Low cost security for sensor networks

Emre Atsan, Iris Safaka, Lorenzo Keller, Christina Fragouli
EPFL, Switzerland

***Abstract*—We design and evaluate a lightweight encryption protocol suitable for sensor networks, that enables weak security in the presence of passive eavesdroppers. At every communication round, our protocol creates a key between each sensor node and the sink, by appropriately mixing and coding information packets that the nodes passively overhear. Evaluation using the TOSSIM simulator indicates that with our protocol we can gain significant security benefits at low overhead cost.** [1]

I. Introduction

This work builds on SenseCode, a protocol we developed and implemented in [1], that uses network coding techniques to increase the reliability of data collection in wireless sensor networks; our simulation and experimental results show that in some situations SenseCode can offer up to 30% benefits in terms of reliable data collection as compared to state of the art alternatives, such as the Collection Tree Protocol (CTP) [2].

In this paper we complement SenseCode with a lightweight encryption protocol, that enables to increase security at low additional cost to the unprotected data collection. Our goal is not to offer perfect (information theoretic) security, but instead, to explore what additional security we can achieve, when constrained not to use (or use very few) additional resources, to those used for the data communication.

Our adversary, Eve, is a passive eavesdropper or an honest and curious network node who is limited in terms of presence over space. That is, Eve is located at (any) one location in the network, for example next to one of our sensor nodes, and attempts to overhear all transmissions; moreover, she may not be permanently overhearing. A use case could be when the collected sensor data are to be sold to customers; a customer, to avoid paying, could potentially setup an eavesdropping node and try to acquire the data while it is transmitting towards the sink. Our scheme aims to constrain the stingy customer to learn at most a small fraction of the data. In another use case, perhaps the sensor nodes themselves would like, if possible, not to reveal their individual measurements to the other participating nodes. Our scheme would in this case increase the privacy of the participating nodes.

Our encryption protocol tries to balance the following requirements: (1) we want to use a one-time-pad approach for encryption, as encoding and decoding has low complexity; that is, at every communication round, use a pairwise secret key known to each sensor node and the sink, as one-time pad to encrypt the sensor data; (2) we want to use a different pairwise key per communication round, so that, an adversary that captures the key of one round cannot unlock subsequent rounds; (3) at the same time we want to avoid the overhead of a dedicated "key generation/discovery phase" to construct a pairwise sensor-node sink key before each communication round.

To address these requirements, we propose to construct each key from past data, collected or overheard in previous communication rounds; in particular, keys are constructed as random linear combinations of the past data that both the sensor node and the sink have. An important aspect of this approach is that we can create these keys with low complexity both computationally and in terms of memory requirements. Thus, we can efficiently renew our keys at each communication round, with very low cost, as we essentially reuse the communication of past data for our key generation.

The security of our keys relies on that, an adversary, will not have overheard exactly the same data as any sensor node in the network. This can happen because the adversary may not be present at all communication rounds, or may be physically separated from the sensor node. Even if none of the above happens, with high probability a passive eavesdropper will not overhear exactly the same transmissions as a node due to the random wireless channel variation and losses. We strengthen this effect by using past data across multiple communication rounds to construct our keys, which would require the adversary to overhear the same data as a node over multiple communication rounds as well.

Note that network coding naturally offers weak security, as observed in the literature in the case of multicasting (the same arguments naturally extend for data collection) [3]. Our protocol can be viewed as enhancing this network coding security, where we now use linear combinations not only of current but also past generations (rounds); our main contribution is on how exactly to perform the mixing across generation so that we maintain low overhead suitable for sensor nodes. In our evaluation, we explicitly compare the security benefits our protocol offers with the security provided inherently by network coding.

Our contributions are:

1) We present a protocol that integrates well with SenseCode, as well as other protocols that employ network coding over sensor networks, and offers increased security at low additional complexity.
2) We experimentally evaluate its performance, using the TOSSIM simulator [4] over several settings.
3) We offer an analysis that supports the trends we observe experimentally.

The paper is organized as follows. First, we state our problem in Section II, and we present our encryption protocol in Section III. Next, we build intuition on its security properties in Section IV, and we experimentally evaluate its performance in Section V. Section VI summarizes related work, and we conclude with a discussion in Section VII.

[1]This work was supported by the European Research Council grant NOWIRE ERC-2009-StG-240317.

