ICT-MobileSummit 2008 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2008
ISBN: 978-1-905824-08-3

# Resilient Network Coding for Wireless Sensor Networks

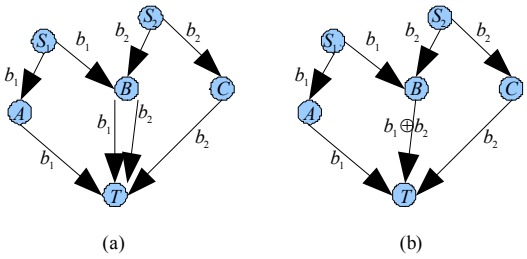
Yee Wei LAW, Marimuthu PALANISWAMI
*Department of Electrical and Electronic Engineering*
*The University of Melbourne, Parkville, VIC 3052, Australia*
*Email: yee.wei.law@gmail.com, swami@ee.unimelb.edu.au*

**Abstract:** Network coding is a novel multicast paradigm for maximising throughput. Resilient network coding is an extension that aims to provide resilience against eavesdropping and Byzantine packet modification by incorporating randomness and redundancy. Our review of resilient network coding schemes shows that, in the context of WSNs, only resilience against eavesdropping is achievable, when used in combination with a cryptographic scheme. We propose Relinc, a security architecture that is built on top of random key pre-distribution (RKP), and that uses *k*-generic random linear network coding as the coding scheme. Through simulations, we show that Relinc is more resilient than RKP by itself. For example, when 1 out of 100 nodes are compromised, between 70% and 90% of random topologies are still secure if Relinc is used, but only between 10% and 40% of random topologies are secure if RKP alone is used. The impact of this finding is that with proper redundancy and randomization, network coding can make a WSN that is already secured by RKP a few times more resilient to eavesdropping attacks.

**Keywords:** Wireless sensor networks, security, network coding

## 1. Introduction

Network coding is coding at a node in a network. Network coding came into the research scene when Ahlswede et al. discover in 2000 [1] that instead of just acting as information relays, the nodes in a network can increase the throughput of the network by acting as information en/decoders. The advantage of network coding is best illustrated using the example in Figure 1. In this example, $S_1$ and $S_2$ are source nodes that try to send $b_1$ and respectively $b_2$ to the sink node $T$. Notice that in order for $T$ to receive both $b_1$ and $b_2$, without network coding, $B$ has to transmit both $b_1$ and $b_2$, incurring a communication cost of 2 (Figure 1(a)); but with network coding, $B$ can just transmit $b_1 \oplus b_2$, incurring a communication cost of only 1 (Figure 1(b)). It can also be easily seen that since a node sends the same message over more than one link, network coding has the additional advantage of increasing the robustness of the network. For example, in Figure 1(b), as $S_1$ sends $S_2$ over more than one link, $T$ receives $b_1$ twice, once from $A$, and once from $B$ (albeit in the form of $b_1 \oplus b_2$). Additionally, while an eavesdropper tapping the link $BT$ in Figure 1(a) can learn about both $b_1$ and $b_2$, the same eavesdropper can learn neither $b_1$ nor $b_2$ in Figure 1(b). This is an interesting aspect of network coding. This is why not long after network coding was proposed, researchers started looking at how the technology could be used to enhance security, and the technique



of *resilient network coding* was born. Resilient network coding offers several advantages over several competing techniques:

- Compared to error-correcting codes, resilient network codes incorporate also throughput maximization but error-correcting codes do not.
- Compared to message authentication codes (MAC), resilient network codes are resilient to packet corruption but message authentication codes are not.

Our objective in this paper is to propose a security architecture for WSNs that is resilient against eavesdropping. We call the architecture *Relinc* and it is based on Eschenauer and Gligor's random key pre-distribution (RKP) [5] and Cai and Yeung's *k*-generic linear network code (LNC) [2]. Relinc is more resilient than an architecture that is based on either RKP or *k*-generic LNC alone, because an architecture based on RKP alone does not provide the kind of redundancy that is required for resilience, whereas an architecture based on *k*-generic LNC alone and totally devoid of any cryptographic mechanism, is virtually wide open in the threat model of WSNs, where the attackers are assumed to be able to tap all unsecured links. Relinc is practical, in fact, when describing the scheme, we specify actual values to be used for the system parameters in practice. Through simulations, we show that Relinc is more resilient than RKP by itself. For example, when 1 out of 100 nodes is compromised, between 70% and 90% of random topologies are still secure if Relinc is used, but only between 10% and 40% of random topologies are secure if RKP alone is used. A WSN implementing *k*-generic LNC alone is effectively "bare naked", so there is no comparison to speak of.

