Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 3473910, 13 pages
https://doi.org/10.1155/2018/3473910

Research Article

Enjoy the Benefit of Network Coding: Combat Pollution Attacks in 5G Multihop Networks

Jian Li[1], **Tongtong Li**[2], **Jian Ren**[2] **and Han-Chieh Chao**[3,4]

[1]*School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China*
[2]*Department of ECE, Michigan State University, East Lansing, MI 48824-1226, USA*
[3]*Department of Electrical Engineering, National Dong Hwa University, Taiwan*
[4]*Institute of Computer Science & Information Engineering and Department of Electronic Engineering, National Ilan University, Taiwan*

Correspondence should be addressed to Jian Li; lijian@bjtu.edu.cn

Guest Editor: Karl Andersson

Copyright © 2018 Jian Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the upcoming 5G era, many new types of networks will greatly expand the connectivity of the world such as vehicular ad hoc networks (VANETs), Internet of Things (IoT), and device-to-device communications (D2D). Network coding is a promising technology that can significantly improve the throughput and robustness of these emerging 5G multihop networks. However, network coding is generally very fragile to malicious attacks such as message content corruption and node compromise attacks. To take advantage of network coding in performance gain while refraining malicious network attacks is an interesting and challenging research issue. In this paper, we propose a new error-detection and error-correction (EDEC) scheme that can jointly detect and remove the malicious attacks based on the underlying error-control scheme for general multihop networks that can model the 5G multihop networks. The proposed scheme can increase the throughput for network with pollution attacks compared to existing error-detection based schemes. Then we propose a low-density parity check (LDPC) decoding based EDEC (LEDEC) scheme. Our theoretical analysis demonstrates that the LEDEC scheme can further increase the throughput for heavily polluted network environments. We also provide extensive performance evaluation and simulation results to validate the proposed schemes. This research ensures the expected performance gain for the application of network coding in the 5G network under malicious pollution attacks.

1. Introduction

Most of the newly emerging networks in the 5G communication network can be modeled using multihop networks, including vehicular ad hoc networks (VANETs), Internet of Things (IoT), and device-to-device communications (D2D). New computing frameworks in the 5G network such as fog computing [1] can also be categorized into multihop networks. The network throughput and robustness could be improved when network coding is utilized. Many researchers have proposed to adopt network coding in the upcoming 5G network, such as in [2–4]. The core notation of network coding is that it allows the participating nodes to encode incoming packets at intermediate network nodes in a way that when a sink receives the packets, it can recover the original message. Network coding provides a trade-off between maximum multicast flow rate and computational complexity.

Network coding was first introduced in the seminal paper by Ahlswede *et al.* [5]. Li *et al.* [6] formulated the multicast problem in network coding as the max-flow from the source to each receiving node. They proved that linear coding is sufficient to achieve the optimum. This work made network coding simpler and more practical. Koetter and Medard [7] have shown that linear codes are sufficient to achieve the multicast capacity by coding on a large enough field. Ho *et al.* [8] have shown that using of random linear network coding is a more practical way to design linear codes. Gkantsidis and Rodriguez [9] have applied the principles of random network coding to the context of peer-to-peer (P2P) content distribution and have shown that file downloading times

can be reduced. Esmaeilzadeh *et al*. [10] proposed to use feedback-free random linear network coding in broadcasting layered video streams over heterogeneous single-hop wireless networks. Wu *et al*. [2] designed an efficient data dissemination protocol for VANETs using network coding. Lei *et al*. [3] applied network coding in Named Data Networking (NDN) in the 5G IoT network and improved the content delivery efficiency. Chi *et al*. [4] proposed to jointly utilize D2D communication and network coding to achieve high communication reliability in the ultra-dense 5G cell deployment.

However, in the context of network coding, all participating nodes must encode the incoming packets according to a fixed coding algorithm. If a packet from an intermediate relay node is corrupted or being tampered, the entire communication may be disrupted.

The main purpose of this paper is to develop schemes that can combat network pollution and malicious attacks from the network nodes based on error-control coding in multihop networks. We propose a new scheme that combines error-detection and error-correcting (EDEC) to combat network pollution attacks. In our scheme, the original message symbol is first encoded using an error-control code before encoded by network coding and transmitted. The application of error-control code gives intermediate nodes the capability to detect possible errors or pollution of the message. Unlike the existing schemes, when an intermediate node detects an error, the packet will continue to be forwarded. As long as the errors are correctable, the sink nodes will be able to recover the corrupted packets and decode the original packet symbols. Then we further extend the EDEC scheme and propose LDPC based EDEC (LEDEC) scheme. In the LEDEC scheme we treat the packets as LDPC codewords at the sink nodes and use the belief propagation algorithm (BPA) to decode the LDPC code. It can guarantee a certain network throughput even for a heavily polluted network environment, while the throughput becomes very low for error-detection based schemes. Moreover, we mainly focus on the throughput impact brought by different strategies (discard versus keep) towards corrupted packets.

The major contributions of this paper are the following:

(i) We propose an EDEC scheme by combining a modified error-control code and network coding. The proposed EDEC scheme can increase the throughput for network environment with pollution attacks compared to existing error-detection based schemes.

(ii) We propose the LEDEC scheme by augmenting the EDEC scheme with the LDPC decoding. The proposed LEDEC can further improve the throughput even for network environment with heavy pollution.

(iii) We conduct extensive simulations to evaluate the performance of the proposed schemes and demonstrate the advantage of the proposed schemes.

The rest of this paper is organized as follows: Section 2 presents the related work. Preliminaries for network coding, error-control coding and the proposed modified error-control code are discussed in Section 3. Adversary model is also presented in Section 3. The EDEC scheme and performance analysis are described in Section 4. Section 5 presents the LDPC decoding and analysis of the LEDEC scheme. We conclude in Section 6.

2. Related Work

Existing work on pollution elimination can largely be divided into error-detection based schemes and error-correction based schemes. For error-detection based schemes, the errors are normally detected at the intermediate forward nodes. For error-correction based schemes, the errors are generally corrected at the sink node.

While the error-correction based schemes seem to be more appealing, the computational complexity for encoding and decoding is relatively high [11]. These schemes are generally designed based on knowledge of the network topology, which makes these schemes less flexible to the current networks. Jaggi *et al*. [12] introduced the first polynomial-time rate-optimal network codes to correct the errors brought by malicious nodes. They also gave the theoretical network capacity with the existence of malicious nodes. However, the communication overhead of the scheme is significant. In [13], the authors proposed a network error-correcting code that combines the nonlinear coding at the source node and linear coding at the intermediate nodes. The code can achieve higher transmission rate than linear network error-correcting codes with exponential encoding and decoding complexity.

The limitations of error-correction based schemes make error-detection attractive in some network scenarios. Krohn *et al*. [14] proposed to use homomorphic hash functions to guarantee correctness of the network flow. The main idea is that each intermediate node will check the correctness of the packets through homomorphic hash functions. If a packet fails the correctness check at an intermediate node, it will be discarded. This approach can reduce the communication overhead and can be used in random network coding. However, the computational complexity is still very high. When the network scale is large, computing too many hash values will also create high delay. Boneh *et al*. [15] provided cryptographic protection against pollution attacks by authenticating linear subspaces in network coding, which incurs less computation delay than [14]. Shang *et al*. [16] proposed new homomorphic signature schemes for generation-based network coding.