II. Setup

A. Problem Statement

We consider a sensor network of N sensor nodes and one single collection point, the sink. The network operates in rounds, and in each round, every node i would like to reliably communicate a message x_i, $i = 1...N$, to the sink. Each source message x_i is a sequence of symbols over a finite field $\mathbb{F}_q$. We assume a data collection protocol that enables network coding; for the sink to decode the linear combinations of source messages, in every packet an N-dimensional coding vector $\mathbb{F}_q^N$ is appended. We also assume that our protocol enables node overhearing; we will use SenseCode in this paper to illustrate this [1].

SenseCode creates and maintains a tree structure in the network for the routing of information towards the sink. Each source node generates r packets, where r is the redundancy factor and routes them using a tree path to the sink. Each node having a packet to forward to a next hop along the path, before doing so, linearly combines it in an opportunistic manner with its own message and messages from other nodes, i.e., messages from its children (if any) and *overheard* messages from its neighbors. Clearly, the overheard information from neighboring nodes can be plain messages x_i's or linear combinations of these (uncodable and codable SenseCode packets, respectively). The sink collects all the forwarded information and tries to decode, so as to obtain the x_i's.

Our goal is to design a practical protocol, on top of SenseCode operations (Fig. 1), that enables each node to securely communicate its messages to the sink, under the presence of an adversary, who is described in the next section.

B. Adversary Model

We are interested in evaluating our protocol in the presence of a passive adversary, Eve, who eavesdrops on the communication. We assume in this paper that Eve is in fact an *honest but curious* sensor node, who follows the protocol but at the same time attempts to extract information regarding the other nodes' messages. Any node in our network could be Eve, we have no information regarding her location in the network. Eve is assumed to have limited network presence, that is, she is restricted to one (any one) location in the network, but we do not make any assumptions about her computational and memory capabilities.

C. Performance metrics

We are interested in a form of weak security. That is, Eve gets no meaningful information about a specific message x_i. In particular, if x_i and x_k take i.i.d. binary values and Eve receives $x_i \oplus x_k$ we consider both x_i and x_k to be *secure* from Eve as she gets no meaningful information about them.

The *reliability* for a node i is the percentage of messages that were communicated securely to the sink from node i during the whole operation of the network, given that one message x_i^t is generated at each round t. It is defined as:

$$\rho_i = \frac{\#\ secure\ x_i\ 's\ from\ any\ other\ node\ j}{\#\ rounds},\ \forall j \neq i.$$

The *average reliability* ρ, captures the performance of our protocol with the respect to the whole network. I.e., $\rho = \text{avg}(\rho_1, \ldots, \rho_N)$.

Note that our security metric is pessimistic in two ways: (i) we consider a packet x_i to be not secure even if it is secure from all other sensor nodes in the network but one; (ii) at every round, we implicitly assume that Eve is the node that can at that round decode, thus we give her a strong advantage.

III. Protocol Description

Our protocol aims to exploit the fact that each node of the network will overhear (and have in common with the sink) a random subset of linear combinations of the source symbols, as dictated by the network topology and the channel conditions. More precisely, every node and the sink share a common collection of x_i's and linear combinations of these, over multiple communication rounds. We can use this common information to encrypt future data of node i from an adversary that does not have full knowledge of the shared data.

We define the following parameters for our protocol:

- μ : number of rounds in the past from which we select packets to be combined to create an encryption key.
- q : the field size of the vector space used.

During the rest of this section, we use the operation $(a||b)$ to represent the concatenation of two given vectors, a and b.

A. Data Structures

- x_i^t: source message generated by node i at round t. The size of each message x_i^t is fixed and equal to L bits.
- s_i^t: encryption key created by node i which is used at round t.
- y_i^t: $s_i^t + x_i^t$, encrypted message of node i at round t. $+$ represents the addition operation over a given finite field.
- w_i^t: encryption coefficients vector for secret key s_i^t.[2]
- $p_\ell^t = (c_\ell^t||d_\ell^t)$: a SenseCode packet at round t. It is a concatenation of its coding (coefficients) vector (c_ℓ^t) and payload (d_ℓ^t) as defined in [1].

