Security Services in Wireless Sensor Networks Using Sparse Random Coding

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

***Abstract*— The task of providing security services for wireless sensor networks is not trivial due to the resource constraints of the sensor nodes. An adversary may launch a wide range of attacks including eavesdropping, message forgery, packet dropping, and noise injection. In this paper, we propose random coding security (RCS) that provides protection against all the aforementioned attacks. For this purpose, the proposed protocol makes extensive use of node collaboration and data redundancy. Moreover, using location information, we both localize adversarial activities to the area under attack and enhance routing the data toward the sink. The objectives of using the novel idea of sparse random coding in RCS 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 the RCS in the real case scenario in which the communication media between the sensors is usually modeled as the erasure channel. The existing protocols cannot be trivially modified to suit this realistic situation. In the overall, RCS provides many security services with computation and communication overheads comparable with other schemes.**

I. Introduction

Wireless sensor networks are expected to play key roles in many applications, such as managing energy plants, logistics and inventory, battlefields, and medical monitoring [1]. Security of wireless sensor networks poses new challenges because of the node constraints and networking features. A typical sensor network may include hundreds to several thousands of sensor nodes that are low cost, low power, and have limited computational power and memory. Sensor networks are often infrastructureless and may be deployed randomly.

The task of the sensor nodes is sensing some attributes of the deployment field (such as temperature, motion, illumination, etc.) and reporting the sensed data back to a sink that is responsible for interpreting the data. In order to avoid flooding of redundant information toward the sink, to enhance the latency, and to conserve energy, a cluster head generates a report on behalf of all the other sensors and sends it to the sink. Considering the wide scattering of the sensors in the field, the data generating sensor is usually distanced from the sink. This situation eliminates the possibility of single-hop communication with the sink. Therefore, the report is forwarded to the sink through multi-hops.

The security of multi-hop data transfer becomes very important especially for the networks that are deployed in hostile areas. In these kinds of areas, there is high probability of node compromise, and adversaries may initiate very serious attacks against the network by compromising a few nodes of the network. In this scenario, four major attacks on the network are:
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.
The cryptographic services required to prevent these attacks are *data confidentiality*, *data authenticity*, and *data availability*.

In this paper, we propose a new scheme called random coding security (RCS) that provides all the aforementioned security services with moderate communication and computation overhead. The proposed scheme 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

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

received by multiple nodes in the proximity. In addition, we partition the terrain into non-overlapping hexagonal cells and employ geographical routing. With this technique, we both localize adversarial activities to the area under attack and provide a robust and simple routing and authentication mechanism.

Random network coding is an essential component of the RCS. 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 message 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. This idealistic feature comes with the cost of bogus-packet propagation since only one bogus packet in a linear combination infects the generated packet. To counter this problem, we set check points on the forwarding path, uniformly distributed, that are responsible for decoding, cleansing, and authenticating the data. The distance between consecutive check points can be selected such that the probability of decodability at the check points is arbitrarily high. Therefore, bogus-packet propagation is completely controllable with this method.

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 and also present the notation used throughout the paper. In Section II, we briefly review the main ideas in sparse random coding. The detailed description of the proposed scheme is provided in Section III. In Section IV, we analyze the security of the RCS in terms of the security services claimed to be provided by this protocol. The communication and computation overheads of the proposed scheme are studied in Section V. Finally, the concluding remarks are provided in Section VI.

A. Related Work

Interleaved hop-by-hop authentication (IHA) is one of the first works in data authentication for wireless sensor networks [2]. 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 the 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 both at the initialization phase of the network operation [3]. 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 en-route filtering (SEF) proposed in [4]. 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 material 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 does not provide 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 [5]. The proposed scheme, called location-based resilient security (LBRS), is conceptually very similar to the SEF. However, the data is forwarded toward the sink in a hop-by-hop fashion. Thus, LBRS localizes the adversarial activities to only the area of the network which is under attack. The 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) [6]. 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 random coding security (RCS) which is a package of different security services for wireless sensor networks. The services provided by this scheme are data confidentiality, data authenticity, and data availability. In RCS, the terrain is divided into non-overlapping hexagonal cells of equal sizes, and sensor nodes are uniformly and randomly scattered in the field. At the secure initialization phase of the protocol, sensor nodes obtain the location information of the cell they reside in. Combining this information with the master key pre-loaded in their memories, sensor nodes derive the cell key, the node key, and the authentication key at the end of the initialization phase.

Because of the density of the node deployment, an event is sensed by multiple sensor nodes. A cluster head, selected at the event cell, is responsible for generating a report. The cluster head broadcasts its own reading to its neighbors, and each one of them generates an encrypted share of the message along with a MAC calculated using an authentication key. A legitimate report is generated by collecting enough shares from the neighbor cells. After collecting enough number of shares, the cluster head generates the report by starting random coding. The report is forwarded cell-by-cell toward the sink, and every node at an intermediate cell on the path processes the data in the same fashion. The correlated data in the intermediate cells improves data availability. The authenticity of the data is checked at uniformly distributed cells on the path that are called check points. The nodes at a check point, decode the data, verify the MACs, and generate a new report using healthy packets. The distance between two consecutive check points is set to have a minimum probability of decodability. The sink is the final station in which the data is authenticated.

The contributions of our scheme are summarized in the following.

