of node capture is $p_{nc} := x/n$.

A. 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. This is because the shares generated at the event cell are encoded using the unique keys pairwise between the report-generating nodes and the sink.

The data confidentiality of the LNCS is the same as that in LEDS proposed in [7]. 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

$$P_{comp} = 1 - \frac{\binom{n-N}{x}}{\binom{n}{x}}. \tag{25}$$

The curves of this probability are provided in [7].

B. Data Authenticity

One possible attack launched by an adversary is capturing enough number of 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

$$p_{auth} = \sum_{j=0}^{t-1} p_c(j) \tag{26}$$

where $p_c(j)$ is the probability that exactly j random nodes in the event cell are captured, i.e.,

$$p_c(j) = \frac{\binom{N}{j}\binom{n-N}{x-j}}{\binom{n}{x}}, \quad j = 0, 1, \ldots, N. \tag{27}$$

The probability of authenticity is plotted in Figure 2 for different values of N and t. In this graph, p_{nc} is the probability of node capture. As these curves show, increasing the value of t improves P_{auth} 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. This is because in a large cell, the probability that a randomly captured node resides in the cell under study is high. As an example, for $t = 40$, the probability of authenticity is 75% when 36% of the nodes are captured.

C. 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 Subsection III-D, ζT_i[1] is the threshold on the number of nonmalicious nodes detected in $\mathcal{V}_{i-1}$ that are required by the cell Δ_i to forward the message to the 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

$$P_{av}^i = \sum_{j=0}^{\lceil T-\zeta T_i \rceil + 1} p_{inv}^i(j) \tag{28}$$

where $p_{inv}^i(j)$ is the probability that among the nodes captured in the cell Δ_i, exactly j of them are involved. Using conditional probability, one can easily show that

$$p_{inv}^i(j) = \sum_{\ell=j}^{N-T_i+j} p_c(\ell) \binom{\ell}{j} \left(\frac{T_i}{N}\right)^j \left(1 - \frac{T_i}{N}\right)^{\ell-j}. \tag{29}$$

A possible attack is selective forwarding in which malicious nodes may refuse to forward the report and simply drop it [14]. 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.

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

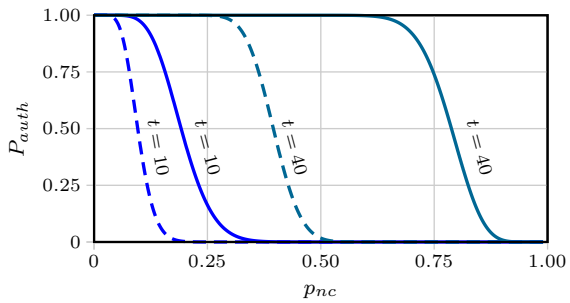
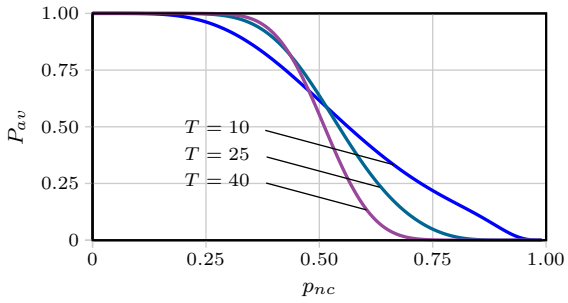


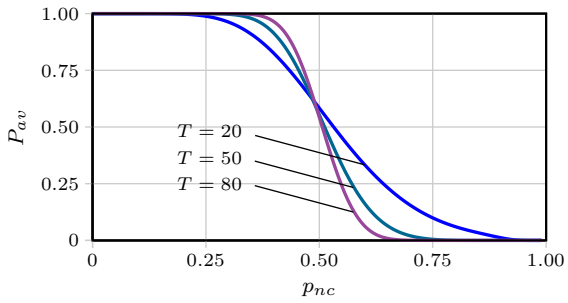
Fig. 2. Probability of authenticity in a network of size $n = 10{,}000$ and cell sizes $N = 50$ (solid line) and $N = 100$ (dashed line).

Assuming $T_0 = T$, the probability of data availability P_{av} is plotted in Figure 3 for different values of N, T, and ζ. In all these curves, for a fixed N, 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 an specific value of p_{nc} this effect reverses, i.e., increasing T decreases the probability P_{av}. This phenomenon becomes clear 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 for a fixed T, increasing N degrades availability since the probability that a node is involved decreases. As the final observation, decreasing ζ improves availability since ζT is the threshold on data availability.