$$d_\ell^t \triangleq \sum_{j=1}^{N} c_\ell^t[j](w_j^t||y_j^t) = \left(\sum_{j=1}^{N} c_\ell^t[j] w_j^t\right)||\gamma_\ell^t,$$

  where $\gamma_\ell^t = \sum_{j=1}^{N} c_\ell^t[j] y_j^t$ and $c_\ell^t \in \mathbb{F}_q^N$.
- $\mathcal{P}_i^t$: set of packets p_ℓ^t overheard by node i at round t.
- Q_i: a FIFO (first-in first-out) bounded queue of size μ. Its elements are encryption keys (s_i^t) and their encryption coefficients (w_i^t) at node i for the next μ rounds.
- $\mathcal{Y}$: a list of all encrypted messages y_i^t for the last μ rounds. This list will be used for the reconstruction of encryption key s_i^t at the sink.

[2]The size of each w_i^t is $\mu \times N(\log_2(q))$ bits.

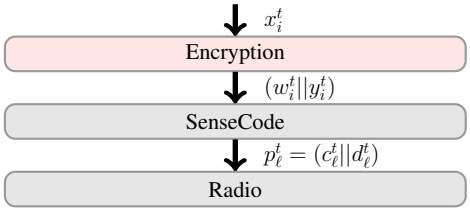

Fig. 1: Protocol stack

B. Algorithm

Message creation & encryption

1) Node i at round t generates a source message x_i^t to be communicated to a common collecting sink.
2) Encrypted message $y_i^t = s_i^t + x_i^t$ is prepared using the key s_i^t pulled from the top of queue Q_i at round t.
3) y_i^t and the encryption coefficients w_i^t of s_i^t are encapsulated into a SenseCode message $(w_i^t||y_i^t)$ which will be communicated to the sink.

Message collection

1) At each round t, node i communicates the encapsulated message $(w_i^t||y_i^t)$ to the sink using the SenseCode collection protocol (see Fig. 1).
2) In order to decode all the encapsulated messages, the sink waits until it receives at least N linearly independent combinations of them. At the end of round t, the sink tries to recover as much encapsulated message as possible from all the packets it receives and overhears.[3]
3) After recovering y_i^t and w_i^t, the sink runs the key reconstruction and message decryption phase to obtain the source message x_i^t.

Key reconstruction & message decryption at sink

1) The sink updates its list $\mathcal{Y}$ of encrypted messages with the new y_i^t.
2) Then, the sink needs to reconstruct the secret s_i^t using the encapsulated encryption coefficients vector w_i^t and the list of encrypted messages list $\mathcal{Y}$ as follows:

$$s_i^t = \sum_{k=1}^{\mu} \sum_{j=1}^{N} w_i^t[(k-1)N + j]\; y_j^{t-k} \tag{1}$$

3) After reconstructing the key s_i^t, node i can obtain the source message $x_i^t = y_i^t - s_i^t$.

Key construction at node i

1) (*Initialization*) Node i initializes its key queue Q_i with a predefined set of initial (possibly insecure and all zero) μ vectors.
2) At every round t, for each overheard packet $p_\ell^t \in \mathcal{P}_i^t$, node i updates all the elements in $\mathcal{Q}_i = \{(w_i^{t+1}||s_i^{t+1}), ..., (w_i^{t+\mu}||s_i^{t+\mu})\}$, $\forall k \in$ 1,2,..,μ:

$$w_i^{t+k}(k) = w_i^{t+k}(k) + \alpha_{\ell,i}^t c_\ell^t \tag{2}$$

$$s_i^{t+k} = s_i^{t+k} + \alpha_{\ell,i}^t \gamma_\ell^t, \tag{3}$$

where $\alpha_{\ell,i}^t \in \mathbb{F}_q$ is the random coefficient generated for p_ℓ^t during this update by node i and $w_i^{t+k}(k) = \left[w_i^{t+k}[(k-1)N+1] \; \ldots \; w_i^{t+k}[kN]\right]$.
3) When a node i pulls an encryption key s_i^t, $(w_i^t||s_i^t)$ from the top of Q_i, it pushes a new all zero element $(w_i^{t+\mu}||s_i^{t+\mu})$ to the bottom of the queue. This ensures the size of Q_i is always fixed and equals to μ.