1) We adopt random network coding in our scheme to provide data availability. The redundant data generated by the intermediate nodes helps recovering the message even when some packets are dropped by malicious nodes. This critical security feature is absent in the previous works such as IHA, SEF, and LBRS. Although LEDS provides data availability using overhearing nodes, the practical implementation of this method is highly complex.
2) The use of hexagonal cells in our scheme provides the best coverage of the terrain. Moreover, RCS localizes the effects of the insider nodes to the area under attack.
3) To the best of our knowledge, the channel effects are not considered in any of the existing authentication protocols. Incorporating the existing coding techniques with any one of the present authentication schemes is nontrivial. If not impossible, we speculate that the high complexity of the final product will render the existing protocols impractical.

C. Notation

The set of positive integers is represented by $\mathbb{N}$. For all $n \in \mathbb{N}$, we define $[n] := \{x \in \mathbb{N} : x \leq n\}$. For an arbitrary set $\mathcal{S}$, the notation $x \in_R \mathcal{S}$ implies that x is uniformly at random is chosen from $\mathcal{S}$. The Galois field $\mathrm{GF}(q)$, where q is a power of 2, 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 $\mathrm{M}_{n,k}(\mathbb{F}_q)$. For the case $n = k$, we use the short notation $\mathrm{M}_n(\mathbb{F}_q)$. For any $r \in \mathbb{N}$, the notation $\mathrm{M}^r_{n,k}(\mathbb{F}_q)$ is used for the set of all sparse matrices $\mathbf{C} \in \mathrm{M}_{n,k}(\mathbb{F}_q)$ 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 the matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\tau$.

In order to facilitate future references, frequently used notations are listed below with their meanings.

CH	Cluster head
N	The average number of nodes inside every cell
T	The total number of packet shares generated by the CH
t	The minimum number of packet shares required to reconstruct the message
ℓ	The number of nodes that participate in random coding at every cell
r	The number of input packets combined to generate a new packet
s	The number of packets generated by every node
$\tilde{r}$	The parameter r when it is a function of T as in (3a)
$\tilde{s}$	The parameter s when it is a function of T as in (3b)

II. Sparse Random Coding

The principle behind network coding is to allow intermediate nodes to encode packets. In network coding, two steps are performed. The first step is computing the minimum cost subgraph, and the second step is to code and send data over the subgraph. The first step is done by solving a linear programming problem (for static networks) which needs complete knowledge about the network topology. Many problems are still open such as the stability under changing conditions (e.g., changing arc costs, changing graph topology), their speed of convergence, and their computational demand [7].

Although the traditional error-correction codes, such as Reed-Solomon codes, are applicable to single-hop erasure channels, their complexity makes them prohibitively expensive for multi-hop networks. Randomized linear network coding has recently gained attention due to its interesting properties such as high throughput and data availability in erasure channels [8]. In RCS, we employ the subclass of sparse random codes because of their efficiency in encoding and decoding [9]. Suppose the source has T packet (i.e., T symbols from $\mathbb{F}_q$) to send to the sink. For encoding a message vector $\mathbf{e}_0 \in \mathbb{F}_q^T$, the source node picks a sparse matrix $\mathbf{C}_0 \in_R \mathrm{M}_{s,T}^r(\mathbb{F}_q)$[1] and generates the vector $\mathbf{e}_1 \in \mathbb{F}_q^s$ of linear combinations as follows.

$$\mathbf{e}_1 = \mathbf{C}_0\,\mathbf{e}_0 \tag{1}$$

We note that none of the columns of the sparse matrix $\mathbf{C}_0$ is entirely zero. Therefore, the signatures of all the entries of the message vector appear in the resultant linear combinations. The vector $\mathbf{e}_1$ along with the coefficients matrix $\mathbf{C}_0$ is transmitted to the next node in the network. An intermediate node, after receiving the packet $\mathbf{e}_{i-1} \in \mathbb{F}_q^s$ and the coefficients matrix $\mathbf{C}_{i-1} \in \mathrm{M}_{s,T}(\mathbb{F}_q)$ for some $i \in \mathbb{N}$, picks the sparse matrix $\mathbf{C}'_i \in_R \mathrm{M}_s^r(\mathbb{F}_q)$ and generates the vector $\mathbf{e}_i \in \mathbb{F}_q^s$ and the matrix $\mathbf{C}_i \in_R \mathrm{M}_{s,T}(\mathbb{F}_q)$ as follows.

$$\mathbf{e}_i = \mathbf{C}'_i\,\mathbf{e}_{i-1}, \qquad \mathbf{C}_i = \mathbf{C}'_i\,\mathbf{C}_{i-1} \tag{2}$$

This node continues the random coding by transmitting $(\mathbf{e}_i, \mathbf{C}_i)$ to the next intermediate node. We note that the complexity of encoding in (2) is $O(rsT)$ since the matrix $\mathbf{C}'_i$ is sparse.

The decoding at the receiver requires Gaussian elimination to solve a linear system for T unknowns. This process has complexity $O\left(T^3\right)$. The receiver requires enough 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 [10] for LT codes, a class of rateless erasure codes. According to this analysis, for the decodability probability of $1-\delta$, where $\delta \in (0,1]$ is a design parameter, the sparsity degree r and the number of equations s generated by the source are sufficient to be set to

$$r = O\left(\ln\left(T/\delta\right)\right) \tag{3a}$$

$$s = T + O\left(\sqrt{T}\,\ln^2\left(T/\delta\right)\right). \tag{3b}$$

These choices make the decoding possible with nearly minimal number of encoding symbols. We note that the intermediate nodes may fix r and s independent of T. To avoid confusion, we denote r and s by $\tilde{r}$ and $\tilde{s}$, respectively, when they are functions of T as in (3).

