Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 216524, 9 pages
doi:10.1155/2010/216524

Research Article

Secure Network Coding against Wiretapping and Byzantine Attacks

Qin Guo,[1,2,3] Mingxing Luo,[1,2,3] Lixiang Li,[1,2,3] and Yixian Yang[1,2,3]

[1] *Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, P. O. Box 145, Beijing 100876, China*
[2] *Key Laboratory of Network and Information Attack and Defence Technology of MOE, Beijing University of Posts and Telecommunications, P. O. Box 145, Beijing 100876, China*
[3] *National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, P. O. Box 145, Beijing 100876, China*

Correspondence should be addressed to Qin Guo, galaxyi@163.com

Received 12 August 2009; Revised 13 December 2009; Accepted 1 April 2010

Academic Editor: Nicholas Kolokotronis

In wireless networks, an attacker can tune a receiver and tap the communication between two nodes. Whether or not some meaningful information is obtained by tapping a wireless connection depends on the transmission scheme. In this paper, we design some secure network coding by combining information-theoretic approaches with cryptographic approaches. It ensures that the wiretapper cannot get any meaningful information no matter how many channels are wiretapped. In addition, if each source packet is augmented with a hash symbol which is computed from a simple nonlinear polynomial function of the data symbols, then the probability of detecting the modification is very high.

1. Introduction

Network coding is a packet-level coding technique that generalizes the classical routing paradigm [1]. Based on linear superposition of incoming packets at intermediate nodes, linear network coding achieves multicast capacity in single-source wired networks [2]. Recently, wireless network coding has gained much attention as one skill to enhance the overall throughput in a wireless multihop network that supports multiple communication flows [1–4]. Like in wired networks, the basic idea is also that a relay node can combine several incoming packets. The wireless communication medium has inherent particularities, such as the broadcast nature, high error rates, and unpredictable signal strength, which create some opportunities for attackers. Thus, secure network coding [5–8] is a hot topic in wireless networks.

In [9], Cai and Yeung proposed a model which incorporates network coding and information security. This is the first time that network coding was used for secure transmission. Later, Jain [5] widened the sufficient condition in [9] and generalized these results to wireless networks. In general, secure network coding is designed against two kinds of attacks: wiretapping and Byzantine modification. And we call them type I and type II secure network coding, respectively.

Type I Secure Network Coding. The wiretapping attack means that some adversary can wiretap some communication signals with the purposes of curiosity or recovering the messages. In traditional transmission, packets are generally encrypted against wiretapping. However, Cai and Yeung [9] found that without cryptographic approaches one can securely transmit the message by using network coding. With the similar model for wireless networks, Jain [5] explored widening the sufficient condition in [9] so that it becomes necessary, too. Under the transmission rate of one unit, the sender can send a message to the receiver without leaking any information to a wiretapper. In [10], Feldman et al. showed that if a small amount of overall capacity is given up, then a random code achieves security by using a much smaller base field than that in [9]. Furthermore, they pointed out that a large field size may sometimes be

required to achieve security without giving up any capacity. In [11], Bhattad and Narayanan generalized the model in [9] and gave a new information theoretic model for security which accommodates a lot more practical requirements on security.

In general, this previous secure network coding by information-theoretic approaches has to restrict the eavesdropping set or give up some capacities. To address this problem, in this paper, we present a new secure network coding scheme by combining information-theoretic and cryptographic approaches. In our scheme, we do not restrict the eavesdropping set, that is, the wiretapper can eavesdrop any communication signals. Moreover, we do not give up any capacity. Based on these superiorities, our scheme is more suitable for wireless multicast compared with the previous schemes [5, 9, 10].