In [17–20], homomorphic message authentication codes (MAC) were designed to detect polluted packets. Yu *et al*. [17] applied multiple MACs to each packet in secure XOR network coding to filter out polluted packets. Agrawal *et al*. [18] also designed a homomorphic MAC to check the integrity of network coded data with less computational overhead. Li *et al*. [19] proposed to use a symmetric key based homomorphic MAC algorithms to detect corrupted packets. In [21], the authors analyzed and improved two homomorphic authentication schemes homomorphic subspace signature (HSS) [22] and key predistribution-based tag encoding (KEPTE) [23] for network coding.

To further reduce computational and communication overhead, Kehdi *et al*. [24] developed a simple error-detection

based *Null Key* scheme. Their idea is to partition the n-dimensional linear space over $GF(q^n)$ into two orthogonal subspaces of dimension k (symbol subspace) and $n-k$ (null key space). Comparing to the homomorphic hash function, the null key scheme is more efficient and has virtually no message delay. Newell *et al.* [25] improved the work of [24] by splitting the null key into one large constant portion and another small periodically updated portion, making it more suitable for network coding in wireless environments.

For all these error-detection based schemes, corrupted packets will be discarded. In packetized networks, a large message is divided into small packets. If a malicious node is able to continuously corrupt even a small fragment (packet) of the message, this fragment will be discarded. As a result, the whole message will be corrupted and the effective throughput will become lower and even close to zero in extreme situations. In our previous work [26], we proposed to keep the corrupted packets instead of dropping them; thus we could decode the corrupted packets in the intermediate forwarding nodes based on the subspace properties. In this paper we further generalize the idea and combine both error-detection and error-correction to guarantee a high throughput even for heavily polluted network environment.

It is worth noting here that the authors in [27] proposed using a cooperative nonparametric statistical framework (COPS) for misbehavior mitigation in network coding. Their framework does not need the knowledge of the data packet contents and provides another possible way to combat the network pollution attacks.

3. Preliminaries

In this section, we will first introduce basic concepts and notations of network coding. Then we will present the preliminary for error-control code and our proposed modified error-control code, which is the base of our proposed schemes. At last we will present our adversarial model adopted in this paper.

3.1. Network Coding. The main idea of network coding can be illustrated through the butterfly graph [6] in Figure 1. Assume the capacity of all the edges is C; the capacity of this network is $2C$ according to the max-flow min-cut theorem. Only by encoding the incoming bits x_1, x_2 at node 3, this network can achieve the maximum capacity.

In this paper, we adopt the notations of [7]. A network is equivalent to a directed graph $G = (V, E)$, where V represents the set of vertices corresponding to the network nodes and E represents all the directed edges between vertices corresponding to the communication link. The start vertex v of an edge e is called the tail of e and written as $v = tail(e)$, while the end vertex u of an edge e is called the head of of e and written as $u = head(e)$.

For a source node u, there is a set of symbols $\mathcal{X}(u) = (\mathbf{x}_1, \ldots, \mathbf{x}_l)$ to be sent, where $\mathbf{x}_i \in GF(2^n)$, $1 \le i \le l$. For a link e between relay nodes r_1 and r_2, written as $e = (r_1, r_2)$, the symbol $\mathbf{y}_e$ transmitted on it is the function of all the $\mathbf{y}_{e'}$ such that $head(e') = r_1$. And y_e can be written as

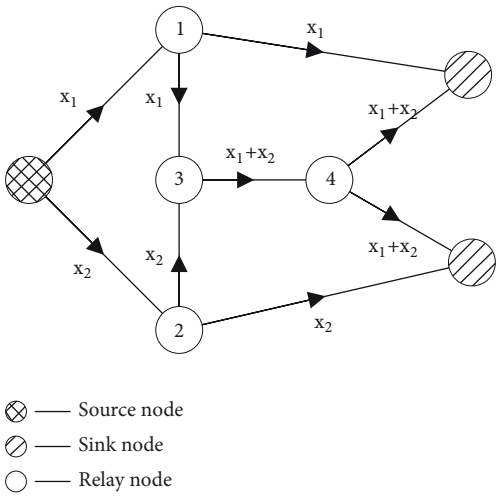


Figure 1: A simple example of network coding.

$$\mathbf{y}_e = \sum_{e':head(e')=r_1} \beta_{e',e} \cdot \mathbf{y}_{e'} = \sum_{i=1}^{l} \beta_{e,i}\mathbf{x}_i = \boldsymbol{\beta}_e \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_l \end{bmatrix}, \quad (1)$$

where $\beta_{e',e} \in GF(2)$ is the local network encoding coefficient, $\beta_{e,i} \in GF(2)$ is the global network encoding coefficient for symbol $\mathbf{y}_e$, and $\boldsymbol{\beta}_e = [\beta_{e,1}, \beta_{e,2}, \ldots, \beta_{e,l}]$ is the network encoding vector. For a sink node v, there is a set of incoming symbols $\mathbf{y}_1, \ldots, \mathbf{y}_m$ from e' where $tail(e') = v$ to be decoded.

For network coding to achieve the expected benefits, all the participating nodes in the network should be free of network pollution and malicious attacks. Suppose, under the linear network coding, the sink node receives m packet symbols $\mathbf{y}_1, \ldots, \mathbf{y}_m$. It decodes the original message symbols $\mathbf{x}_1, \ldots, \mathbf{x}_l$ by solving a set of linear equations:

$$\begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_m \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_l \end{bmatrix} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1l} \\ \beta_{21} & \cdots & \beta_{2l} \\ \vdots & \ddots & \vdots \\ \beta_{m1} & \cdots & \beta_{ml} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_l \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_m \end{bmatrix}. \quad (2)$$

For clarity purpose, we number the network coding vectors $\boldsymbol{\beta}_e$ received in the sink node from 1 to m. If all the relay nodes encode correctly and $m \ge l$, the sink node can decode all the message symbols successfully.

3.2. Error-Control Coding

3.2.1. Error-Detection. Suppose the original message symbols are in the k-dimensional linear space over $GF(2^k)$. After we encode the symbols using the $k \times n$ generating matrix $\mathbf{G}$ of an (n, k) block code, the encoded codewords will form a linear subspace over $GF(2^n)$ of dimension k. So there will be another $n-k$ dimensional subspace over the n dimensional space, which is orthogonal to the codewords subspace. Denote a

valid codeword by $\mathbf{c}$ and the bases for the $n-k$ dimensional subspace by $\mathbf{h}_1, \ldots, \mathbf{h}_{n-k}$; we have $< \mathbf{c}, \mathbf{h}_i >= 0,\ 1 \le i \le n-k$, where $< \cdot, \cdot >$ represents the inner product.

Let $(n-k) \times n$ matrix $\mathbf{H} = [\mathbf{h}_1^T, \ldots, \mathbf{h}_{n-k}^T]^T$; $\mathbf{H}$ forms a parity check matrix of the codewords and satisfies

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}. \tag{3}$$

Suppose $\mathbf{r} = \mathbf{c} + \mathbf{e}$ is a received codeword, where $\mathbf{e}$ is an n-tuple error generated by a malicious node. For the received word $\mathbf{r}$, we can get the *syndrome* of error pattern $\mathbf{e}$ as follows:

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = (\mathbf{c} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{c} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T. \tag{4}$$

For a received codeword $\mathbf{r}$, there are two cases: (i) $\mathbf{e}$ equals to a legitimate codeword generated by $\mathbf{G}$. In this case, although $\mathbf{r}$ contains error, the error is undetectable using conventional error-control coding techniques; (ii) $\mathbf{e}$ contains a nonzero projection to the orthogonal parity check subspace, which satisfies $\mathbf{r} \cdot \mathbf{H}^T \neq 0$. In this case, we can detect that the received word contains error.