III. Security via Random Coding

We propose security services for wireless sensor networks using sparse random coding [9]. The proposed scheme, called random coding based security (RCS), consists of the following four phases.

A. Setup

This phase is performed by the sink before the actual deployment of the network in the field. 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 [11]. Hence, similar to mobile communication systems, a honeycomb-like structure of communication cells provides the most efficient coverage [12]. The advantage of using hexagons over squares is that the deployment field can be covered with smaller number of cells with the former choice. The coverage of the hexagonal cells versus that of the square cells is compared in Figure 1. As this figure shows, a hexagonal cell gives a better approximation of the circular wireless transmission-coverage of a sensor.

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 since the data is transmitted in a cell-by-cell manner to the sink. To minimize the energy consumption, the report generated in a cell is routed toward the sink on a shortest cell path that is 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.

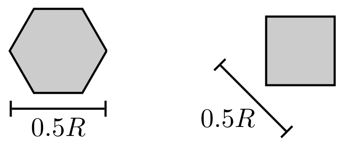


Fig. 1. Hexagonal versus square cell with the same communication range

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

In the setup phase, every node is loaded with the description of a symmetric encryption algorithm $\mathsf{Enc}_k : \mathbb{F}_q \to \mathbb{F}_q$ and a MAC algorithm $\mathsf{MAC}_k : \mathbb{F}_q \to \mathfrak{M}$ where $k \in \mathbb{F}_q$ is the secret key and $\mathfrak{M}$ is the MAC range. As we will explain in Subsection III-C, nodes in the event cell collaborate in the report generation by calculating a share of the message. For this purpose, we employ a (T, t) threshold secret sharing algorithm similar to the Shamir's algorithm [13]. Given a message, this scheme produces T shares in forms of packets, i.e., symbols in $\mathbb{F}_q$. Any subset of at least $t \le T$ shares is enough to reconstruct the original message. An adversary will not obtain any information about the message by knowing at most up to $t-1$ shares. In RCS, we employ the secret sharing algorithm $\mathsf{SSA}_k : \mathbb{F}_q \to \mathbb{F}_q$, where k is the secret key, as follows

$$\begin{array}{rccl} \mathsf{SSA}_k & : & \mathbb{F}_q & \longrightarrow & \mathbb{F}_q \\ & & M & \longmapsto & M + \sum_{i=1}^{t-1} f^{(i)}(M)\, k^i \end{array} \tag{4}$$

Here, $f(\cdot)$ is the shift function that cyclically shifts its input one bit to the right and $f^{(i)}(\cdot)$ is a notation for $i \in \mathbb{N}$ iterative applications of the function f. In addition to the required algorithms, every node is loaded with the values of necessary design parameters as follows:

1) The location (x_0, y_0) of the sink;
2) A master secret key K that is used to derive the cell, node, and authentication keys;
3) The number of cells λ between any two consecutive check points;
4) The secret sharing parameters t and T;
5) The encoding parameters r and s; and
6) The number of encoding nodes ℓ in a cell.

The parameters $\lambda, r, s,$ and ℓ will be explained in the following subsections. Throughout the paper, for reasons that will be explained later, we assume the following inequalities hold.

$$r \le T \le \ell s \tag{5}$$

$$t \le \ell \tag{6}$$

B. Secure Initialization

The initialization phase is a short period of time after the network deployment in which every sensor node u performs the following:

1) It obtains the location (x_c, y_c) of the center of the cell in which it is residing on using a localization scheme [14].
2) Assuming that there are N nodes in every cell, using Algorithm 1 with $n = N$, the node u obtains a unique ID that is an integer $u \in [N]$. (The symbol u is used both as the name and the ID of the node.) We note that the ID of every node is unique only within a cell. With this choice of the node ID, we establish a one-to-one correspondence between the nodes in any two cells.
3) The node u derives the necessary keys required for security services as explained in the following.

Algorithm 1: Node Selection

```
INPUT: Total number of nodes $N$, the number of nodes
       to be selected $n$
OUTPUT: An ID in $\{0, 1, \dots, n\}$ for all $N$ nodes
1. Let $u_1, \dots, u_N$ be the nodes among which we want to
   select only $n \le N$. In addition, let $\gamma \ge N$ be a fixed
   integer.
2. for $i = 1$ to $N$ do The node $u_i$ runs a timer initially set
   to $t_i \in_R [\gamma]$. It also sets its counter $c_i \leftarrow 1$.
3. repeat
4.     For all $i \in [N]$, 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.
5. until The value of the last broadcast is $n$
   ▷ The timers of two nodes may fire at the same time in
     which case collision happens. However, by increasing
     $\gamma$, the probability of collision can be decreased.
6. Other nodes that never get the access of the medium, set
   their IDs to zero.
```

Similar to [15], we assume that the initialization phase after deployment is secure, i.e., none of the nodes is captured.

To explain the derivation of different keys, in the rest of this subsection, we consider an arbitrary node u inside the cell C attributed with the location (x_c, y_c). The first kind of key is the *cell key* k_c that is used to secure the communications within a cell. This key, shared by all the nodes in a cell, is obtained as follows

$$k_c := H\left(K || x_c || y_c\right) \tag{7}$$

where $H(\cdot)$ is a pre-image resistant hash function and the symbol $||$ denotes the concatenation of two binary strings.