Type II Secure Network Coding. Byzantine attack means that attackers may modify the coded packets. For this case, since some important packets that are modified by an adversary will mislead the receivers and maybe cause the receivers to make wrong decisions, the modification detection of packets transmitted is also very important compared with the modification correction. How to detect the modification is a hot topic in network coding theory. With cryptographic approaches, Charles et al. [6] proposed a signature scheme for network coding based on Weil pairing on elliptic curves. Later, in [7, 8] the authors proposed some signature schemes by the linearity property of the packets in a coded system for network coding. In these schemes, one detects the modifications at intermediate nodes, so they are computationally expensive. With information-theoretic approaches, Ho et al. [12] showed a scheme in which Byzantine modification detections are done at sink nodes. In this scheme, he used random network coding by incorporating a polynomial hash value in each packet. By this way, the computing complexity is much less.

In this paper, to optimize the capacity loss and computation complexity, we propose a new scheme with Byzantine modification detection. Our scheme only needs one hash symbol which is much less than previous results in [12] and can achieve higher detection probability. Moreover, its computation complexity is lower than that in [12]. Furthermore, by combining cryptographic and information-theoretic approaches, we present secure network coding against wiretapping and Byzantine attacks.

The rest of this paper is organized as follows. In Section 2, we mainly give some necessary notations and definitions. In Section 3, we show some secure network coding against wiretapping or detecting Byzantine modification. One example is given at the end of this section. Some conclusions are presented in Section 4.

2. Primaries

In this section, some necessary notations and definitions such as network model, descriptions of a linear network code and all-or-nothing transform are presented. We denote matrices and linear spaces with bold uppercase letters and vectors with bold lowercase letters. All vectors are column unless some additional illustrations.

2.1. Network Model. Network coding has been leveraged as a generic technique in several types of wireless networks, such as vehicular ad hoc networks [13], wireless sensor networks [14], and Mesh networks [15],. In this paper, our focus is on secure network coding for acyclic wired networks and acyclic wireless networks which include parts of vehicular ad hoc networks, wireless sensor networks and Mesh networks. In detail, by the broadcast nature of the wireless interface each node is possiblly connected to several other nodes, where one node u connects to node v means that v is in the coverage of u's signal. By this way, we can obtain a directed graph G. Our attentions are mainly focused on the acyclic wired networks and acyclic wireless networks, both of which can be represented by a directed acyclic network $G = (V, E)$, where V is the set of nodes and E is the set of edges. The source node is denoted by s, and edges are denoted by round brackets $e = (u, v) \in E$, in which $v = \text{head}(e)$ and $u = \text{tail}(e)$. Let $\text{In}(v)(\text{Out}(v))$ be the set of edges that end (start) at a vertex $v \in V$.

2.2. Descriptions of a Linear Network Code. Now we give two kinds of descriptions of a linear network code.

Definition 1 (see [16] (Local Description of a Linear Network Code)). An ω-dimensional linear network code on an acyclic network over a base field $\mathbb{F}_q$ consists of a scalar $k_{d,e}$, called the local encoding coefficient, for every adjacent pair of channels (d, e) in the network. The matrix

$$\mathbf{K}_t = \left[k_{d,e}\right]_{d\in \text{In}(t), e\in \text{Out}(t)} \tag{1}$$

is called the local encoding kernel at node t.

Definition 2 (see [16] (Global Description of a Linear Network Code)). An ω-dimensional linear network code on an acyclic network over a base field $\mathbb{F}_q$ consists of a scalar $k_{d,e}$ for every adjacent pair of channels (d, e) in the network as well as a column ω-vector $\mathbf{f}_e$ for every channel e such that

(1) $\mathbf{f}_e = \sum_{d\in \text{In}(t)} k_{d,e}\mathbf{f}_d$ for $e \in \text{Out}(t)$;

(2) the vectors for the ω imaginary channels $e \in \text{In}(s)$ form a standard basis of $\mathbb{F}_q^\omega$.

The vector $\mathbf{f}_e$ is called the global encoding kernel for channel e.

For convenience of decode, during the transmission process, global encoding kernels are combined in the head of packets.