In network coding, suppose $\mathbf{c}_j$'s are a set of valid codewords, $\mathbf{c} = \sum_j \beta_j \mathbf{c}_j$ is a linear combination of these codewords, where β_j is the network encoding coefficient. It can be easily verified that for $1 \le i \le n-k$,

$$\mathbf{c} \cdot \mathbf{h}_i^T = \sum_j \beta_j \mathbf{c}_j \cdot \mathbf{h}_i^T = 0. \tag{5}$$

Equation (5) ensures that we can still check the correctness of packet symbols using row vectors of $\mathbf{H}$ after they are encoded by network coding. Similar to [24], we call the row vectors $\mathbf{h}_i,\ 1 \le i \le n-k$ null keys.

3.2.2. Error-Correction. From (4), it is clear that $\mathbf{r}$ is a codeword if and only if $\mathbf{s} = 0$. The task of maximum likelihood decoding is to find the minimum weight error pattern $\mathbf{e}$ such that $\mathbf{r} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T$. In this case, the received $\mathbf{r}$ is corrected to $\mathbf{r} + \mathbf{e} = \mathbf{c}$.

In linear network coding, although the packet symbols are not the original ones sent from source nodes, we can still perform error-correction using (4). Suppose $\mathbf{r}_1, \ldots, \mathbf{r}_i$ are received codewords from i incoming edges; $\mathbf{e}$ is the error vector added to the network encoding $\mathbf{r} = \sum_j \beta_j \mathbf{r}_j$ and $\mathbf{r}' = \mathbf{r} + \mathbf{e}$. If the error is within the correction capability of the error-control code, the syndrome will still be $\mathbf{r}' \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T + \sum_j \beta_j \mathbf{r}_j \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T + \mathbf{0} = \mathbf{e} \cdot \mathbf{H}^T$. Thus we can correct the error using syndrome decoding.

The error-detection and error-correction capabilities are determined by the (n, k) code structure. Based on the pollution levels of the network, we can select the error-control codes accordingly base on the following proposition.

Proposition 1 (see [28]). *For an (n, k) block code with the minimum distance D, it can detect all the $D-1$ or less errors, or it can correct all the $\lfloor (D-1)/2 \rfloor$ or less errors.*

In our proposed schemes, the corrupted packets detected at the intermediate nodes will not be discarded. Both the intact and corrupted packets will be gathered by the sink nodes. The sink nodes can correct the corrupted packets and have a higher communication throughput than the error-detection based schemes.

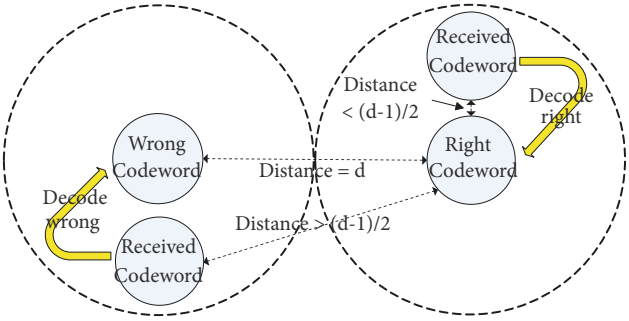


Figure 2: Limitations of error-control codes.

3.3. Modified Error-Control Code. The error-detection based schemes mainly focus on detecting the corrupt packets. As mentioned in Section 2, when a corrupt packet is detected, it will be discarded. So if an adversary continues to corrupt certain packets, these packets will be continuously dropped and the communication may never succeed. In this paper we try to utilize the corrupted packets to improve the throughput in these situations. To achieve this goal, we need to propose a modified error-control code.

3.3.1. Limitations of Conventional Error-Control Code. A linear error-correcting code encodes the original k bits message symbol $\mathbf{m}$ to an n bits codeword $\mathbf{c}$ using a $k \times n$ generating matrix $\mathbf{G}$. Suppose the minimum distance is D, according to the results in Section 3.2.2, the maximum number of errors we can correct is $\lfloor (D-1)/2 \rfloor$. If the number of errors is more than this amount, we may correct the corrupted codeword into a false one, as illustrated in Figure 2.

3.3.2. Proposed Error-Control Code That Can Detect Erroneous Decodings. The conventional error-control code may have undetected decoding errors. This is an inherent nature. No matter how low we set the code rate, these undetected errors may still exist. The decoding errors can only be detected using mechanisms other than a stand-alone error-correcting code.

Therefore, we propose to apply modified error-control code to both message symbols and network coding coefficients in (1). In this section, we will use the message symbol as an example. The original message symbol $\mathbf{m}$ is first mapped to a t bit value $\mathbf{h}$ using homomorphic hashes. Then $\mathbf{h}$ will be appended to $\mathbf{m}$ to form a new $k+t$ bits message symbol. By encoding this new message symbol we can get the final codeword. So the code becomes an $(n, k+t)$ code. By adding the extra bits, we can mitigate limitations of the conventional error-control code. Figure 3 illustrates the modified encoding scheme.

At the sink nodes, the decoded symbol is first split into two parts $\mathbf{m}'$ and $\mathbf{h}'$ after the decoding as shown in Figure 4. Then we calculate the mapping $\mathbf{h}''$ from $\mathbf{m}'$. If $\mathbf{h}''$

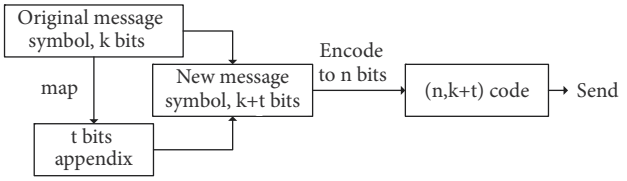


Figure 3: The encoding process of modified error-control code in EDEC scheme.

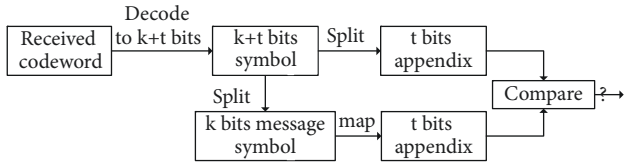


Figure 4: The decoding process of modified error-control code in EDEC scheme.

and $\mathbf{h}'$ are different, we can detect a decoding error. Our modification is equivalent to choose 2^k message symbols from 2^{k+t} symbols. Other message symbols in the 2^{k+t} symbol space are considered to be illegal. Because the decoding algorithm only guarantees that the decoded codeword is in the $k + t$ dimensional subspace, the decoded codeword belonging to the $2^{k+t} - 2^k$ symbol space implies that the decoding contains error.

Theorem 2. *Suppose a decoding error occurs, the wrong codeword will be any codeword in the 2^{k+t} symbol space. So the probability of detecting an erroneous decoding is*

$$p = \frac{2^{k+t} - 2^k}{2^{k+t}} = \frac{2^t - 1}{2^t} = 1 - \frac{1}{2^t}. \tag{6}$$

Through properly choosing the parameters k, t, we can detect erroneous decodings with little additional overhead.

3.4. Adversary Model. In network coding, a small error injected into an intermediate relay node may diffuse to many packets at the sink node, which will incur errors in the message symbols after the solving of (2). This can cause a significant waste of network resources and sometimes can even ruin the whole network communication. This kind of attack is called *pollution attack*.