The rest of this paper is organised as follows. Section 2 introduces the basics of network coding, the help the readers' understanding of the proposed architecture. Section 3 reviews existing resilient network coding schemes, and identifies *k*-generic LNC as the most relevant scheme to WSNs. Section 4 describes the innards of Relinc in detail. Section 5 provides an in-depth simulation study of Relinc's resilience properties. Section 6 describes the business benefits of Relinc, giving coral reef monitoring as a practical example. Finally, Section 7 concludes. Table 1 contains a partial list of symbols and notations that are used in this paper – omitted symbols are either explained on the spot, or should be clear from the context.

*Table 1: Partial List of Symbols and Notations (Not Ordered According to Time of Appearance)*

| **Symbol** | **Semantics** | **Symbol** | **Semantics** |
|---|---|---|---|
| $G(V,E)$ | Graph with vertex set $V$ and edge set $E$ | $\leftarrow_R \bullet$ | Choose randomly from $\bullet$ |
| $\langle\bullet\rangle$ | Linear span of $\bullet$ | $q$ | Size of a Galois field (finite field) |
| $GF(q)$ | Galois field (finite field) of order $q$ | $GF(q)^n$ | An $n$-dimensional row vector space over a Galois field (finite field) of order $q$ |
| $(i, j)$ | The edge connecting vertices $i$ and $j$ | $x_i$ | The $i$th source process |
| $y(l)$ | A process on link $l$ | $z_i$ | The $i$th output process |
| $head(l)$ | The node of which $l$ is an incoming link | $tail(l)$ | The node of which $l$ is an outgoing link |
| $f_{j,l}$ | Local encoding kernel between link $i$ and $j$ | $f_j$ | Global encoding kernel of link $j$ |
| $E_{att}$ | A subset of the channels being eavesdropped on by an adversary | $\mathrm{E}_{att}^{*}$ | A collection of subsets of channels being eavesdropped on by an adversary; an adversary can access only one member of $\mathrm{E}_{att}^{*}$ at any given time |
| $k$ | $\lvert E_{att}\rvert$, i.e. no. of channels compromised at a time | $\sigma$ | Ratio of data symbols to hash/parity/redundant symbols in a packet |
| $b$ | Number of packets in a batch | $p$ | Number of pristine, un-tampered packets |
| $I_n$ | An $n \times n$ identity matrix | $N$ | Total number of nodes in a network |
| $\delta_i$ | Degree of a vertex $i$ | $\delta$ | Average degree of a graph |
| $P$ | Size of a key pool | $K$ | Size of a key ring |
| $h$ | Number of source processes | | |

## 2. Network Coding Basics

Following convention, we represent a network by a directed multigraph $G(V, E)$. A directed edge $\in E$ *re*presents a channel, defined as a noiseless means of transmission which transmits one data symbol per unit time. The *capacity* between two nodes measures the number of channels between the two nodes. Under the prevalent scalar algebraic coding framework [11], a source process[1] is represented by an element of $GF(q)$; $h$ source processes are represented by a $h$-dimensional row vector $\in GF(q)^h$. The process transmitted on a link $l \in E$ is a linear combination of the processes $x_i$ observed at the node $tail(l)$, and the processes on the incoming links of $tail(l)$:

$$y(l) = \sum_{i=1}^{s} a_{i,l} x_i + \sum_{j:head(j)=tail(l)} f_{j,l} y(j) \quad (1)$$

On a sink node, the $i$th output process is:

$$z_i = \sum_{l:head(l)=v} b_{i,l} y(l) \quad (2)$$

The scalars $f_{j,l}$ are called the *local encoding kernels*, from which we can derive the $h$-dimensional column vectors called *global encoding kernels*:

$$f_j = \sum_{i:head(i)=tail(j)} f_i f_{i,j} \quad (3)$$

Collecting the coefficients $a_{i,l}$, $f_{j,l}$ and $b_{i,l}$ into the matrices $A$, $F$ and $B$ respectively, we can write $(z_1 \ \ldots \ z_h) = (x_1 \ \ldots \ x_h)AFB^T$. The *transfer matrix* is defined as $M = AFB^T$. In general though, taking all paths into account,

$$M = A(I + F + F^2 + \ldots)B^T = A(I - F)^{-1}B^T \quad (4)$$

Unless the network topology is known *a priori* and static, the only scalable linear coding schemes are schemes that are distributed and randomized, i.e., schemes that choose the coefficients $a_{i,l}$, $f_{j,l}$ and $b_{i,l}$ locally, and randomly from $GF(q)$ [8]. Using random encoding, the output processes can be decoded at any given receiver with probability at least $1 - \varepsilon$, as long as $q \geq |E|/\varepsilon$ [9]. Compared to randomized routing, randomized coding offers a lower bound on the success probability (success = all source processes are correctly received) that is higher than the upper bound offered by randomized routing [6]. Moreover, randomized coding compresses linearly correlated sources effectively to the capacity of any cut that the code passes through. In practice, decoding is based on either Gaussian elimination [3] or minimum-entropy universal decoding [4].