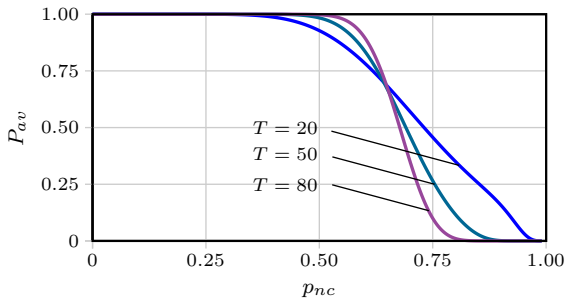
To mention a few numerical examples, in Figure 3(b), at the crossing point of all curves, data availability is 60% when 50% of the nodes are captured. For the same number of malicious nodes and $T = 20$, in Figure 3(c), data availability improves to 93%.



(a) $N = 50$ and $\zeta = 1/2$



(b) $N = 100$ and $\zeta = 1/2$



(c) $N = 100$ and $\zeta = 1/3$

Fig. 3. Probability of availability in a network of size $n = 10{,}000$.

V. Performance Evaluation of the 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 Subsection III-D, a forwarding cell may request the retransmission of the report from the previous cell. Hence, we calculate the probability of retransmissions that is considered as communication overhead. Throughout this section, we assume $T' = O(T_0)$ that is a feasible assumption in network coding.

A. Computation Overhead

The first phase in our scheme is report generation. The generation of the matrix $\mathbf{C}_0$ as in (8) and the calculation of the vector $\mathbf{e}_0$ in (10) 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 report generation. We note that data aggregation in (6) is usually a fast operation. For example, the computational complexity of calculating median, as suggested in Subsection III-C, is $O(T_0 \log_2 T_0)$ [15].

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 $O(\log_2 T_0)$.

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 (18) that costs $T' \hat{T} T_0 = O\left(T_0^3\right)$. Finally, we conclude that the computational complexity of our scheme is $O\left(T_0^3\right)$ per sensor node.

The computational complexity of the LNCS can be reduced if we employ sparse random matrices in the network coding phase. To guarantee the invertibility of a sparse random matrix, with a high probability, we may randomly select the entries of the matrix using the distribution proposed in [16]. In this case, the computational complexity of the LNCS reduces to $O\left(T_0^2 \ln T_0\right)$.

B. Communication Overhead

In this subsection, we calculate the communication overhead per sensor node in terms of the number of elements of $\mathbb{F}$ 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 report generation, every node v_i^0 transmits the set of packets $\mathcal{P}_i^0$ as in (13) to the next cell. The number of packets in this set approximately is $\frac{T'}{T_0}(1 + \log_2 T') + T' = O(T_0)$. 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 (14) that approximately consists of $T' + T' T_0 + T \frac{T'}{T_0} \log_2 T' = O\left(T_0^2\right)$ packets. In addition, every such node transmits a set of packets as in (20) 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 $O\left(T_0^2\right)$.

C. Retransmission

As explained in Subsection III-D, 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 authentic packets ρ_p^i is strictly less than T_0 while the number of nonmalicious nodes detected in the previous cell ρ_v^i is greater than or equal to ζT_i. 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

$$\eta_i := \left\lfloor T_i \left(1 - \frac{T_0}{T'}\right) \right\rfloor + 1 \tag{30}$$

nodes from $\mathcal{V}_i$. In 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 $\eta_i < T_i(1-\zeta)$, i.e.,

$$T_0 > \zeta (T_0 + \tau) \tag{31}$$

by (9). In this case, the probability of retransmission requested by the nodes in Δ_{i+1} is

$$P_{re}^{i+1} := \sum_{j=\eta_i}^{\lfloor T_i(1-\zeta) \rfloor} p_{inv}^i(j). \tag{32}$$

Here, $p_{inv}^i(j)$, given in (29), is the probability that exactly j involved nodes in the cell Δ_i are captured.

The probability of retransmission for different ratios of over-transmission τ/T is plotted in Figure 4. As the curves in this figure show, increasing over-transmission decreases the probability of retransmission, which intuitively makes sense. It can also be mathematically explained noting that by increasing τ, the threshold η in (30) 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 the large number of captured nodes causes the report drop and the breakdown of the protocol.

VI. 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. 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. 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

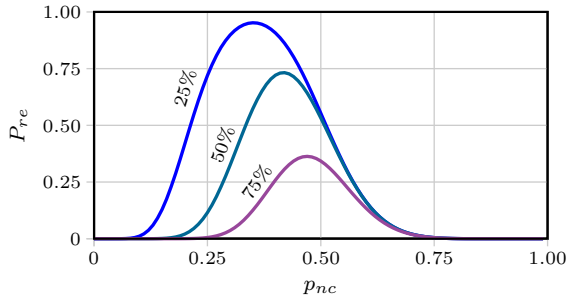


Fig. 4. Probability of retransmission in a network of size $n = 10{,}000$ and other parameters $N = 100$, $T_0 = T = 50$, $\zeta = 0.5$, and $\tau/T \in \{0.25, 0.5, 0.75\}$.