In this paper, we assume that the malicious node can inject bogus packets into the network. If the succeeding relay nodes do not detect the bogus packets and produce network coded packets using the bogus packets, we call these packets *corrupted packets*. The malicious nodes try to forge legitimate packets that can pass the check in (5). In error-detection based schemes that use null keys to check the validity of packet symbols, the source node will randomly distribute s null keys in each intermediate relay node for packet checking. The malicious node can increase its successful attack probability by producing bogus symbols orthogonal to the space spanned by the s null keys it receives.

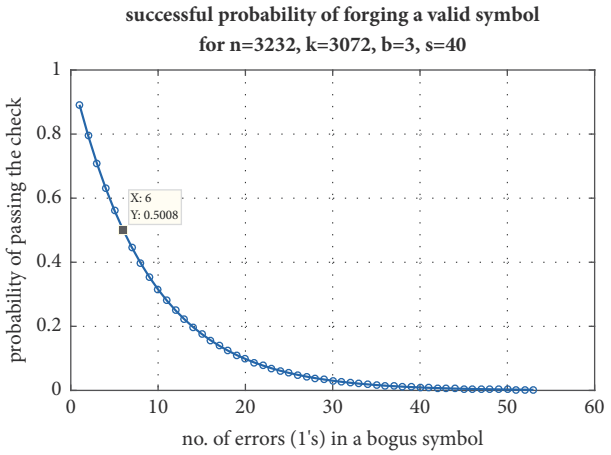


Figure 5: Probability for a forged symbol to pass the check.

When the size of the generating matrix is large, we will use sparse matrices $\mathbf{G}$ and $\mathbf{H}$ to reduce the encoding and decoding complexity. Suppose there are b 1's in each row and in each column on average in matrix $\mathbf{F} = \begin{bmatrix} \mathbf{G} \\ \mathbf{H} \end{bmatrix}$ for $b \ll n$, we have the following theorem.

Theorem 3. *The probability p_m that a malicious node successfully forges a valid symbol is*

$$p_m = \frac{\binom{k}{bw}}{\binom{n-s}{bw}}, \tag{7}$$

where $\binom{n}{k}$ is the number of k-combinations from a set of n elements and $w \ll n$ is the number of 1's in a bogus symbol.

Proof (sketch). When the number of errors, which are 1's in the bogus symbol, is small compared to n, for each error bit with index i there will be b distinct vectors with 1's at the index i in $\mathbf{F}$. The bogus symbol will be linear dependent with these b vectors with high probability. Thus a bogus symbol with w errors ($w \ll n$) will be linear dependent with at most bw vectors, which may include rows from both matrices $\mathbf{G}$ and $\mathbf{H}$.

When a malicious node forges a symbol which is only linear dependent with the row vectors in $\mathbf{G}$, it will successfully forge a valid symbol. Since the malicious node does not know the row vectors of the matrices $\mathbf{G}$ and $\mathbf{H}$ except for the null key vectors it has, these row vectors can be viewed as random to the malicious node. The successfully probability is the number of ways of choosing bw vectors in all the row vectors of $\mathbf{G}$ divided by that of choosing bw vectors in all the row vectors of $\mathbf{F}$ excluding the s null key vectors the malicious node has already known. □

For large n, k, the malicious node has to make w small to achieve a high probability p_m. In a typical parameter setting with $n = 3232$, $k = 3072$, $b = 3$, $s = 40$, each intermediate node will receive 40 null keys randomly selected from 160 null keys. The successful probability of forging a valid symbol is shown in Figure 5. From the figure we can see that to achieve successful probability $p_m = 0.5$, the malicious node

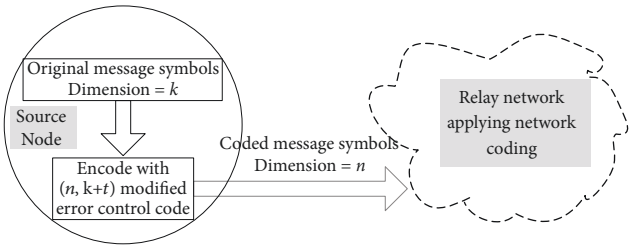


Figure 6: Apply error-control codes in linear network coding.

can only add 6 errors into a forged symbol, which are easily correctable by the (3232, 3072) code with minimum distance $D = 161$. *Thus for the malicious node, in order to produce bogus symbols that can pass the check, it will forge symbols with a small number of errors (1's) and orthogonal to the null key vectors it has.*

For the error-control code used to detect the bogus symbols, the errors within the bogus symbols are correctable. However, in the error-detection based schemes that do not perform error-correcting, the bogus symbols that pass the intermediate check will result in a failure of decoding the original message symbols at sink nodes.

4. Proposed EDEC Scheme and Evaluation

Similar to the null key scheme, our approach also utilizes the orthogonal space properties of the error-control code, but we use both the error-detection and error-correction techniques. The source node will encode the original message symbols using the modified error-control code prior to message transmission as shown in Figure 6. The properties of the error-control code keep unchanged during the linear network coding. When a corrupted packet is detected, instead of dropping it, we forward it to the sink nodes so that the packet can be used along with other packets to recover the original messages. However, the corrupted packet will not participate in network coding in the subsequent relay nodes once it is detected to be corrupted. The sink node can make use of the corrupted packets according to the decoding results of the modified error-control code in Section 3.3.

4.1. The Proposed EDEC Scheme. The proposed EDEC scheme is divided into two phases: initialization phase and transmission phase. The initialization phase is for null key and security parameter distribution while packet symbols are transmitted through network coding in the transmission phase.

4.1.1. Initialization Phase. In the initialization phase, the source node will first distribute s row vectors of the parity check matrix orthogonal to $\mathbf{G}$ in Algorithm 1 (null keys) to each of the relay nodes through a secret transmission protocol. Unlike normal linear network coding in which the network encoding vectors are attached to the start or the end of the packets, we propose to distribute the encoded encoding vectors to predetermined secret locations in the packets. The source node will send the location information to all the sink nodes during the initialization phase through the secure transmission protocol. This will prevent the malicious nodes from corrupting the encoding vector, which is essential for the data decoding. Moreover, the source node will also send the encoding matrix $\mathbf{G}_c$ for network encoding vectors and $\mathbf{G}$ for packet symbols to all the sink nodes. Once the initialization phase is done, the source nodes can multicast any number of packets to sink nodes. The overhead of the initialization phase is negligible.

4.1.2. Transmission Phase. In the transmission phase, the source node, relay nodes, and sink nodes will perform the proposed EDEC scheme according to Algorithms 1, 2, and 3.

In Algorithm 1, the source node will encode the network encoding vector $\boldsymbol{\beta}_i$ using the modified error-control code with a much longer appendix and a much lower code rate, compared to the encoding of packet symbols. This can improve the error resistance and detection probability for erroneous decodings to guarantee the correctness of the network encoding vectors used for data decoding. Since there is only one network encoding vector in each packet, the overhead brought by this higher security level is negligible.

Algorithm 2 presents the EDEC algorithm for relay nodes. Since the null key vectors are already distributed in the initialization phase, the relay nodes can check whether a packet is intact.

Algorithm 3 presents the EDEC algorithm for sink nodes. Since a sink node has already received the encoding matrix $\mathbf{G}_c$ and $\mathbf{G}$ in Algorithm 1 in the initialization phase, it can perform the error-control code decoding and detect the erroneous decodings. When collecting enough intact packets, it can derive the original data symbols through the decoding of network code.