## 3. Review of Resilient Network Coding Schemes

Usually, an attacker either wants to steal information or disrupt the network operation by injecting false data. In the threat model of WSNs, an attacker can compromise any node in the network and tap the corresponding compromised links simultaneously. Also, an attacker can arbitrarily corrupt any packet in the network (such an attacker is called a Byzantine attacker). Thus, a WSN should be (1) resilient to eavesdropping, i.e., even when an attacker taps a network at several links, it cannot recover any information about the source processes; and (2) able to detect Byzantine corruption, or preferably be resilient to Byzantine data corruption. Several schemes that implement some of these properties have been proposed. We classify them in Table 2.

Among these schemes, the proposals that address the problem of corruption resilience commonly assume an adversary can only inject its packets at a rate smaller than the receiver capacity [10][16]. This assumption fails to hold in WSNs where an adversary can easily use a transceiver with a larger data rate. Therefore we would not discuss these proposals any further.

*Table 2: Resilient Network Coding Schemes*

| | Resilience to eavesdropping | Corruption detection | Resilience to corruption |
|---|---|---|---|
| Jaggi et al. [10] | × | × | ✓ |
| Ngai and Yang [16] | ✓ | × | ✓ |
| Ho et al. [7] | × | ✓ | × |
| Tan and Médard [18] | ✓ | × | × |
| Cai and Leung [2] | ✓ | × | × |
| Lima et al. [15] | ✓ | × | × |

To detect packet corruption, Ho et al. [7] propose to use network coding to encode packets, and attach hashes calculated using simple polynomial functions to these packets. Under this construction, whenever $b - p$ packets are tampered (again, refer to Table 1 for the symbols' definition), a sink node will see at most $[(\sigma+1)/q]^p$ of its decoded packets are consistent, i.e., having matching data and hash symbols *when this should not happen*. However, if the adversary happens to be the *only* node supplying information to a sink on a *particular subset* of the original packets, then the adversary will be able to falsify information to the sink without being detected. In WSNs, an adversary can achieve this by simply jamming the sink's neighbours, and feeding the sink with false packets at the same time. MAC is a more reliable means for tamper detection.

Since Li et al. [14], network coding can also been formulated as an optimization problem. As is common in WSNs, we often associate more cost with links that are less stable and nodes that have lower energy, it will be more useful if a formulation takes transmission cost into account. The notion that security and transmission cost go hand-in-hand, lends itself to the formulation of an optimization problem. Tan and Médard [18] have this idea: establish an overall cost that is an increasing function of both transmission cost and vulnerability – either transmission cost or vulnerability would increase the overall cost of the network – and minimize (optimize) the overall cost. There are some limitations to this approach as applied to WSNs however. Firstly, the optimization problem has to be solved based on global information: the network topology. Secondly, the optimization problem has to be solved for every $E_{att} \in \mathrm{E}_{att}^{*}$. Thirdly, the probability that a link is compromised needs to be calculated, but this is hard to determine in practice.

Cai and Yeung [2] define a network code as "secure" if

- the code satisfies the rate constraints, i.e., does not overwhelm the links' rate limit;
- the code is uniquely decodable at every sink; and
- the code satisfies the security condition, which basically stipulates that an adversary's knowledge of the processes on $k$ links does not give it any information about the source processes.

Cai and Yeung's construction that satisfies these conditions is called $k$-generic linear network code[ii] (LNC). For completeness, we repeat the algorithms of $k$-generic LNC here:

1. Choose suitable positive integers $\omega$, $h$, $k$ with the relationship $\omega = h + k$. Let $\hat{x} = (x, r)$, where $x \in GF(q)^h$ is the original source vector, each element of which represents a source process; and $r \in GF(q)^k$ is a random vector.
2. Choose a suitable linear network code of dimension $\omega$.
3. Encode the vector $\hat{x}$ by the linear network code. A $k$-generic LNC is *admissible* for a sufficiently large field size $q$, if the following conditions are satisfied [2]:

Condition 1: There are at least $\omega$ pairwise disjoint paths from the sources to the sink.
Condition 2: Denote the $\omega$ or more pairwise disjoint paths from the sources to the sink by $E_{src \to snk}$. For every $E_{att} \in \mathrm{E}_{att}^{*}$, there are at most $k$ pairwise edge-disjoint paths from the sources to the channels in $E_{att} \cap E_{src \to snk}$.