The share generated by the node u is encrypted using a unique secret key k_u that is knowing only to the sink. This key is derived from the master key as

$$k_u := H\left(K || x_c || y_c || u\right). \tag{8}$$

Before explaining the derivation of the authentication keys, some terminologies must be defined. In RCS, the authenticity of the data is verified on the forwarding path to the sink at equally distanced cells called *check points*. The sequence of check points forms a chain that is called the *authentication chain*. A cell may be located on multiple authentication chains. Every two consecutive cells on an authentication chain must have a pairwise key to verify the MACs tagged to the report and update them. Therefore, in the initialization phase, all nodes in the cell C discover all their immediate cell neighbors on all authentication chains that C belongs to. This set is denoted by $\mathcal{A}_C$. The number of cells between any two consecutive cells on any authentication chain is a design parameter denoted by λ. As an example, consider the cell C in Figure 2, and assume $\lambda = 1$. The cell C_1 is on the rout toward the sink, and it is only one cell apart from C. Thus, $C_1 \in \mathcal{A}_C$. The cell C itself is on the authentication chains of the cells C_2, C_3, and C_4. This is because these cells are inside the acute angle generated by two rays from the source and adjacent to the cell C. Therefore, $\mathcal{A}_C = \{ C_1, C_2, C_3, C_4 \}$.

For every cell $D \in \mathcal{A}_C$ located at (x_d, y_d), the node u derives the authentication key $k_{u,D}$ as follows

$$k_{u,D} := H\left(K || \left(x_c + x_d\right) || \left(y_c + y_d\right) || u\right). \qquad (9)$$

By the specific choice of the node ID, it is guaranteed the existence of a node in the cell D with the same ID as that of node u. That node in the cell D also obtains the exact same key. Any of these two nodes is able to verify the MAC generated by the other one. At the end of the initialization phase, all the sensor nodes delete the secret master key K from their memories. This prevents the threat of compromising the master key after the initialization phase.

C. Report Generation

To eliminate report forgery by a single node, we require the collaboration of a minimum number of nodes to generate a report. For this purpose, the density of the network and the sensing range of every sensor must be chosen properly.

Triggered by an event or upon a sink query, the nodes in the event cell C_0 elect a cluster head CH.

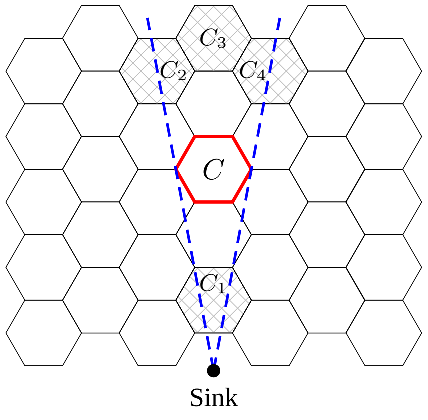


Fig. 2. Set of authentication cells of the cell C

The CH election is performed either randomly or by the use of a reputation scheme (e.g., [16]) in which a node with the highest reputation is selected as the CH. The only necessary assumption is that CH is not compromised[2]. Let M be the sensor reading of the CH. The CH broadcasts the encrypted message $\mathsf{Enc}_{k_c}(M)$ to all its neighbors in the cell C_0 where k_c is the cell key as in (7). Upon receiving such a message, a neighbor node u calculates the encrypted share

$$e_u = \mathsf{Enc}_{k_u}\left(\mathsf{SSA}_{k_u}\left(M\right)\right) \qquad (10)$$

using the secret sharing algorithm in (4) and the node key k_u as in (8). By using a (T, t) secret sharing scheme, the sink is still able to reconstruct the message even if up to $T - t$ nodes generate bogus packets. For every authentication cell $D \in \mathcal{A}_C$, the node u also calculates the MAC of the encrypted share as $m_u = \mathsf{MAC}_{k_{u,D}}\left(e_u\right)$ where $k_{u,D}$ is the authentication key as in (9).

The CH collects T encrypted shares $e_1, \ldots, e_T$ with the corresponding MAC values $m_1, \ldots, m_T$ from the report endorsing nodes $u_1, \ldots, u_T$, respectively. Let

$$\mathbf{e}_{\mathrm{CH}} := \left[e_1, \ldots, e_T \right]^{\tau} \qquad (11a)$$

$$\mathbf{m}_{\mathrm{CH}} := \left[m_1, \ldots, m_T \right]^{\tau} \qquad (11b)$$

and

$$\mathcal{U} := \{ u_1, \ldots, u_T \}. \qquad (12)$$

The CH starts the random coding by selecting a sparse matrix $\mathbf{C}_0 \in_R \mathrm{M}_{\tilde{s},T}^{\tilde{r}}\left(\mathbb{F}_q\right)$ and obtaining the following column vectors.

$$\mathbf{e}_0 := \mathbf{C}_0 \, \mathbf{e}_{\mathrm{CH}} \in \mathbb{F}_q^{\tilde{s}}, \qquad \mathbf{m}_0 := \mathbf{C}_0 \, \mathbf{m}_{\mathrm{CH}} \in \mathbb{F}_q^{\tilde{s}} \qquad (13)$$

Here, $\tilde{r}$ and $\tilde{s}$ are functions of T as in (3) to ensure decodability at the check point. The final report generated by the CH is as follows

$$\mathcal{R} := \left(\mathbf{e}_0, \mathbf{m}_0, \mathbf{C}_0, C_0, C_\lambda, \left(u_1, \ldots, u_T \right) \right) \qquad (14)$$

where C_0 and C_λ are the locations of the event cell and the first check point.[3] This report is broadcast to the next cell on the forwarding path toward the sink.