4.2. Simulation Results of EDEC Scheme. In this section, we first present the simulation platform for EDEC scheme. Then we will compare the EDEC scheme and the error-detection based schemes, which are represented by the null key scheme in the simulation.

4.2.1. Simulation Platform. We simulate the EDEC scheme using ns-2. The scenario is set as a grid network with one source node, a number of relay nodes, and sink nodes. All the nodes are set as wireless nodes using wireless physical layer and 802.11 MAC protocol. The wireless channel is set to TwoRayGround. Once a node receives a packet, it will start the corresponding operations depending on its type (source, relay, malicious, and sink) and the packet content.

Figure 7 shows the topology of the simulated network. The source node is located at the lower left corner and 13 sink nodes lie at the upper right. The rest of the nodes are all intermediate nodes that can relay packets. In the simulation, we randomly choose a number of intermediate nodes as malicious nodes to perform pollution attacks. These nodes try to send out bogus packets that can pass the intermediate check as described in Section 3.4 to pollute the network. We can change the number of malicious nodes to evaluate performance of the algorithms under different network conditions. The percentage of malicious nodes is

(1) **for** packet i **do**
(2) //*Encode network encoding vector* $\boldsymbol{\beta}_i$ *in equation (1) using the modified error-control code (Figure 3)*
(3) $\mathbf{h}_c \longleftarrow \text{map}(\boldsymbol{\beta}_i)$
(4) $\mathbf{u}_c \longleftarrow (\boldsymbol{\beta}_i \mid \mathbf{h}_c)$
(5) Encoded network encoding vector $\longleftarrow \mathbf{u}_c \cdot \mathbf{G}_c$
(6) Distribute the encoded network encoding vector into predetermined locations of the packet
(7) **for** every symbol **m** of the packet **do**
(8) //*Encode* **m** *using the modified error-control code (Figure 3)*
(9) $\mathbf{h} \longleftarrow \text{map}(\mathbf{m})$
(10) $\mathbf{u} \longleftarrow (\mathbf{m} \mid \mathbf{h})$
(11) Encoded symbol $\longleftarrow \mathbf{u} \cdot \mathbf{G}$
(12) **end for**
(13) Send out the encoded symbols
(14) **end for**

Algorithm 1: EDEC algorithm for source nodes.

(1) **if** every symbol in the received packet is intact **then**
(2) **if** x (a predetermined number) packets received **then**
(3) Generate x randomly, linearly combined packets using the received packets (network coding)
(4) Send out the network encoded packets
(5) **end if**
(6) **else**
(7) Mark the packet as corrupted and send it out
(8) **end if**

Algorithm 2: EDEC algorithm for relay nodes.

(1) A packet is received
(2) Decode every symbol in the packet using the decoding algorithm for the modified error-control code (Figure 4)
(3) Reassemble the encoded network encoding vector from the predetermined secret locations and decode the network encoding vector
(4) **if** the network encoding vector and all symbols are decoded correctly **then**
(5) **if** the packet is independent **then**
(6) Save the packet
(7) **if** l (in equation (2)) independent packets are saved **then**
(8) Solve the network coding equations
(9) **end if**
(10) **end if**
(11) **end if**

Algorithm 3: EDEC algorithm for sink nodes.

about 20% in Figure 7. The rest of the intermediate nodes act as relay nodes. After receiving a packet, they will first conduct pollution detection. In the error-detection based schemes, if the packet is corrupted, it will be dropped. In our proposed EDEC scheme, we will continue to forward the corrupted packet to the sink nodes. However, the corrupted packet will not participate in the succeeding network coding. The nodes behaviors will be detailed in the next section.

Because the packets are transmitted through broadcasting, although the MAC protocol is IEEE 802.11, we will still have packets collisions that will eventually influence the simulation results. In this paper, we only focus on evaluating the proposed EDEC and LEDEC schemes. According to the transmission range of a single node, adjacent nodes are assigned different time slots (see Figure 8) to avoid packets collisions. This can guarantee that the transmission range belonging to the same time slot would not overlap. We divide the entire time duration into 9 time slots and the duration of each time slot is 100 ms. The nodes are allowed to send packets only if they are in their own slots. If not, they will have to wait

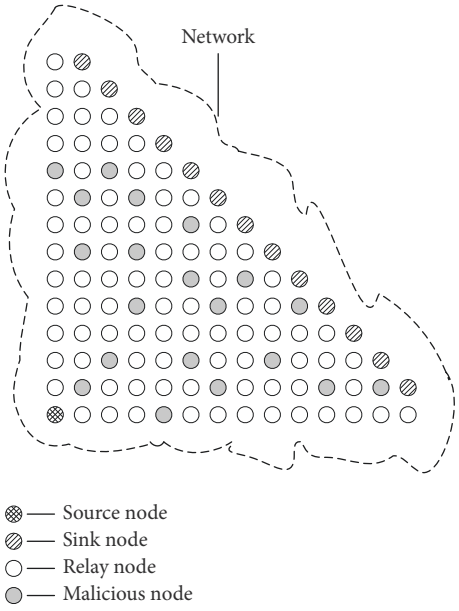


Figure 7: Simulation scenario.

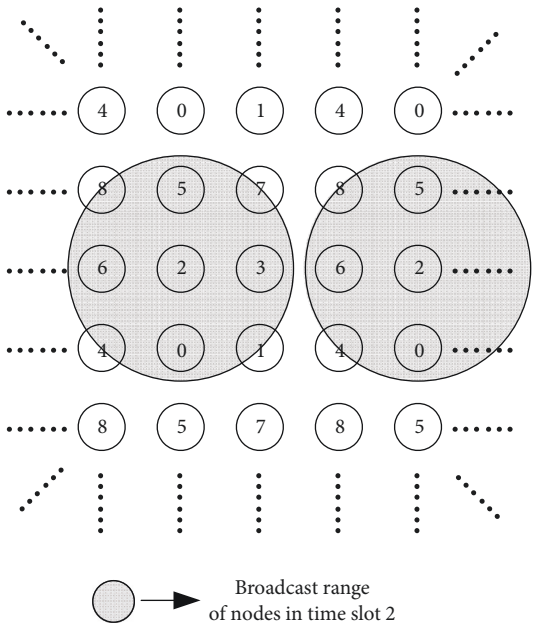


Figure 8: Illustration of the 9 time slots to avoid packets collisions.

until their next slots. In Figure 8 is an example in which the nodes belonging to time slot 2 can simultaneously transmit without packet collision.

4.2.2. Nodes Design. Four types of nodes are designed according to the algorithms described in Section 4.1 on the simulation platform.

(a) Source Node. In the simulation, the source node has $l = 32$ data packets to send. After initializing the network according to Section 4.1.1, the source node will encode the network encoding vector $\boldsymbol{\beta}_i$ for each packet using the modified error-control code presented in Figure 3 with $t = 32$. According to Theorem 2, the probability of detecting an erroneous decoding for the network encoding vector is $1 - 2^{-32}$, which means once the decoded network encoding vector passes the verification in Figure 4, we can view the network encoding vector as intact. Then the source node will distribute the encoded network encoding vector into predetermined locations in each packet. We fragment each packet embedding the encoded network encoding vector into symbols with the size 3056 bits and encode each packet symbol using the modified error-control code with $t = 16$. At last, the encoded packets are sent out into the network.