A theorem due to Lima et al. [15] says that given a large enough field size $q$ and large enough number of source processes $h$, an intermediate node with less than $h$ incoming links is unlikely to be able to decode/intercept information on the way to a sink. The significance of this theorem is that by carefully choosing suitable values for $q$ and $h$ alone relative to a known network topology, we can make sure no intermediate node, once compromised, can decode the source processes.

We summarize this review with three points:

- There is no resilient network coding scheme for WSNs that provides meaningful resilience against false data injection.
- MAC is a better approach for detecting tampering than resilient network code in the context of WSNs.
- Assuming encryption and MAC are used in a network to encrypt and authenticate data, $k$-generic LNC can provide added value in the form of resilience against compromised nodes in the network. In other words, $k$-generic LNC can maintain the security of a network even when some of its secure links are compromised, provided Conditions 1 and 2 are satisfied.

In the next section, we will describe Relinc, our complete security architecture that leverages $k$-generic LNC.

## 4. Technology Description of Relinc

Relinc consist of three main parts:

1. Route set-up: After the nodes are deployed, every node will try to find *all* the shortest paths to the gateway/sink. If we represent the network as a tree (Figure 4), a node only needs to remember the IDs of its immediate parents on the shortest paths.
2. The cryptographic scheme: The scheme we use is RKP, which will be discussed in detail in Section 4.1.
3. The network coding scheme: The scheme we use is $k$-generic random linear network coding (RLNC), which we derive from $k$-generic LNC. More detail is given in Section 4.2.

### *4.1 Cryptographic Scheme*

RKP is a key management scheme. In this scheme, each sensor is pre-assigned a *key ring* of $K$ secret keys randomly drawn from a common pool of $P$ keys. The sensor nodes are then randomly deployed in the field. Every node would engage in a key discovery protocol with its neighbours. Two nodes can then establish a secure communication link when they find out they share at least a common pre-assigned key. The probability such a secure link exists between two nodes is given by:

$$p_s = 1 - \binom{P-K}{K} \Big/ \binom{P}{K} \tag{5}$$

A fundamental challenge is to choose $K$ and $P$ such that the network is *securely* connected. Our earlier results [12] allow us to adjust $K$ and $P$ such that not only the network is connected, but also connected at a certain mean connectivity.

### *4.2 Network Coding Scheme*

Our network code component is called $k$-generic RLNC and is adapted from $k$-generic LNC. In our adaptation, we replace linear network coding with random linear network coding, and we call the resultant adaptation $k$-generic random linear network code (RLNC). The reason for using randomized coding is that WSNs do not have pre-configured topologies, and even when they do, their topologies tend to be change over time (node deaths, node mobility, environmental effects are just some of the driving forces of network

dynamism). Therefore, computing a global, fixed network code is not practical. The remainder of this subsection is divided into two parts: the first part talks about estimating the field size $q$ in order to emulate linear network coding with random linear network coding, whereas the second part talks about the actual encoding and decoding scheme.

### *4.2.1 Estimation of the Field Size*

As we are now emulating a linear network code with a random linear network code, to guarantee any given receiver can decode its received packets with probability at least $1-\varepsilon$, we need $q \geq |E_{sec}|/\varepsilon$ [9], where $E_{sec} \subset E$ is the set of all secure links. In other words, we need to estimate $|E_{sec}|$. We start with the basic identity:

$$|E| = \frac{1}{2}\sum_{i=1}^{|V|}\delta_i \Rightarrow |E_{\text{sec}}| \approx |E| p_s = \frac{|V|\delta p_s}{2} \tag{6}$$

Substituting (5) into (6), we then get the minimum admissible value of $q$:

$$q \geq \frac{|V|\delta}{2\varepsilon}\left[1-\binom{P-K}{K}\Big/\binom{P}{K}\right] \tag{7}$$