A key construction example: Let $\mu = 4$ and suppose we are interested in the key construction procedure at node 1, starting from round $t+1$ up to $t+4$. Assume that node 1, collects only one packet over these rounds, i.e., p_l^{t+1}, p_l^{t+2}, p_l^{t+3} and p_l^{t+4}. Table I shows the contents of the queue $\mathcal{Q}_1$ at round $t+4$. The first row represents the head of the queue, the last row the tail, the first four columns the contents of vector w_1^{t+k} and the last column the encryption key s_1^{t+k}. For example, at round $t+5$, the key s_1^{t+5} is going to be removed from the top of the queue to be used for encrypting the source message x_1^{t+5} and a new key s_1^{t+9} will be initialized and added to the end of the queue. Once a key s_1^{t+k} is used for encryption, its corresponding coefficients vector w_1^{t+k} is attached to the encrypted message y_1^{t+k}. For simplicity, in this example we used $\alpha_{\ell,1}^t = 1, \forall t, \forall \ell$.

C. Cost analysis

Memory requirements: Each node i at any given round t has to keep the queue Q_i in its memory. The size of Q_i is fixed and $\mu \times (L + \mu(N \times \log_2(q)))$ bits. In other words, the queue has μ elements of size $(L + \mu(N \times \log_2(q)))$. An element of the queue $(w_i^t||s_i^t)$ has an encryption key (s_i^t) of L bits and encryption coefficients of size $\mu \times N \times \log_2(q)$ bits.

In order to regenerate the encryption keys at a given round t, the sink should keep a list of encrypted messages $\mathcal{Y}$ for all the N nodes in the last μ rounds. The size of this list in memory is $(\mu \times L \times N)$ bits.

Communication overhead: The size of a packet transmitted by our protocol is $((\mu+1) \times N \times \log_2(q) + L)$ bits, where L is the size of a plain message x_i^t. On the other side the size of a Sensecode packet (without any encryption of message) is $(N \times \log_2(q) + L)$ bits. In other words, encrypting messages at the nodes costs an extra $N \times \mu \times \log_2(q)$ bits per packet transmission.

Note that we are actually using the standard SenseCode protocol with larger messages. We can, therefore, claim that the number of packets transmitted in the network does not change compared to SenseCode without encryption (we found that larger packet sizes do not increase the packet error rates substantially). Moreover, we can significantly compress the coding vectors using techniques similar to [5].

Operational complexity overhead: For every overheard packet in $\mathcal{P}_i^t$ at round t, every node i updates (an addition operation over finite field $\mathbb{F}_q$) μ vectors in its key queue Q_i. Thus, the main computational overhead introduced (per node per round) by encrypting messages is:

$$|\mathcal{P}_i^t| \times \mu \text{ additions of 2 vectors of size } N \times \log_2(q) + L$$

[3]The payload d_l^t of the packets received at the sink is random linear combinations of encapsulated messages $(w_i^t||y_i^t)$ of each node $i \in 1...N$.

$\mathbf{t+k}$	$\mathbf{w_1^{t+k}}$				$\mathbf{s_1^{t+k}}$
5	c_l^{t+1}	c_l^{t+2}	c_l^{t+3}	c_l^{t+4}	$\gamma_l^{t+1}+\gamma_l^{t+2}+\gamma_l^{t+3}+\gamma_l^{t+4}$
6	c_l^{t+2}	c_l^{t+3}	c_l^{t+4}		$\gamma_l^{t+2}+\gamma_l^{t+3}+\gamma_l^{t+4}$
7	c_l^{t+3}	c_l^{t+4}			$\gamma_l^{t+3}+\gamma_l^{t+4}$
8	c_l^{t+4}				γ_l^{t+4}

TABLE I: Contents of queue $\mathcal{Q}_1$ at round $t+4$

On the sink side, the overhead for reconstructing all N keys s_i^t is $N^2 \times \mu$ multiplications of a vector (size L bits) and a scalar (see Equation 1). After reconstructing the keys, sink should compute the source messages x_i^t, which requires an extra N additions of 2 vectors of size L.

IV. Analysis

We start our analysis by observing that the payload of all packets sent by the protocol are linear combinations of source messages. We can therefore uniquely represent every packet as a vector that collects the coefficients used for linear combining. In this section we will call this vector the *coding vector* of the packet. We define the vector such that the coefficient used to linearly combine x_i^t is at position $Nt+i$ of the vector. The length of the vector is in principle unbounded, but in the following we will always be able to think about the vector as having a sufficient length, as it will be clear from the context.