D. Report Forwarding Using Random Coding

Let $C_0, C_1, C_2, \ldots$ be the sequence of forwarding cells starting at the event cell C_0. At the end of the report generation phase, every node in the cell C_1 has received the report $\mathcal{R}$ in (14). The nodes in C_1 run Algorithm 1 with input $n = \ell$ to obtain a new identifier that we call active ID. At the end of running this algorithm,

[2]Using a reputation system helps increasing the feasibility of this assumption.

[3]We note that there may be different choices for the first check point. The CH randomly chooses one of them.

only ℓ nodes have nonzero active IDs that form the subset $\mathcal{V}_1 := \{ v_{1,1}, \ldots, v_{1,\ell} \}$. The nodes in this subset participate in the random coding and data transfer. The rest of the nodes with active ID zero remain inactive until the next report forwarding cycle.

For every $i \in [\ell]$, the node $v_{1,i} \in \mathcal{V}_1$ performs the following. It selects a sparse matrix $\mathbf{C}'_{1,i} \in_R \mathrm{M}^r_{s,\tilde{s}}(\mathbb{F}_q)$[4], where r and s are constants and calculates the vectors

$$\mathbf{e}_{1,i} := \mathbf{C}'_{1,i}\,\mathbf{e}_0 \in \mathbb{F}_q^s, \qquad \mathbf{m}_{1,i} := \mathbf{C}'_{1,i}\,\mathbf{m}_0 \in \mathbb{F}_q^s \quad (15)$$

and the matrix

$$\mathbf{C}_{1,i} := \mathbf{C}'_{1,i}\,\mathbf{C}_0 \in \mathrm{M}_{s,T}(\mathbb{F}_q)\,. \quad (16)$$

After the calculation is over, the nodes in $\mathcal{V}_1$ take control of the medium according to their active IDs. When the node $v_{1,i}$ gets the access to the channel, it broadcasts $(\mathbf{e}_{1,i}, \mathbf{m}_{1,i}, \mathbf{C}_{1,i})$ to the next forwarding cell.

After the transmission is over, every node in C_2 has stored the following information in its memory.

$$\mathbf{e}_1 := \left[\, \mathbf{e}^\tau_{1,1}, \ldots, \mathbf{e}^\tau_{1,\ell} \,\right]^\tau \in \mathbb{F}_q^{s\ell} \quad (17a)$$

$$\mathbf{m}_1 := \left[\, \mathbf{m}^\tau_{1,1}, \ldots, \mathbf{m}^\tau_{1,\ell} \,\right]^\tau \in \mathbb{F}_q^{s\ell} \quad (17b)$$

$$\mathbf{C}_1 := \left[\, \mathbf{C}^\tau_{1,1}, \ldots, \mathbf{C}^\tau_{1,\ell} \,\right]^\tau \in \mathrm{M}_{s\ell,T}(\mathbb{F}_q) \quad (17c)$$

Similar to C_1, the nodes in the cell C_2 select the subset of active nodes $\mathcal{V}_2 := \{ v_{2,1}, \ldots, v_{2,\ell} \}$. For all $i \in [\ell]$, the node $v_{2,i}$ selects the sparse matrix $\mathbf{C}'_{2,i} \in_R \mathrm{M}^r_{s,s\ell}(\mathbb{F}_q)$[5] and calculates the vectors

$$\mathbf{e}_{2,i} := \mathbf{C}'_{2,i}\,\mathbf{e}_1 \in \mathbb{F}_q^s, \qquad \mathbf{m}_{2,i} := \mathbf{C}'_{2,i}\,\mathbf{m}_0 \in \mathbb{F}_q^s \quad (18)$$

and the matrix

$$\mathbf{C}_{2,i} := \mathbf{C}'_{2,i}\,\mathbf{C}_1 \in \mathrm{M}_{s,T}(\mathbb{F}_q)\,. \quad (19)$$

The same procedure is performed by all the other intermediate cells. Using the formulation in this subsection, it can be easily verified that for any $n \in \mathbb{N}$, the intermediate cell C_n generates the vectors

$$\mathbf{e}_n := \mathbf{C}'_n\,\mathbf{e}_{n-1} \in \mathbb{F}_q^{s\ell}, \qquad \mathbf{m}_n := \mathbf{C}'_n\,\mathbf{m}_{n-1} \in \mathbb{F}_q^{s\ell} \quad (20)$$

and the matrix

$$\mathbf{C}_n := \mathbf{C}'_n\,\mathbf{C}_{n-1} \in \mathrm{M}_{s\ell,T}(\mathbb{F}_q) \quad (21)$$

where $\mathbf{C}'_n := \left[\, \mathbf{C}'^\tau_{n,1}, \ldots, \mathbf{C}'^\tau_{n,\ell} \,\right]^\tau$. We note that every report forwarded to the next cell on the forwarding path conveys the locations of the event cell C_0, the location of the next check point, and the IDs of the report endorsing nodes $(u_1, \ldots, u_T)$ similar to the original report in (14). For simplicity, we have dropped these details from the description of the protocol.

[4]We note that by (5), we have $r \leq T$. Therefore, $\mathbf{C}'_{1,i}$ is sparse.