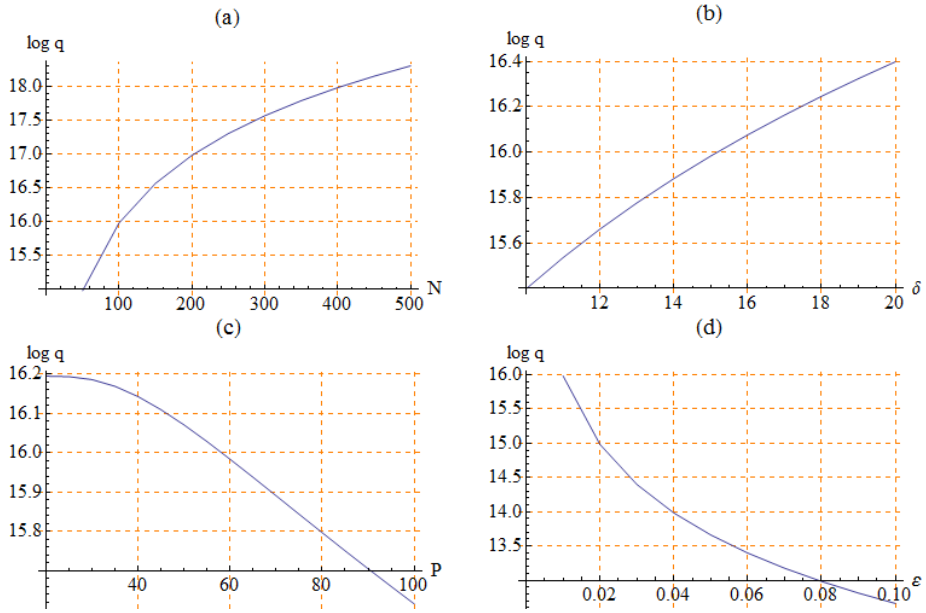



*Figure 2: (a) log q vs N with δ = 15, K = 10, P =60, ε = 0.01; (b) log q vs δ with N = 100, K = 10, P = 60, ε = 0.01; (c) log q vs P with N = 100, δ = 15, K = 10, ε = 0.01; (d) log q vs ε with N = 100, δ = 15, K = 10, P = 60*

We usually pick $q$ to be a power of 2 because our processors handle data in bits and bytes. Figure 2 shows that we need at least 16 bits for the field size. A field size of 16 bits is good because there are many 16-bit ultra low-power processors in the market, but if we use only 16 bits, we should limit the network size to be around 100 nodes, the density to be below 15 neighbours per node, and we should keep the pool size sufficiently high with respect to the key ring size, which is usually determined by the hardware constraints.

It is important to realize that (7) only gives us the minimum field size to guarantee decodability. We also need to consider suitable values for $K$ and $P$ to satisfy Condition 1. For this, we can refer to Law et al.'s results [12]. If a particular combination of $K$ and $P$ produces an expected *secure* connectivity of $\kappa$, the combination is good for supporting $h = \kappa - k$ source processes or preferably less. We have to ignore Condition 2 for the time being because we cannot predict which links will be compromised, but we will investigate this in Section 5.

We conclude this subsection with an example. The choice of parameters: $q = 2^{16}$, $N = 100$, $\delta = 15$, $K = 10$, $P = 60$, will support on average $7 - k$ source processes (7 is the

expected connectivity for these parameters), and ensure a *k*-generic RLNC to be decodable at any given receiver.

### *4.2.2 Encoding and Decoding Scheme*

Suppose the minimum of the max-flows from the sources to the sink is $\omega$, then we can support a maximum of $h = \omega - k$ source processes. The $i$th process produces a stream of source symbols denoted by $x_{i,1}, x_{i,2}, \ldots$ Every consecutive $n$ symbols are put into a packet, and $h$ packets – one from each source process – are grouped in a batch or generation [3], represented by a matrix $X$. $X$ is transformed into $\hat{X}$ (Equation (8)). The source node applies first random linear encoding and then encryption on $\hat{X}$, and sends out the encrypted encoded $\hat{X}$ appended with the associated MAC. On one of its incoming links, denoted $l$, a non-source node receives an encrypted blob. The node first authenticates and then decrypts the blob. The result looks like $Y(l)$ in (8).

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{h,1} & \cdots & x_{h,n} \end{bmatrix},\ R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{k,1} & \cdots & r_{k,n} \end{bmatrix},\ \hat{X} = \begin{bmatrix} X \\ R \end{bmatrix},\ \omega = h + r,\ \ Y(l) = F_l \left[ I_\omega \middle| \begin{matrix} \hat{X} \\ R \end{matrix} \right] \quad (8)$$

In (8), $F_l$ is the global encoding kernel associated with link $l$. Note that while in (3), the global encoding kernel was a column vector, it is now a $\omega \times \omega$ matrix because we are dealing with packets instead of individual symbols. The purpose of the identity matrix $I_\omega$ is so that the receiver can derive $F_l$ by performing Gaussian elimination on $Y(l)$, and thereby decoding $X$ [3].

## 5. Resilience Properties

To study the resilience properties of our security architecture, let us consider the following problem: given $t$ compromised nodes, what is the probability that a WSN with random topology, implementing Relinc, satisfies Conditions 1 and 2? As a reminder, satisfying Conditions 1 and 2 is equivalent to providing resilience to eavesdropping.