Our analysis in the following assumes that the x_i^t's are statistically independent across sources and rounds and are uniformly distributed. This could be because we use distributed source coding or because the nature of the application data is so. If this is not the case, we will have a corresponding reduction in the expected secrecy as determined by the specific correlation patterns.

We denote as $\Pi_i^t \in \mathbb{F}_q^N$ the subspace spanned by the coding vectors of the packets node i collected at round t. We define Z_i^t the subspace spanned by the basis vectors of $\{\Pi_i^1, \ldots, \Pi_i^t\}$, Z_i^t represents all information that a node i can potentially collect up to round t and therefore use to recover messages sent by other nodes. We define W_i^μ the subspace spanned by the basis vectors of $\{\Pi_i^{t-\mu+1}, \ldots, \Pi_i^t\}$, this is the subspace from which the encryption keys s_i^t is chosen.

In the following we will exploit the fact that if the vector $\mathbf{e}_{Nt+i}$ is not in $Z_j^{t'}$, i.e., node j cannot reconstruct X_i^t from the linear combinations it has overheard then X_i^t is *weakly secure* from node j up to time t', i.e., $H(X_i^t|Z_j^{t'})$. To prove this it is sufficient to observe that $\mathbf{e}_{Nt+i}$ can be used to extend a base of Z_j^t and obtain a set of linearly independent vectors, then observe that the corresponding linear combinations of source messages are statistically independent, which implies that X_i^t is statistically independent from what node j knows and, therefore, secure. In this section we want to make two points:

- If the subspace overheard by the adversary doesn't contain the subspace used to create the secret key of round t on node i then with a non-zero probability the data sent by node i will be secure even if the adversary overhears all the packets sent in this round.
- By using linear combinations of past data to create keys our protocol doesn't compromise the security of the data sent in the previous rounds.

For the first point, we define a function g_i^t as follows:

$$g_i^t(\mu) \stackrel{\triangle}{=} \min_{j \neq i}[\dim(W_i^\mu) - \dim(W_i^\mu \cap Z_j^t)].$$

We will show that $g_i^t(\mu)$ determines the probability of picking a key that makes the transmission of node i in round t secure. The actual value of $g_i^t(\mu)$ depends on the network conditions. Here, we want to show that if it is bigger than 1 (i.e.the adversary does not collect everything that node i collected) then our protocol improves the security.

We first observe that our protocol chooses keys uniformly at random over W_i^μ. Indeed what our protocol does is to create a linear combination with random coefficients of the packets it has overheard. The following lemma shows that this linear combination is distributed uniformly.

Lemma 1. *Given a set of k vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}_q^N$ and k uniform and independent random variables $c_1, \ldots, c_k \in \mathbb{F}_q$, then $\sum_{i=1}^k c_i \cdot \mathbf{v}_i$ is uniformly distributed over $\langle \mathbf{v}_1, \ldots, \mathbf{v}_k \rangle$.*

Now we can use the following lemma to find what is the probability that a given key is not in the subspace overheard by the adversary as stated in Proposition 1:

Lemma 2. *Let two subspaces Π_i and Π_j of $\mathbb{F}_q^N$, for which $\Pi_i \not\subseteq \Pi_j$. Let also $\mathbf{v}$ a vector uniformly chosen in Π_i. Then*

$$\Pr[\mathbf{v} \in \Pi_j] = \frac{1}{q^\beta},$$

where $\beta = \min_j[\dim(\Pi_i) - \dim(\Pi_i \cap \Pi_j)]$, $\forall i \neq j$.

Proposition 1. *An encryption key s_i^t produced by our protocol by node i at round t is secure by an other node j with probability $\delta = 1 - \frac{1}{q^{g_i^t(\mu)}}$.*

Now, what remains to be proven is that by using a key that is not in Z_j^{t-1} we actually secure the data being transmitted at round t, against node j.

Proposition 2. *If $s_i^t \notin Z_j^{t-1}$ then $H(X_i^t|Z_j^t) = H(X_i^t) \forall j \neq i$.*

Proof. Node i selects an encryption key s_i^t at round t to encrypt its message x_i^t, i.e. $y_i^t = s_i^t + x_i^t$. For simplicity we will write the coding vectors of the packets as elements of $\mathbb{F}_q^{Nt}$. Now, let $\mathbf{v}_i = [\mathbf{w}_i \ \mathbf{e}_i]$, where $\mathbf{w}_i \in \mathbb{F}_q^{N(t-1)}$, represent the coding vector of the key s_i^t. Also, let $\mathbf{b}_1 \ldots \mathbf{b}_m$ a basis of Z_j^{t-1} and assume that node j overhears all the y_i^t for round t. Then $Z_j^t = \langle \mathbf{b}_1 \ldots \mathbf{b}_m, \mathbf{v}_1, \ldots, \mathbf{v}_N \rangle$, where vectors $\mathbf{b}_i$ have zeros in the last N entries.