[5]We note that by (5), we have $r \leq s\ell$. Therefore, $\mathbf{C}'_{2,i}$ is sparse.

E. Check Points

The check points are cells on the forwarding path that are equally spaced with every two consecutive check points $\lambda \in \mathbb{N}$ cells apart from each other. Thus, in the forwarding sequence $C_0, C_1, C_2, \ldots$, the cells $C_{j\lambda}$ are the check points for all $j \in \mathbb{N}$. The task of the check point is data cleansing: the active nodes in the check points decode the data, verify the MACs, and start random coding anew.

To explain the steps taken by a check point, consider an arbitrary check point $C_{j\lambda}$ for some $j \in \mathbb{N}$. Every node in this cell has access to the vectors $\mathbf{e}_{j\lambda-1}, \mathbf{m}_{j\lambda-1} \in \mathbb{F}_q^{s\ell}$ and the matrix $\mathbf{C}_{j\lambda-1} \in \mathrm{M}_{s\ell,T}(\mathbb{F}_q)$. Therefore, by (20) and using Gaussian elimination, each one of them is able to decode for the source vectors $\mathbf{e}_{\mathrm{CH}}$ and $\mathbf{m}_{\mathrm{CH}}$. However, only the nodes with IDs in the set $\mathcal{U}$ as in (12) perform the decoding. We note that each one of these nodes has access to $s\ell$ equations while only T ones are sufficient for decoding. As we will explain in Subsection IV-C, the distance of the check points is designed to guarantee decodability with a high probability. This design criteria implies that any set of T equations are decodable with a high probability.

After the decoding, an arbitrary node v with ID $v \in \mathcal{U}$ is able to verify the authenticity of the packet e_v using the corresponding MAC m_v and an authentication key shared with the previous check point $C_{(j-1)\lambda}$. Based on the verification result, the node v takes one of the following actions:

1) If the MAC is verified, the node v recalculates the MAC of e_v using the authentication key shared with the next check point $C_{(j+1)\lambda}$.
2) Packets with unverified MACs are considered bogus and dropped.

At the end of the MAC verification, the nodes in the check point elect a cluster head CH in a way similar to the event cell C_0. Every node participating in the MAC verification broadcasts the packet along with the new MAC value and its ID. The CH collects at least t packets with the corresponding MACs and generates a new report as in (14). We note that by (6), there are enough number of packets to generate a report. Moreover, since we are using a secret sharing scheme, even if up to $\ell - t$ nodes in the check point generate bogus packets, the message is recoverable at the sink. Using the newly generated report, the CH starts random coding exactly the same way as the cluster head in the event cell.

F. Sink Recovery

In the ideal case, the sink receives T encrypted shares $e_1, \ldots, e_T$ as in (10). It randomly picks a subset of size t and decrypts them using the corresponding node keys. Let the decrypted shares be $g_1, \ldots, g_t$ where $g_i = \mathsf{SSA}_{k_{u_i}}(M)$ for all $i \in [t]$. To recover the message M, the sink solves the following system of linear equations for M.

$$\begin{cases} M + f(M)\, k_{u_1} + \cdots + f^{(t-1)}(M)\, (k_{u_1})^{t-1} = g_1 \\ \vdots \\ M + f(M)\, k_{u_t} + \cdots + f^{(t-1)}(M)\, (k_{u_t})^{t-1} = g_t \end{cases} \tag{22}$$

If the message M is meaningful, then the recovery has been successful. Otherwise, the recovery procedure is repeated with a different subset of encrypted shares.

IV. Security Analysis of the RCS

In this section, we analyze the security of the RCS in terms of data confidentiality, data authenticity, and data availability. Throughout this section, we assume n is the total number of nodes in the network, N is the average number of nodes in every cell, and x is the total number of captured nodes in the entire network.

A. Data Confidentiality

In RCS, all the inner cell communications are secured using the single cell key k_c in (7). Thus, an adversary can break down the security of communications in a cell by just capturing a single node within that cell. However, since during the report generation phase, the cluster head broadcasts its own sensor reading to all its neighbors, a single key shared with all the nodes in the event cell minimizes the communication overhead. We note that since distinct cell keys are used for different cells, the security failure of one cell does not affect the security of any other cells. An alternative approach is using a location-aware key pre-distribution scheme such as the one in [17] that provides a much higher security. With this choice, the cluster head has to encrypt the message separately for each one of its neighbor nodes using the pairwise key with that node. To avoid the communication overhead of this approach, we use a single cell-wide key. This method provides protection against a passive adversary who is only eavesdropping the channel.

The data confidentiality level of RCS is the same as that in LEDS proposed in [6]. A cell is compromised when at least one node inside that cell is captured. Therefore, the probability P_c^{conf} of cell compromise with respect to data confidentiality is

$$P_c^{conf} = 1 - \frac{\binom{n-N}{x}}{\binom{n}{x}}. \tag{23}$$

The curves of this probability are provided in [6].