We start with an analytic approach and look specifically at the cases $t = 1$ and $t = 2$. When a node is compromised, all the $K$ keys in its key ring are compromised. Let us denote the compromised key ring by $C$. Any secure link established using any subset of $C$ is considered compromised as well. Let us pick a random secure link $l$ from the network, and denote the set of keys by which the secure link is established by $S$. The probability that $l$ is compromised is given by

$$\begin{aligned} \Pr[S \subseteq C] &= \sum_{i=1}^{K} \Pr[S \subseteq C \,||S| = i] \Pr[|S| = i] \\ &= \sum_{i=1}^{K} \frac{\binom{K}{i}}{\binom{P}{i}} \left[ \frac{\binom{P}{i}\binom{P-i}{K-i}\binom{P-K}{K-i}}{\binom{P}{K}\binom{P}{K}} \right] = \sum_{i=1}^{K} \binom{K}{i}\binom{P-i}{K-i}\binom{P-K}{K-i} \Big/ \binom{P}{K}^2 \end{aligned} \quad (9)$$

The ratio of the number of compromised links to the total number of secure links is also given by (9). Out of these compromised links, we need to consider the links that lie in the paths between the source and the sink. For this consideration, we resort to simulations.

Here is our simulation approach. For every topology we simulate, we fix the sink at coordinates (0,0) and the source at coordinates (0.5,0.5) in a unit square. Toroidal distance instead of Euclidean distance is used, to eliminate border effects. The rest of the $N - 2$ nodes are positioned at random within the unit square. The communication radius is chosen such that the average number of neighbours per node is $\delta$. For each value of $P$, 50 random topologies are generated, and for each topology, we find the number of paths from the

source to the sink, $\kappa$, and set $\omega = \kappa$, $h = \lceil \omega/2 \rceil$ or $\lceil 2\omega/3 \rceil$, and $k = \omega - h$. A topology is inadmissible when there are more than $k$ number of paths from the source to any of the compromised links. For comparison, we consider the base case where the network only implements RKP. In this base case, a topology is considered inadmissible when there is at least one path from the source to any of the compromised links. Our objective is to observe the ratios of inadmissible topologies to all generated topologies, for both the cases when Relinc is used, and when it is not. A comparison between the two cases tells us how much more Relinc is resilient to eavesdropping.

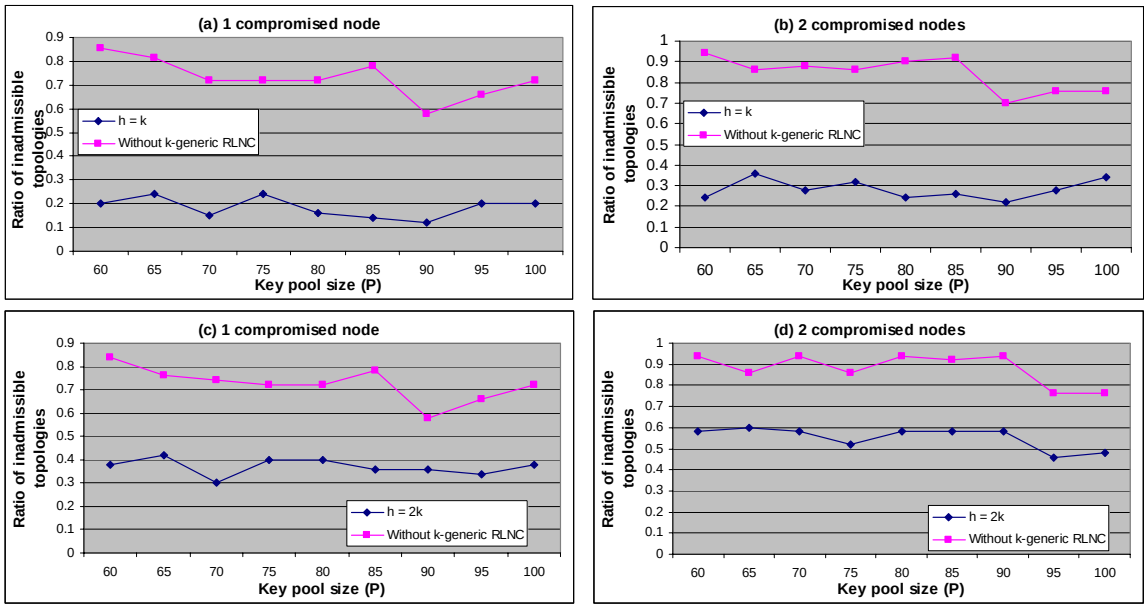



*Figure 3: Ratio of Inadmissible Topologies to Total Generated Topologies for (a) the Case $h = k$ with 1 Compromised Node; (b) the Case $h = k$ with 2 Compromised Nodes; (c) the Case $h = 2k$ with 1 Compromised Node; and (d) the Case $h = 2k$ with 2 compromised nodes*