We want to show that node j cannot decode data from node i, i.e., $e_{N\cdot(t-1)+i} \notin Z_j^t$, $\forall i \neq j$. Suppose that this is not the case, then we can find α_r and β_r such that:

$$\sum_{k=1}^{Nt} \left(\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[k] + \sum_{r=1}^{N} \beta_r \mathbf{v}_r[k] \right) \mathbf{e}_k = \mathbf{e}_{N(t-1)+i} \qquad (4)$$

The above equation holds if:

$$\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[N(t-1)+i] + \sum_{r=1}^{N} \beta_r \mathbf{v}_r[N(t-1)+i] = 1$$

and

$$\sum_{\substack{k=1 \\ k \neq N(t-1)+i}}^{Nt} \left(\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[k] + \sum_{r=1}^{N} \beta_r \mathbf{v}_r[k] \right) \mathbf{e}_k = 0.$$

The first equation implies that $\beta_r = 1$, for $r = i$, since $\mathbf{b}_r[N(t-1)+i] = 0$ for all r, and $\mathbf{v}_r[N(t-1)+i] = 1$ for $r = i$ and 0 otherwise. The second equation implies that all β_r, for $r \neq i$, must be zero, for we will not be able to cancel out the corresponding $\mathbf{e}_{N(t-1)+r}$ term. Now, Equation 4 can be rewritten as:

$$\sum_{\substack{k=1 \\ k \neq N(t-1)+i}}^{Nt} \left(\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[k] \mathbf{v}_i[k] \right) \mathbf{e}_k + \mathbf{e}_{N(t-1)+i} = \mathbf{e}_{N(t-1)+i} \Rightarrow$$

$$\sum_{r=1}^{m} \alpha_r \mathbf{b}_r + \sum_{\substack{k=1 \\ k \neq N(t-1)+i}}^{Nt} \mathbf{v}_i[k] \mathbf{e}_k = 0 \quad (5)$$

However, Equation 5 implies that $[\mathbf{w}_i \ \mathbf{0}_N]$ should have been in the span of $\mathbf{b}_i \dots \mathbf{b}_m$, which is a contradiction. Therefore node j cannot decode x_i^t and so $H(X_i^t|Z_j^t) = H(X_i^t)$. □

In the next proposition, we argue that using an encryption key at round t, produced by our protocol, does not compromise the security of the data sent in previous rounds, for which the adversary already had maximum uncertainty.

Proposition 3. *If $H(X_i^t|Z_j^t) = H(X_i^t)$ then $H(X_i^t|Z_j^{t+1}) = H(X_i^t)$, $\forall j \neq i$.*

Proof. Let $\mathbf{b}_1 \dots \mathbf{b}_m$ a basis of $\mathbb{F}_q^{Nt}$ that spans Z_j^t and $\mathbf{c}_1 \dots \mathbf{c}_m$ a basis of $\mathbb{F}_q^{Nt+1}$, where $\mathbf{c}_i = [\mathbf{b}_i \ 0]$.

By assumption it holds that $\mathbf{e}_l \notin \langle \mathbf{b}_1 \dots \mathbf{b}_m \rangle$, where $\mathbf{e}_l \in \mathbb{F}_q^{tN}$. Given that, it is straightforward to show that also $\mathbf{e}_l \notin \langle \mathbf{c}_1 \dots \mathbf{c}_m \rangle$, where $\mathbf{e}_l \in \mathbb{F}_q^{Nt+1}$, and vice versa.