B. Data Authenticity

In order to deceive the sink, the adversary has to capture at least t nodes in an intermediate cell since the sink requires at least t shares to reconstruct the message. Therefore, the probability P_c^{auth} of cell compromise with respect to data authenticity, i.e., the probability of capturing at least t nodes in a cell, is

$$P_c^{auth} = \sum_{i=t}^{N} P_{c,i} \tag{24}$$

where $P_{c,i}$, given below, is the probability that exactly i nodes are captured inside an arbitrary cell

$$P_{c,i} = \frac{\binom{N}{i}\binom{n-N}{x-i}}{\binom{n}{x}}, \quad \forall i \in \{0, 1, \ldots, N\} \tag{25}$$

The probability of cell compromise P_c^{auth} with respect to data authenticity is plotted versus the number of captured nodes x in Figure 3. As this figure shows, by increasing the number of captured nodes, the probability of cell compromise increases as well. Increasing the number of nodes inside a cell has the same effect since large cells are more susceptible to compromise than small cells. We note that LEDS in [6] provides the same level of data confidentiality.

C. Bogus-Packet Propagation and Data Availability

As explained in Section II, using random coding, every node generates random linear combinations of a few received packets. Therefore, if only one of the received packets is bogus, the generated packet will be bogus as well although the generating node may not be malicious itself. In other words, the noise injected by a malicious node or introduced by the channel rapidly propagates in the network. As explained in the description of the proposed protocol, the task of the check points is to authenticate the packets and drop the noisy ones. By placing the check points far from each other, the data may not be decodable because of the rapid propagation of the bogus packet. In the other hand, if the check points are too close to each other, the computational

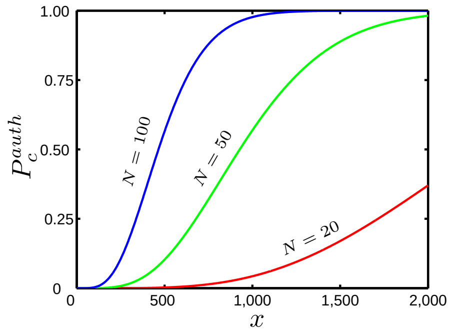


Fig. 3. Data authenticity in RCS for $t = 5$ and $n = 10{,}000$.

complexity would be high. Therefore, in this subsection, we analytically obtain the maximum distance of check points that is a tradeoff between the aforementioned effects.

Let $C_0, C_1, C_2, \ldots$ be the sequence of forwarding cells where C_0 is the event cell.

Lemma 1: Let $i \in \mathbb{N}$ be an arbitrary integer. Assume the nodes in the cell C_i receive bogus packets from C_{i-1} with probability P_{i-1}. The probability that a node in C_i generates a bogus packet is

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

Proof: A linear combination generated by a node in C_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 packets is $(1 - P_{i-1})^r$. ■

In order to determine the value of the parameter λ, it suffices to obtain the distance of the first check point C_λ to the event cell since all check points are equally distanced. By Lemma 1, the probability that one of the packets in the memory of an arbitrary node in C_λ is bogus is

$$P_{\lambda-1} = 1 - (1 - P_1)^{(\lambda-2)r} \tag{26}$$

where P_1 is the probability of generating a bogus packet by a malicious node in the cell C_1. Every node in C_λ has $s\ell$ packets in its memory. Since any number of packets are independently bogus each with probability $P_{\lambda-1}$, the number of bogus packets has a binomial distribution. A node in C_λ requires to receive at least T healthy packets to decode. Therefore, the probability that any node in C_λ is unable to decode is

$$P_{undec} = \sum_{i=0}^{T-1} \binom{s\ell}{i} P_{\lambda-1}^{s\ell-i} (1 - P_{\lambda-1})^i . \tag{27}$$

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

$$P_{undec} \leq \epsilon. \tag{28}$$

For the binomial distribution $B(n, p)$, we define the function $Q(p;\, n, x)$ as follows

$$Q(p;\, n, x) := \sum_{i=0}^{x} \binom{n}{i} p^i (1-p)^{n-i}. \tag{29}$$

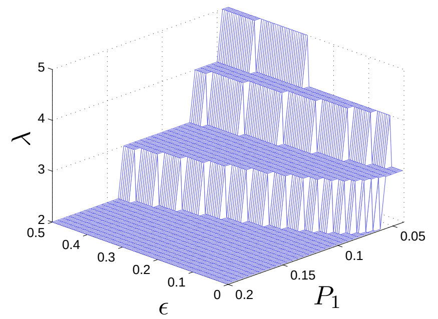
Using this notation, inequality (28) translates to $Q(1 - P_{\lambda-1};\, s\ell, T-1) \leq \epsilon$. Since Q is a monotonically decreasing function, we have

$$P_{\lambda-1} \leq 1 - Q^{-1}(\epsilon;\, s\ell, T-1) .$$

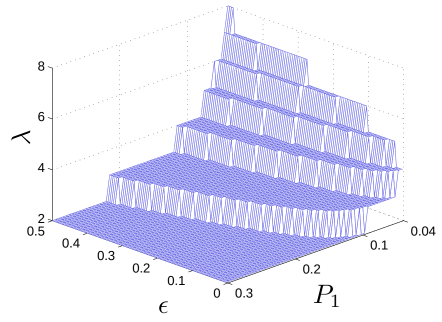
Combining this result with (26), we obtain

$$\lambda \leq 2 + \left\lfloor \frac{\ln Q^{-1}(\epsilon;\, s\ell, T-1)}{r \ln (1 - P_1)} \right\rfloor . \tag{30}$$

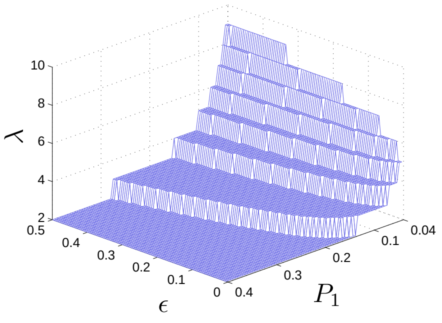
The maximum tolerable value of λ given by (30) is plotted versus P_1 and ϵ in Figure 4 for different values of $s\ell, T,$ and r. As these curves show, by fixing the parameters $s\ell$ and T, the distance between the check points λ increases when we decrease r. This is because for small values of r, every packet generated by an intermediate node is a linear combination of only a few received packets. Hence, the probability of generating bogus packets is low, and we can place the check points further apart from each other. Another observation is that when $s\ell$ and r are fixed, λ increases by decreasing T. This is because for small values of T only a few equations are required to decode the information at the check points. Therefore, the probability of decodability is higher, and the check points can be further from each other.