Figure 3 shows our simulation results for $N = 100$, $\delta = 15$, $K = 10$ with $P$ and the number of compromised nodes as the variables. As can be seen, when $k$-generic RLNC is used, we get a few times more topologies that are resilient to eavesdropping. For example, when 1 out of 100 nodes is compromised, between 70% and 90% of random topologies are still secure if Relinc is used, but only between 10% and 40% of random topologies are secure if RKP alone is used. Consistent with intuition, Relinc fares better when we incorporate more redundancy in the packets ($h = k$ which has more redundancy vs. $h = 2k$ which has less redundancy). A pure RKP scheme also shows more variation in resilience with the key pool size $P$.

## 6. Business Benefits

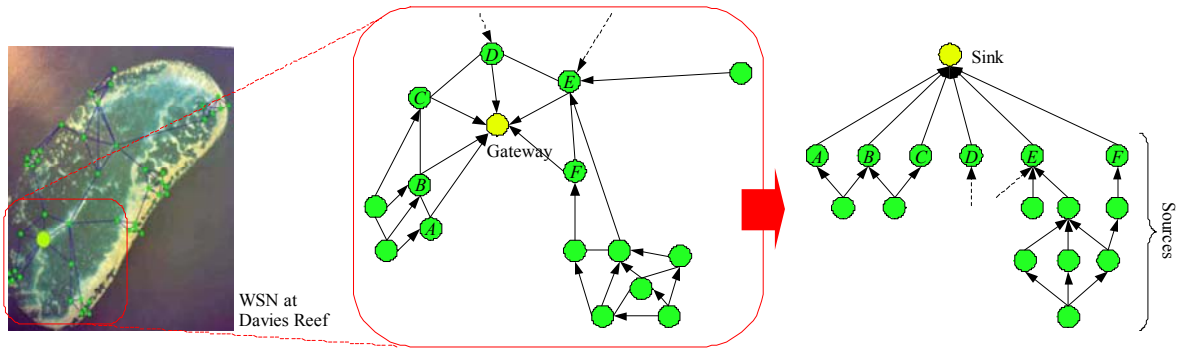



*Figure 4: Applying k-Generic Random Linear Network Coding to the Davies Reef WSN*

As part of the research activities of the Australian research network ISSNIP, several Australian universities are collaborating with the Australian Institute of Marine Science (AIMS) to set up WSNs at the GBR to collect data, like temperature, for the study of coral bleaching and other aspects of marine ecology. One such deployment is at Davies Reef (Figure 4). One primary requirement is to minimize data loss due to environmental effects such as ocean waves (which disrupt radio communications), sharks (which like to chew on the sensor strings) and tourists (who occasionally take away the sensor buoys as souvenirs). Another requirement is to protect the data, which are results of years of investment by AIMS. Network coding with built-in redundancy seems like a good candidate for the first requirement, with the added advantage of throughput maximization. By implementing Relinc, we get the added value of resilience against eavesdropping. As opposed to the conventional assumption, the nodes are carefully and not randomly deployed, allowing us to estimate in advance the resilience properties of the WSN (Figure 4).

## 7. Conclusions and Future Work

Among existing resilient network coding schemes, we identify *k*-generic LNC as the only meaningful scheme for WSNs. We have derived *k*-generic RLNC from *k*-generic LNC. Based on *k*-generic RLNC and a key management scheme called RKP, we have proposed a security architecture called Relinc. Relinc leverages RKP for encryption and authentication, and *k*-generic RLNC for resilience against eavesdropping. Relinc is more resilient than an architecture that is based on either RKP or *k*-generic LNC alone, because an architecture based on RKP alone does not provide the kind of redundancy that is required for resilience, whereas an architecture based on *k*-generic LNC alone is effectively "bare naked". Through simulations, we show that Relinc is indeed more resilient than RKP by itself. For example, when 1 out of 100 nodes is compromised, between 70% and 90% of random topologies are still secure if Relinc is used, but only between 10% and 40% of random topologies are secure if RKP alone is used. The caveat is that, to be used with a field size of $2^{16}$ (16 bits are a common word length for ultra low-power processors), Relinc is limited to moderate-scale WSNs that have around a hundred nodes, less than fifteen neighbours per node, a key ring size and a key pool size that give sufficient connectivity to support the intended number of source processes. The impact of this finding is that with proper redundancy and randomization, network coding can make a WSN that is already secured by RKP a few times more resilient to eavesdropping attacks.