Let the vector $\mathbf{v} = [\mathbf{w} \ 1]$, where $\mathbf{w} \in \mathbb{F}_q^{Nt}$, represent a coding vector of the key to be used in round $t+1$, and assume that $\mathbf{e}_l \in \langle \mathbf{c}_1 \dots \mathbf{c}_m, \mathbf{v} \rangle$. We can write:

$$\sum_{i=1}^{m} \sum_{r=1}^{Nt+1} \alpha_i \mathbf{c}_i[r] \mathbf{e}_r + \sum_{r=1}^{Nt+1} \gamma \mathbf{v}[r] \mathbf{e}_r = \mathbf{e}_l \Rightarrow \quad (6)$$

$$\sum_{i=1}^{m} \sum_{r=1}^{Nt} \alpha_i \mathbf{b}_i[r] \mathbf{e}_r + \sum_{r=1}^{Nt} \gamma \mathbf{w}[r] \mathbf{e}_r + \gamma \mathbf{e}_{Nt+1} = \mathbf{e}_l. \quad (7)$$

For $l \leq Nt$, for Equation 7 to hold, it should be $\gamma = 0$. What remains cannot hold, because it contradicts the assumption $\mathbf{e}_l \notin \langle \mathbf{b}_1 \dots \mathbf{b}_m \rangle$. For $l = Nt + 1$, Equation 7 does not hold because by construction, the basis $\mathbf{c}_1 \dots \mathbf{c}_m$ cannot span the vector $\mathbf{e}_{Nt+1}$. We conclude that $\mathbf{e}_l \notin \langle \mathbf{c}_1 \dots \mathbf{c}_m, \mathbf{v} \rangle \ \forall 1 \leq l \leq Nt + 1$, and therefore the uncertainty about message x_i^t after the observation of round $t + 1$ remains the same. □

V. Evaluation

A. Simulation Environment & parameters

SenseCode is implemented as a TinyOs module and tested with the TOSSIM simulator [4]. We implemented our encryption protocol in Java, and we evaluate its performance using the TOSSIM simulation results and the *nc-utils* toolbox [6]. We used a fixed field $\mathbb{F}_{2^4}$ for the network coding operations. Each TOSSIM simulation consists of 100 consecutive communication rounds. We consider the following topologies:

1) a 7×7 square grid ($N = 49$), with the sink located in the middle of the grid,
2) a 3×16 rectangular grid ($N = 48$), with the sink located in the middle of the short edge of the grid.

We configure each topology with two different inter-node distances: 20 m (sparse deployment) and 10 m (dense deployment), each of them yielding a different density of the network. For all the above network deployments, we test the performance of our scheme under the following scenarios:

- Every node is permanently present in the network.
- Node i can be in either connected or disconnected state. We set the *mean time between failures* (MTBF) to 400 sec, and the *mean time to repair* (MTTR) to 40 sec[4].

For the first scenario, we consider coded communication, for which nodes introduce a single codable packet per round (in [1] we describe as *codable* the packets that we allow the sensor nodes to linearly combine with other packets before forwarding them to the sink). For the second scenario, we consider redundancy $r = 2$, where each node injecting in network one uncodable and one codable packets per round. We assume that the uncodable packet is encrypted while the codable is not.

B. Simulation Results

We present average reliability results as a function of μ to observe the effect of using larger windows when creating secret keys. Note that $\mu = 0$ corresponds to the inherent security provided from the network coding operations in SenseCode. We emphasize that our reliability metric (see Section II-C) considers as secure only messages that are secured concurrently from all possible eavesdroppers, i.e., a communication is secure if none of the network nodes can recover what was sent.

Fig. 2 presents the average reliability ρ as a function of μ, for the square grid topology. We observe that our protocol increases the average number of secretly communicated messages by exploiting past communications. In other words, when we increase the parameter μ, we increase the time window of secrecy accumulation over time. As expected, for higher values of μ, we provide a better reliability.

For only one coded packet, we see that in a dense deployment the reliability is less, in comparison with the sparse, since the nodes overhear more common packets, but still increases fast with μ. For $r = 2$, the reliability is degraded due to fact

[4]Selected from [1], as a meaningful use-case for SenseCode.

sparse, 3×16, $r = 2$:	79.0243%
sparse, 3×16, only-coded:	64.4745%
sparse, 7×7, $r = 2$:	95.8544%
sparse, 7×7, only coded:	79.6517%
dense, 3×16, only-coded:	76.5313%
dense, 7×7, only coded:	90.8671%

TABLE II: Measured average delivery ratio per scheme.