(b) Relay Nodes. Relay nodes will perform EDEC scheme according to Algorithm 2. Because the network is collision free and all the transmitted packets can be received, each packet only needs to be transmitted once. To utilize network coding efficiently while minimizing the transmission delay, each relay node will perform network encoding for every $x = 4$ valid packets it receiveds.

(c) Malicious Nodes. We assume that the malicious nodes will not perform network encoding. They only send out forged packets and conduct pollution attacks as described in Section 3.4.

(d) Sink Nodes. Sink nodes will decode both the modified error-control code and network code according to Algorithm 3.

4.2.3. Simulation Results. We conduct simulations under different percentages of malicious relay nodes. The throughput comparison between the EDEC scheme and the error-detection based schemes is shown in Figure 9. From the figure we can see that the EDEC scheme outperforms the error-detection based schemes in throughput: (i) When the percentage of malicious nodes is less than 10%, the two schemes have almost the same performance. (ii) With the increasing of malicious nodes, the performance of error-detection based schemes degrades significantly, while the throughput of the EDEC scheme remains almost unchanged. (iii) When the percentage of malicious nodes is larger than 40%, the throughput of the EDEC scheme begins to decrease because the cumulative number of errors in packet symbols becomes too large to correct by the modified error-control code. The uncorrectable packet symbols have the same degrading effect on the throughput as the packets discarded in the error-detection based schemes.

5. LDPC Decoding and LEDEC Scheme

In the EDEC scheme, only linearly independent packets participate in the network decoding at the sink nodes. Corrupted or linearly dependent packets will not be used, which is a waste of the resource. In this section, through utilizing these packets to recover more message symbols

(1) **while** There are check nodes connected to only one unknown symbol node **do**
(2) **for** Each of these check nodes **do**
(3) The unknown symbol node ⟵ xor (All of the other symbol nodes connected to the check node)
(4) **end for**
(5) **end while**
(6) **if** All the unknown symbol nodes are recovered **then**
(7) Decode successfully
(8) **end if**

Algorithm 4: BPA decoding algorithm for BEC.

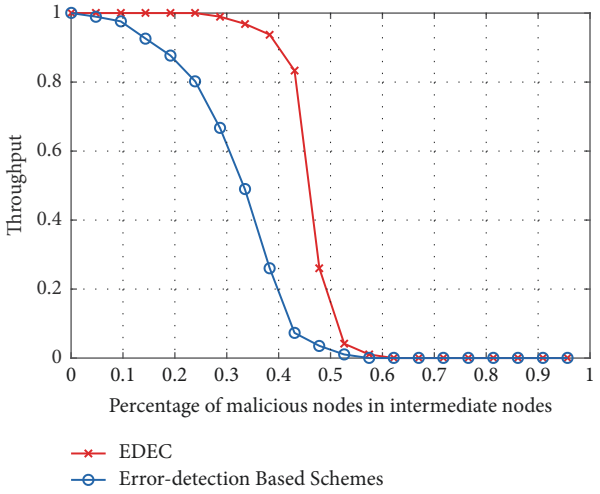


Figure 9: Throughput comparison between EDEC scheme and the error-detection schemes.

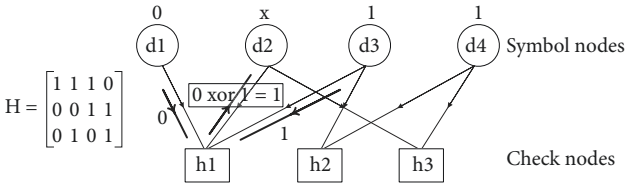


Figure 10: An illustrative example of parity check matrix and Tanner graph.

using low-density parity check (LDPC) decoding, we propose an LDPC decoding based EDEC (LEDEC) scheme.

5.1. LDPC Code. LDPC linear block code was first introduced by Gallager in 1962 [29]. One of the important characteristics of LDPC code is its sparse parity check matrix. By using iterative decoding, LDPC code can achieve error-correction performance close to Shannon bounds [30]. LDPC codes can be categorized as the regular LDPC code, of which the parity check matrix **H** has a fixed number of 1's per column and per row, and the irregular LDPC code, of which the parity check matrix may have different numbers of 1's in each column and each row. In this section, we will formulate the network coding to the irregular LDPC code.

5.2. Decoding of LDPC Code. The iterative decoding algorithm, known as belief propagation algorithm (BPA), is generally used to decode the LDPC code. Among all the channel models the BPA algorithm for the binary erasure channel (BEC) is the simplest, where only three numbers need to be considered: 0, 1, and x (erasure). The BPA can be described over the Tanner graph [31], which is a bipartite graph. In a Tanner graph, there are two types of nodes: the symbol nodes (corresponding to the received bits) and the check nodes (corresponding to the rows of the parity check matrix). An illustrative example of the parity check matrix and its Tanner graph is shown in Figure 10. In the parity check matrix, every row represents a parity check equation. The symbol nodes, which correspond to the bits equal to 1's in a row of the parity check matrix, are connected to the check node which corresponds to the same row. These nodes and edges in the Tanner graph express the parity check equation of that row. In Figure 10, node $h1$ represents the first row of the parity check matrix. The first, second, and third elements of the first row in parity check matrix are 1's, so symbol nodes $d1$, $d2$, and $d3$ are connected to $h1$ in the Tanner graph.

The decoding algorithm can be described through Algorithm 4 .

5.3. Relationship between Linear Network Code and LDPC Code. In linear network coding, packets are linearly combined at the intermediate nodes. The packets received at the sink nodes satisfy (2). In the network code decoding part of the EDEC algorithm, only independent valid packets are used. However, there is also helpful information in the linearly dependent packets or corrupted packets. If we can exploit and use these packets, the system performance can be further improved. Denote the received encoding vector as $\boldsymbol{\beta}_i = (\beta_{i,1}, \ldots, \beta_{i,l})$, where $1 \le i \le m$ and m is the number of received packets. Then the generation matrix of the block network code $\mathbf{G}_N$ can be defined as

$$\mathbf{G}_N = \left[\boldsymbol{\beta}_1^T \cdots \boldsymbol{\beta}_l^T, \boldsymbol{\beta}_{l+1}^T \cdots \boldsymbol{\beta}_m^T\right]^T. \tag{8}$$

As an example, suppose there is only one bit x_i in each original packet of the source node for $1 \le i \le l$. Define $\mathbf{x} = (x_1, \ldots, x_l)^T$. In this case, each received packet in the sink

nodes also contains only one bit y_j for $1 \le j \le m$. Denote all the m received packets as a vector $\mathbf{y} = (y_1, \ldots, y_m)^T$. We have the following encoding equation:

$$\mathbf{y} = \mathbf{G}_N \mathbf{x}. \tag{9}$$

Theorem 4. *The linear network code can be viewed as a rateless LDPC code and can be decoded using the BPA algorithm.*