We acknowledge the *potential* drawback of network coding, especially for nonlinearly correlated sources, is energy-efficiency since network coding relies on redundancy. One way to reduce energy usage of network coding is to compress the source processes using distributed source coding. It is however generally infeasible to apply network coding without destroying the structure in the 'source code' [17], barring a few special cases [17] and some heuristics that does not scale [19]. On the other hand, not all source processes are compressible, i.e., distributed source coding may not be applicable. We therefore advocate a hybrid approach: use resilient network coding when resilience warrants priority over energy-efficiency; use conventional routing when otherwise.

As future work, we would establish an analytical model to quantify Relinc's resilience properties. We will also implement Relinc on actual testbeds to identify potential pitfalls, and measure its performance and energy-efficiency.

## Acknowledgements

The authors are sponsored by the Australian Research Council Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), and the DEST International Science and Linkage Grant.

## References


[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, “Network Information Flow”, IEEE Transaction on Information Theory, 46(4): 1204–1216, 2000.

[2] N. Cai and R.W. Yeung, “Secure Network Coding”, in Proc. IEEE International Symposium on Information Theory, 2002.

[3] P.A. Chou, Y. Wu and K. Jain, “Practical network coding”, in Proc. 42st Annu. Allerton Conf. Communication, Control, and Computing, Monticello, IL, Oct. 2003.

[4] T. Cui, L. Chen, T. Ho, S.H. Low and L.L.H. Andrew, “Opportunistic Source Coding for Data Gathering in Wireless Sensor Networks”, in the 4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2007), IEEE Computer Society, 2007, ISBN 1-4244-1455-5.

[5] L. Eschenauer and V.D. Gligor, “A key-management scheme for distributed sensor networks”, in Proc. 9th ACM Conference on Computer and Communications Security, pp. 41—47, 2002.

[6] T. Ho, R. Koetter, M. Médard, D.R. Karger and M. Effros, “The benefits of coding over routing in a randomized setting”, in Proc. IEEE Symp. Information Theory, Yokohama, Japan, 2003.

[7] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros and D.R. Karger, “Byzantine modification detection in multicast networks using randomized network coding”, in Proc. IEEE International Symposium on Information Theory (ISIT), pages 144, IEEE, 2004.

[8] T. Ho, M. Médard, R. Koetter, D.R. Karger, M. Effros, J. Shi and B. Leong, “A random linear network coding approach to multicast”, IEEE Trans. Inform. Theory, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[9] S. Jaggi, S., P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain and L.M.G.M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," IEEE Transactions on Information Theory, 51(6):1973-1982, 2005.

[10] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi and M. Médard, “Resilient Network Coding in the Presence of Byzantine Adversaries”, in the 26th IEEE Conf. on Computer Communications (INFOCOM 2007), pages 616-624, IEEE, 2007.

[11] R. Koetter and M. Médard, “Beyond Routing: An Algebraic Approach to Network Coding”, in Proceedings of INFOCOM, New York, June 2002.

[12] Y.W. Law, L.-H. Yen, R. Di Pietro and M. Palaniswami, “Secure *k*-Connectivity Properties of Wireless Sensor Networks”, 4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2007): 3rd IEEE International Workshop on Wireless and Sensor Networks Security (WSNS’07), IEEE Computer Society, 2007.

[13] S.-Y.R. Li, N. Cai and R.W. Yeung, "On theory of linear network coding," Proc. Int. Symp. on Information Theory (ISIT), pp. 273-277, 2005.

[14] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear Network Coding”, IEEE Transactions on Information Theory, 49(2): 371–381, 2003.

[15] L. Lima, M. Médard and J. Barros, “Random linear network coding: a free cipher?”, eprint arXiv 0705.1789, 2007.

[16] C.K. Ngai and S. Yang, “Deterministic Secure Error-Correcting (SEC) Network Codes”, in Proc. Information Theory Workshop (ITW ’07), pages 96-101, IEEE, 2007.

[17] A. Ramamoorthy, K. Jain, P.A. Chou and M. Effros, “Separating distributed source coding from network coding”, IEEE/ACM Trans. Netw., 14(SI): 2785-2795, IEEE Press, 2006.

[18] J. Tan and M. Médard, “Secure Network Coding with a Cost Criterion”, in Proc. 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pages 1-6, IEEE, 2006.

[19] X. Zhang and S.B. Wicker, “Robustness vs. efficiency in sensor networks”, in Proc. 4th international symposium on Information processing in sensor networks (IPSN '05), pages 30-35, IEEE Press, 2005.


[i] The events under measurement are often represented by random processes, hence the term ‘processes’.

[ii] The original term “linear code multicast” was first proposed in [14], but was later superseded by “linear network code” [13].