(a) $s\ell = 20,\ T = 10,\ r = 5$



(b) $s\ell = 20,\ T = 10,\ r = 3$



(c) $s\ell = 20,\ T = 8,\ r = 3$

Fig. 4. The distance λ between the check points

V. Computation and Communication Overheads

We analyze the performance of the RCS in terms of computation and communication overhead and also storage memory in this section. In addition, we compare the results with LEDS in [6].

In order to analyze the complexity of RCS, let $\mathcal{N}$ be the number of nodes involved at each step of the protocol. Moreover, assume $\mathcal{C}_p$ and $\mathcal{C}_m$ denote the computation and communication overheads, respectively. A summary of the overhead analysis of the RCS is provide in Table I. In this table, μ_e and μ_s are the time-complexities of the encryption and secret sharing algorithms, respectively. Combining the results in this table, the overall average complexity of delivering a report to the sink is

$$\mathcal{C}_p = T\left(\mu_e + \mu_s\right) + O\left(\ell L_{av} T\right) + O\left(\ell L_{av} T^3/\lambda\right) \quad (31a)$$
$$\mathcal{C}_m = O\left(\ell L_{av} T^2 \log q\right) \quad (31b)$$

where L_{av} is the average length of the forwarding path. In this calculation, we have used the fact that the coding is performed at every cell. However, the check points are located λ cells away from each other. Therefore, to be fair in calculation, we have taken this fact into account. By using the maximum distance of the check points in (30), we can minimize the computational complexity. We note that the most complex operation in the RCS is the gaussian elimination that has complexity $O\left(T^3\right)$. The computational complexity of the LEDS is $O(T)$.

VI. Conclusion

In this paper, we proposed random coding security (RCS) that provides security services for wireless sensor networks including data confidentiality, data authenticity, and data availability. In the design of RCS, we take advantage of the interesting properties of random coding. This kind of coding intrinsically provides data availability at the receiver. The data on the forwarding path toward the sink may be dropped either because of the malicious activities of insider nodes or because of the erasure property of the communication channel. With a proper design, random coding guarantees data decodability at the receiver. In the RCS, we use the collaboration of the sensor nodes and also secret sharing to generate a report. Therefore, an adversary has to capture a minimum number of nodes to be able to forge a message. Using the location information in RCS limits the malicious activity of the captured nodes to a small area of the field. RCS provides many security services with comparable computation and communication complexity.

TABLE I
Computation and Communication Overhead in RCS

	$\mathcal{N}$	$\mathcal{C}_p/\mathcal{N}$	$\mathcal{C}_m/\mathcal{N}$
Report Endorsement	T	$\mu_e + \mu_s$	$O\left(\log q\right)$
Report Generation	1	$O\left(T \ln T\right)$	$O\left(T^2 \log q\right)$
Coding	ℓ	$O(T)$	$O\left(T^2 \log q\right)$
Check points	ℓ	$O\left(T^3\right)$	$O\left(T^2 \log q\right)$

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. 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] D. S. Lun *et al.*, “Minimum-cost multicast over coded packet networks,” 2006, submitted to IEEE Trans. Inform. Theory, available online at http://arxiv.org/pdf/cs.IT/0503064.

[8] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, “On randomized network coding,” in *Proc. Allerton Conf. Commun., Control and Comput.*, Oct. 2003.

[9] P. Pakzad, C. Fragouli, and A. Shokrollahi, “Coding schemes for line networks,” in *Proc. IEEE Int. Symp. Inform. Theory - ISIT’05*. Adelaide: IEEE, Sept. 2005, pp. 1853–1857.

[10] M. Luby, “LT codes,” in *Proc. IEEE Symp. Found. Computer Science*. Vancouver: IEEE Comput. Soc., Nov. 2002, pp. 271–280.

[11] A. Perrig and J. D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*. Boston: Kluwer Academic, 2003.

[12] G. L. Stüber, *Principles of Mobile Communication*, 2nd ed. Boston: Kluwer Academic, 2001.

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

[14] 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.

[15] R. Anderson, H. Chan, and A. Perrig, “Key infection: Smart trust for smart dust,” in *Proc. IEEE Int. Conf. Network Protocols - ICNP’04*. CA: IEEE Comput. Soc., 2004, pp. 206–215.

[16] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in *Proc. Int. Conf. Mobile Computing and Networking - MobiCom’00*. NY: ACM Press, Aug. 2000, pp. 255–265.

[17] F. Delgosha and F. Fekri, “Key pre-distribution in wireless sensor networks using multivariate polynomials,” in *Proc. IEEE Commun. Soc. Conf. Sensor and Ad Hoc Commun. and Networks - SECON’05*. NJ: IEEE, 2005, CD-ROM.