Proof (sketch). Since we can have an uncertain number of $m > l$ received encoded bits in the encoding equation (9), we can view the network code as a rateless code. The generating matrix $\mathbf{G}_N$ can be rewritten as

$$\mathbf{G}_N = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} I_l \\ P_2 P_1^{-1} \end{bmatrix} P_1, \tag{10}$$

where matrix P_1 can be made as an $l \times l$ full rank matrix through row exchange after l independent packets are received, P_2 is an $(m-l) \times l$ matrix, and I_l is an $l \times l$ identity matrix. The corresponding parity check matrix $\mathbf{H}_N$ can be written as

$$\mathbf{H}_N = \begin{bmatrix} \left(P_2 P_1^{-1}\right)^T \\ I_{m-l} \end{bmatrix}, \tag{11}$$

where I_{m-l} is an $(m-l) \times (m-l)$ identity matrix. We can verify the correctness of $\mathbf{H}_N$ by verifying the follow equation:

$$\mathbf{H}_N^T \mathbf{G}_N = \begin{bmatrix} \left(P_2 P_1^{-1}\right)^T \\ I_{m-l} \end{bmatrix}^T \begin{bmatrix} I_l \\ P_2 P_1^{-1} \end{bmatrix} P_1 = \mathbf{0}. \tag{12}$$

After deriving the corresponding parity check matrix $\mathbf{H}_N$, we can decode the linear network code using the BPA algorithm. The linear network code can be viewed as a rateless LDPC code and has the property of error-control codes. □

Although linear network codes can be viewed as rateless LDPC codes, the BPA algorithm cannot be used to decode a network code if the network code is derived after the decoding of a conventional error-control code, because we cannot detect the incorrect decodings which should be viewed as erasures. For the modified error-control codes in the EDEC scheme, we can determine the erroneous decodings and mark the corresponding bits as erasures. Therefore, we can decode the linear network code using the BPA algorithm. Figure 11 illustrates this main idea of the LEDEC scheme.

5.4. Theoretical Analysis. Through the information in linearly dependent packets, the LEDEC scheme can get additional benefits from BPA decoding of the LDPC code. Consider the case in which the percentage of malicious nodes is high, most of the packets are corrupted by the malicious nodes. The error-detection based schemes cannot work because all of the packets are discarded. The EDEC scheme does not work well either because with high erasure probability P_e there will not be enough correctable packets to solve the network coding equations (2).

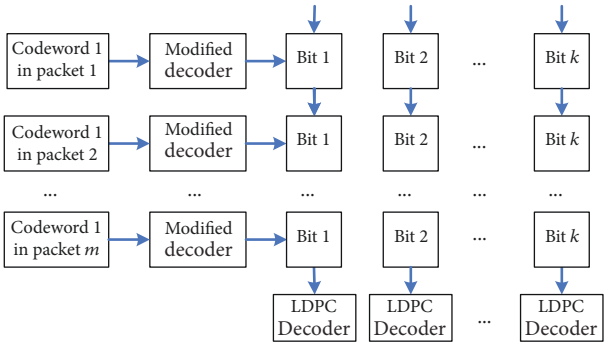


Figure 11: Main idea of the LEDEC scheme.

For the LEDEC scheme, let λ_d denote the probability that an edge from a check node is connected to a symbol node of degree d, and ρ_d denote the probability that an edge from a symbol node is connected to a check node of degree d in the Tanner graph of the corresponding LDPC code. The generating functions for an LDPC code is defined as: $\lambda(x) = \sum_d \lambda_d x^{d-1}$, $\rho(x) = \sum_d \rho_d x^{d-1}$. According to [32], the maximal fraction of erasures that a random LDPC code with given generating functions can correct is bounded by $P_{\max} = \min\{x/\lambda(1-\rho(1-x))\}$ $(0 < x < 1)$ with probability at least $1 - \mathcal{O}(n^{-3/4})$, where n is the length of the code. For the throughput of the LEDEC scheme, we have Theorem 5.

Theorem 5. *The throughput of the LEDEC scheme is*

$$F = \sum_{i=0}^{\lfloor N \cdot P_{\max} \rfloor} \binom{N}{i} P_e^i \left(1 - P_e\right)^{N-i}, \tag{13}$$

where $P_{\max} = \min\{x/\lambda(1-\rho(1-x))\}$ $(0 < x < 1)$, P_e *is the erasure probability,* N *is the number of packets a sink node received, and* $\lfloor \cdot \rfloor$ *is the floor function.*

Proof. Suppose a sink node receives N packets and the erasures in the packets are independent; the distribution of the number of erasures i in N received packet symbols is a binomial distribution with $\Pr(i) = \binom{N}{i} P_e^i (1-P_e)^{N-i}$, $0 \le i \le N$ as the probability mass function (PMF).

The proposed scheme can combat all erasures up to $N \cdot P_{\max}$ with probability at least $1 - \mathcal{O}(N^{-3/4})$, which is close to 1. Thus the throughput can be written as $F = \sum_{i=0}^{\lfloor N \cdot P_{\max} \rfloor} \Pr(i)$. □

5.5. Performance Analysis and Simulation. In this section, we provide simulation results of the LEDEC scheme on the simulation platform presented in Section 4.2. All the settings and parameters are the same as in Section 4.2.

5.5.1. Nodes Design. For the LEDEC scheme, the source node, relay nodes, and malicious nodes are the same as those in Section 4.2. The decoding process in the sink nodes is different, in which all packets received will be used. The BPA decoding will not start until the sink nodes collect all

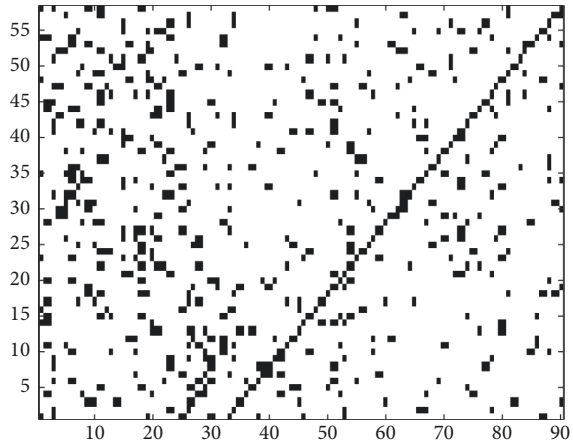


Figure 12: An example of the parity check matrix (transposed) in network coding.

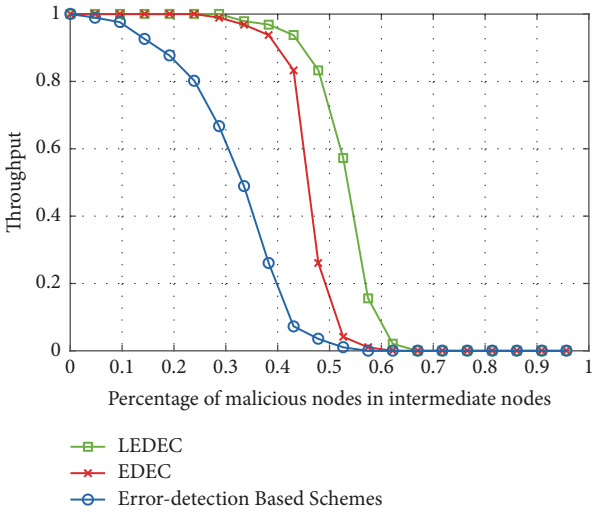


Figure 13: Performance evaluation of the LEDEC scheme.

the $l = 32$ independent packets. After receiving $l = 32$ independent packets, we can use the BPA algorithm to decode whenever a new packet arrives. However, there is a trade-off in determining when to start the BPA algorithm. When the algorithm is performed too frequently, the computational overhead will be high. On the other side, if we do not start the BPA decoding until we have collected a large number of packets, the communication delay will be high. To get a trade-off, the sink nodes will trigger the BPA decoding upon the receiving of every 10 new packets. This process will continue until all the message symbols have been successfully decoded.