CH. The existence of such 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, excessive amount of redundant communication between adjacent cells is necessary to collect the votes of the nodes in the previous cell on the broadcast message. The LNCS does not employ a voting system. Therefore, it does not bear with the communication overhead required for such a system.
3) In LEDS, the 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 data availability.
   In Figure 5, we compare LNCS with LES in terms of data availability. In this experiment, the number of involved nodes in every cells is 40. Since in LEDS, all the nodes in every forwarding cell participate in the protocol, we have assumed there are 40 nodes in every cell for a fair comparison. The other assumption we have made is $t = T/2$ that provides a fair tradeoff between data availability and 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, the probabilities of data availability in LNCS and LEDS are 98% and 56%, respectively. The payoff for increasing data availability in LNCS is the 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, the LEDS outperforms LNCS. We note that communication overhead is the intrinsic drawback of all networks using random network coding.

VII. Conclusion

In this paper, we proposed a package of security services for wireless sensor networks as a protocol named location-aware network coding security (LNCS). As the name of the protocol implies, the nodes take advantage of the 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 to 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 employed 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 the corresponding authentication information. The sink is the final entity being able to reconstruct the original message using a few shares of the message. A comparison with the previous schemes revealed significant improvement in data availability.

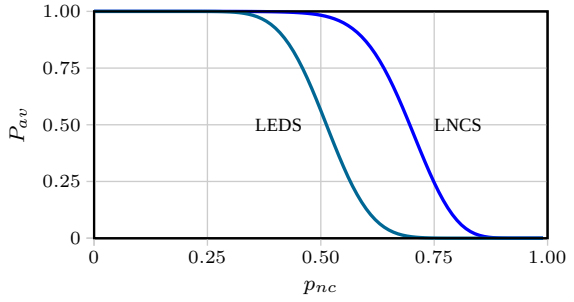


Fig. 5. Comparing data availability between LNCS and LEDS. The network size is $n = 10{,}000$ and other parameters are $N = 50$, $T_0 = T = 40$, and $\zeta = 1/3$. In LEDS, we have assume $t = 20$ and the number of nodes per cell is 40.

References

[1] T. Arampatzis, J. Lygeros, , and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Proc. IEEE Int. Symp. Intelligent Control*, vol. 1. Limassol, Cyprus: IEEE, June 2005, pp. 719–724.

[2] S.-Y. R. Li *et al.*, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[3] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *Proc. IEEE Symp. Security and Privacy*. CA: IEEE Comput. Soc., May 2004, pp. 259–271.

[4] Y. Zhang, "The interleaved authentication for filtering false reports in multipath routing based sensor networks," 2005, available at: http://www.cs.pitt.edu/arch/Youtao_Zhang_paper.pdf.

[5] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 839–850, Apr. 2005.

[6] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward resilient security in wireless sensor networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Network. Comput. - MobiHoc'05*. NY: ACM Press, 2005, pp. 34–45.

[7] K. Ren, W. Lou, and Y. Zhang, "LEDS: Providing location-aware end-to-end data security in wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. - INFOCOM'06*, 2006.

[8] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbood of Applied Cryptography*. NY: CRC Press, 1997.

[9] M. Bellare and P. Rogaway, "Introduction to modern cryptography," 2005, available at: http://www-cse.ucsd.edu/~mihir/cse207/classnotes.html.

[10] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Adv. Cryptol. - CRYPTO'87*, ser. Lecture Notes in Computer Science, C. Pomerance, Ed., vol. 293. Berlin: Springer-Verlag, 1987, pp. 369–378.

[11] M. Szydlo, "Merkle tree traversal in log space and time," in *Adv. Cryptol. - EUROCRYPT'04*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Berlin: Springer-Verlag, May 2004, pp. 541–554.

[12] L. Lazos, R. Poovendran, and S. Čapkun, "ROPE: Robust position estimation in wireless sensor networks," in *Proc. Int. Symp. Inform. Process. Sensor Networks - IPSN'05*. CA: IEEE, 2005, pp. 324–331.

[13] D. Wagner, "Resilient aggregation in sensor networks," in *Proc. ACM Workshop Secur. Ad Hoc Sens. Netw. - SASN'04*. NY: ACM Press, 2004, pp. 78–87.

[14] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *Proc. Int. Workshop Sens. Network Protocols and App. - SNPA'03*. NJ: IEEE, May 2003, pp. 113–127.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MA: The MIT Press, 2001.

[16] J. Blömer, R. Karp, and E. Welzl, "The rank of sparse random matrices over finite fields," *Random Struct. Algorithms*, vol. 10, no. 4, pp. 407–419, July 1997.