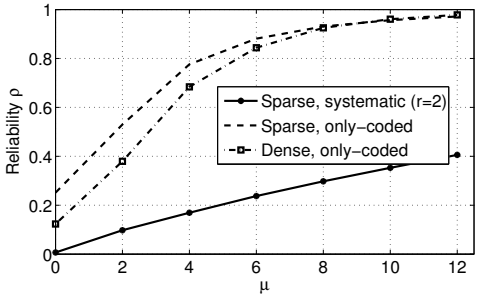

Fig. 2: Reliability - 7×7 Square Topology

that every node sends at least one uncoded SenseCode packet, facilitating in that way the task of the adversary. Nevertheless, our scheme achieves 20% improvement in reliability for small values of μ.

Fig. 3 presents similar results for the rectangular topology. In general, this topology provides less secrecy compared to a square grid, because the information flow (collection tree) is more concentrated to several points (nodes) in the network compared to a square grid. Even in this challenging topology, we provide up to 40% increase in terms of reliability for as low as $\mu = 4$.

Note that in both Fig. 2 and 3, our approach achieves higher reliability as compared to $\mu = 0$, the reliability achieved by SenseCode alone. For completeness, we provide in Table II the average percentage of y-packets decoded correctly to the sink per round with each scheme.

VI. Related work

Key distribution and establishment in wireless sensor networks has different characteristics, requirements and limitations than in traditional networks mainly due to the limited resources on sensor nodes. As a result, the widely accepted key management schemes for traditional networks have drawbacks for sensor network environments [7], [8], [9].

Closer to our work is the popular in the literature scheme of Eschenauer and Gligor (EG) [10] and follow up works (eg. [11]), where each node is provided with a set of keys (key ring) randomly selected from a common pool and use these as common randomness to create keys. Like EG-based schemes, our nodes also collect random keys, yet our randomness comes not from pre-distribution but from the randomness of the wireless channel conditions and topology; this enables us to easily refresh our keys at every round.

Our work can also be seen as offering weak security through network coding [3]; lightweight protocols have been developed

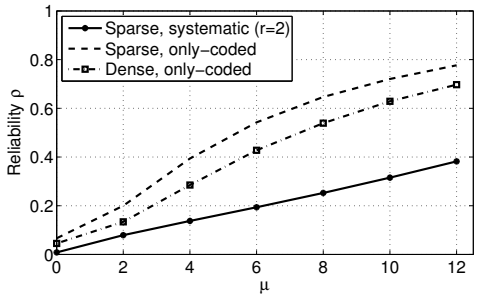

Fig. 3: Reliability - 3×16 Rectangular Topology

for weak security as in [12], [13] yet not applied to sensor networks. The work in [14] looks at security for network coded sensor networks with keys distributed by a mobile agent that visits the nodes.

VII. Conclusion

We presented the design and evaluation of a protocol that uses previous data as one-time pad to enhance the inherent weak security network coding offers. Our protocol operates cost sufficiently low to be well suited for sensor network applications.

References

[1] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, "Sensecode: Network coding for reliable sensor networks," *ACM Transactions on Sensor Networks*, vol. 9, no. 2, 2013.

[2] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 1–14.

[3] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," *NetCod, Apr*, 2005.

[4] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137.

[5] M. Jafari, L. Keller, C. Fragouli, and K. Argyraki, "Compressed network coding vectors," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE, 2009, pp. 109–113.

[6] "Nc-utils: Network coding utilities toolbox." [Online]. Available: http://code.google.com/p/ncutils/

[7] C.-Y. Chen and H.-C. Chao, "A survey of key distribution in wireless sensor networks," *Security and Communication Networks*, 2011.

[8] H. Chan, A. Perrig, and D. Song, "Key distribution techniques for sensor networks," *Wireless sensor networks*, pp. 277–303, 2004.

[9] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," *Rensselaer Polytechnic Institute, Troy, New York, Technical Report*, pp. 05–07, 2005.

[10] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 41–47.

[11] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 197–213.

[12] P. F. Oliveira and J. Barros, "A network coding approach to secret key distribution," *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 3, pp. 414–423, 2008.

[13] P. Zhang, Y. Jiang, C. Lin, Y. Fan, and X. Shen, "P-coding: secure network coding against eavesdropping attacks," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[14] Y. Wei, Z. Yu, and Y. Guan, "Efficient weakly-secure network coding schemes against wiretapping attacks," in *Network Coding (NetCod), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–6.