5.5.2. Simulation Results. Same as in Section 4.2.3, the simulations in this section are carried out under different percentage of malicious relay nodes. One example of the parity check matrix (transposed) generated in the linear network coding is shown in Figure 12. In this example, the sink node receives 90 packets and decodes the linear network code using the BPA algorithm. In the figure, white squares represent 0 and black squares represent 1. The performance of the LEDEC scheme is shown in Figure 13.

Remark 6. When the percentage of the malicious nodes is less than 40%, the performance of the LEDEC scheme is slightly better than the EDEC scheme. This is because the sink nodes can successfully correct most of the errors in the corrupted packets and decode the original symbols using error-free packet symbols.

Remark 7. Because the sink nodes can recover extra information from the corrupted packets, when the percentage of malicious nodes is between 40% and 60%, the LEDEC scheme outperforms the EDEC and the error-detection based schemes.

6. Conclusion

In the paper, we focus on combating pollution attacks in multihop networks that utilize network coding, which can model most of the newly emerging networks in the 5G network. Our purpose is to maintain the network throughput even when the percentage of malicious nodes is large (≥30%). We first introduce an error-detection and error-correction (EDEC) scheme. By utilizing the information available in the corrupted packets, the network throughput can be increased with only a slight increase of the computational overhead compared to the error-detection based schemes. To further increase the throughput for the network environment with heavy pollution, we introduce LEDEC scheme that enables channel information be exploited and belief propagation algorithm (BPA) be used for the packet symbol recovery. This scheme can guarantee the throughput under the heavy pollution (percentage of malicious nodes is larger than 40%). We formulate the throughput of the LEDEC scheme through theoretical analysis and conduct comprehensive simulations to evaluate the performance. Our extensive simulation results show that the LEDEC scheme achieves better throughput than the EDEC scheme.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

Jian Li's first part of the work was done when he was with the Department of ECE, Michigan State University, East Lansing, MI 48824-1226.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities KWJB17052536

and in part by the National Natural Science Foundation of China under Grant 61701019. This work is based on the work of the PhD dissertation of Jian Li [33].

References

[1] W. Zhang, Z. Zhang, and H. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[2] C. Wu, X. Chen, Y. Ji, S. Ohzahata, and T. Kato, "Efficient broadcasting in VANETs using dynamic backbone and network coding," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6057–6071, 2015.

[3] K. Lei, S. Zhong, F. Zhu, K. Xu, and H. Zhang, "An ndn iot content distribution model with network coding enhanced forwarding strategy for 5g," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 2725–2735, 2018.

[4] K. Chi, L. Huang, Y. Li, Y.-H. Zhu, X.-Z. Tian, and M. Xia, "Efficient and reliable multicast using device-to-device communication and network coding for a 5G network," *IEEE Network*, vol. 31, no. 4, pp. 78–84, 2017.

[5] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[6] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[7] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[8] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Proceedings of the International Symposium on Information Theory ISIT' 04*, p. 44, July 2004.

[9] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *Proceedings of the IEEE INFOCOM '06*, pp. 1–13, April 2006.

[10] M. Esmaeilzadeh, P. Sadeghi, and N. Aboutorab, "Random linear network coding for wireless layered video broadcast: general design methods for adaptive feedback-free transmission," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 790–805, 2017.

[11] N. Cai and R. W. Yeung, "Network coding and error correction," in *Proceedings of the IEEE Information Theory Workshop (ITW '02)*, pp. 119–122, 2002.

[12] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of Byzantine adversaries," in *Proceedings of the IEEE INFOCOM 2007: 26th IEEE International Conference on Computer Communications*, pp. 616–624, USA, May 2007.

[13] W. Guo, D. He, and N. Cai, "On capacity of network error correction coding with random errors," *IEEE Communications Letters*, vol. 22, no. 4, pp. 696–699, 2018.

[14] M. N. Krohn, M. J. Freedman, and D. Mazières, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pp. 226–239, May 2004.

[15] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: signature schemes for network coding," in *Public Key Cryptography-PKC 2009*, vol. 5443 of *Lecture Notes in Comput. Sci.*, pp. 68–87, Springer, Berlin, Germany, 2009.

[16] T. Shang, T. Peng, Q. Lei, and J. Liu, "Homomorphic Signature for Generation-based Network Coding," in *Proceedings of the 2016 IEEE International Conference on Smart Cloud, SmartCloud 2016*, pp. 269–273, USA, November 2016.

[17] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing XOR network coding against pollution attacks," in *Proceedings of the 28th Conference on Computer Communications (INFOCOM '09)*, pp. 406–414, April 2009.

[18] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2–5, 2009. Proceedings*, vol. 5536 of *Lecture Notes in Computer Science*, pp. 292–305, Springer, Berlin, Germany, 2009.

[19] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, "RIPPLE authentication for network coding," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, March 2010.

[20] A. Le and A. Markopoulou, "Cooperative defense against pollution attacks in network coding using SpaceMac," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 2, pp. 442–449, 2012.

[21] C. Cheng, J. Lee, T. Jiang, and T. Takagi, "Security analysis and improvements on two homomorphic authentication schemes for network coding," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 993–1002, 2016.

[22] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. Shenz, "Padding for orthogonality: Efficient subspace authentication for network coding," in *Proceedings of the IEEE INFOCOM 2011*, pp. 1026–1034, China, April 2011.

[23] W. Xiaohu, X. Yinlong, Y. Chau, and X. Liping, "A tag encoding scheme against pollution attack to linear network coding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 33–42, 2014.

[24] E. Kehdi and B. Li, "Null keys: limiting malicious attacks via null space properties of network coding," in *Proceedings of the 28th Conference on Computer Communications, IEEE INFOCOM '09*, pp. 1224–1232, April 2009.

[25] A. Newell and C. Nita-Rotaru, "Split Null Keys: A null space based defense for pollution attacks in wireless network coding," in *Proceedings of the 2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2012*, pp. 479–487, Republic of Korea, June 2012.

[26] W. Qiao, J. Li, and J. Ren, "An efficient error-detection and error-correction (edec) scheme for network coding," in *Proceedings of the IEEE Globecom '11*, pp. 1–5, December 2011.

[27] A. Antonopoulos and C. Verikoukis, "COPS: cooperative statistical misbehavior mitigation in network-coding-aided wireless networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1436–1446, 2017.

[28] S. Lin and D. J. Costello, *Error Control Coding*, Prentice Hall, 2nd edition, June 2004.

[29] R. G. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 8, pp. 21–28, 1962.

[30] C. E. Shannon, "A mathematical theory of communication," *Bell Labs Technical Journal*, vol. 27, pp. 379–423, 1948.

[31] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[32] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[33] J. Li, *Capacity assurance in hostile networks [Ph.D. dissertation]*, Michigan State University, 2015.


Journal of Engineering
The Scientific World Journal
International Journal of Rotating Machinery
Journal of Sensors
Advances in Multimedia
Advances in Civil Engineering
Hindawi
Submit your manuscripts at
www.hindawi.com
Journal of Control Science and Engineering
Journal of Robotics
Journal of Electrical and Computer Engineering
Advances in OptoElectronics
VLSI Design
International Journal of Navigation and Observation
Modelling & Simulation in Engineering
International Journal of Aerospace Engineering
International Journal of Chemical Engineering
International Journal of Antennas and Propagation
Active and Passive Electronic Components
Shock and Vibration
Advances in Acoustics and Vibration