2.3. All-Or-Nothing Transform. In [17], Rivest presented a model of encryption for block ciphers, which is called *all-or-nothing transform* (AONT in short). AONT is defined for information-theoretic security [18]. In detail, let $\mathbb{F}_q$ be

a finite field and $\mathbb{F}_q^n$ be an n-dimensional space over $\mathbb{F}_q$. Suppose that $\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$. ϕ is named as an (n, q)-AONT if ϕ satisfies the following properties:

(1) ϕ is a bijection;

(2) If any $n-1$ of the n output values $y_1, y_2, \ldots, y_n$ is fixed, then the value of any input value x_i $(1 \le i \le n)$ is completely undetermined.

From this definition, for some input vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ (a basis of the space $\mathbb{F}_q^n$) and the corresponding output vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$, we have the following result.

Theorem 1. *For any (n, q)-AONT$\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$, if any $n-1$ of n output vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ is fixed, then each input vector $\mathbf{v}_i$ $(1 \le i \le n)$ is completely undetermined.*

Proof. Let $\mathbf{v}_i = (v_{i,1}\ v_{i,2} \cdots v_{i,n})^T$ and $\mathbf{u}_i = (u_{i,1}\ u_{i,2} \cdots u_{i,n})^T, 1 \le i \le n$, where $\mathbf{v}^T$ denotes the transpose of vector $\mathbf{v}$. For any i, from the definition of AONT if any $n-1$ of the n output $u_{1,i}, u_{2,i}, \ldots, u_{n,i}$ is fixed, then the value of any input $v_{1,i}, v_{2,i}, \ldots, v_{n,i}$ is completely undetermined. Therefore, when any $n-1$ of n output vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ is fixed, any input vector $\mathbf{v}_j$ is completely undetermined. □

If an (n, q)-AONT $\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is also $\mathbb{F}_q$-linear, ϕ is called a linear all-or-nothing transform. In fact, the linear AONT is very useful for constructing secure linear network coding because of its low computation complexity and convenience for decoding. In [18], Stinson proved that for prime power $q > 2$ and positive integer n there exists a linear (n, q)-ANOT. Moreover, he constructed the following linear AONT which can be implemented very efficiently. Let $q = p^k$, where p is prime and k is a positive integer. $\lambda \in \mathbb{F}_q$ such that $\lambda \notin \{n-1 \bmod p, n-2 \bmod p\}$. Then the linear function $\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ defined by $\phi(\mathbf{v}_1, \ldots, \mathbf{v}_n) = (\mathbf{u}_1, \ldots, \mathbf{u}_n)\mathbf{T}_S$ is a linear (n, q)-ANOT, where

$$\mathbf{T}_S = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & \lambda \end{pmatrix}. \tag{2}$$

We call $\mathbf{T}_S$ an (n, q)-ANOT matrix. This transform (and the inverse transform) can be implemented very efficiently. Given $\mathbf{V} = (\mathbf{v}_1 \cdots \mathbf{v}_n)$, we can compute $\mathbf{U} = (\mathbf{u}_1 \cdots \mathbf{u}_n)$ as follows:

$$\begin{aligned} \mathbf{u}_i &= \mathbf{v}_i + \mathbf{v}_n, \quad i = 1, \ldots, n-1, \\ \mathbf{u}_n &= \mathbf{v}_1 + \cdots + \mathbf{v}_{n-1} + \lambda \mathbf{v}_n. \end{aligned} \tag{3}$$

Conversely, given $\mathbf{U} = (\mathbf{u}_1 \cdots \mathbf{u}_n)$, we can compute $\mathbf{V} = (\mathbf{v}_1 \cdots \mathbf{v}_n)$ as follows:

$$\begin{aligned} \mathbf{v}_i &= \mathbf{u}_i - \gamma(\mathbf{u}_1 + \cdots + \mathbf{u}_{n-1} - \mathbf{u}_n), \quad i = 1, \ldots, n-1, \\ \mathbf{v}_n &= \gamma(\mathbf{u}_1 + \cdots + \mathbf{u}_{n-1} - \mathbf{u}_n), \end{aligned} \tag{4}$$

where $\gamma = (n - 1 - \lambda)^{-1}$.

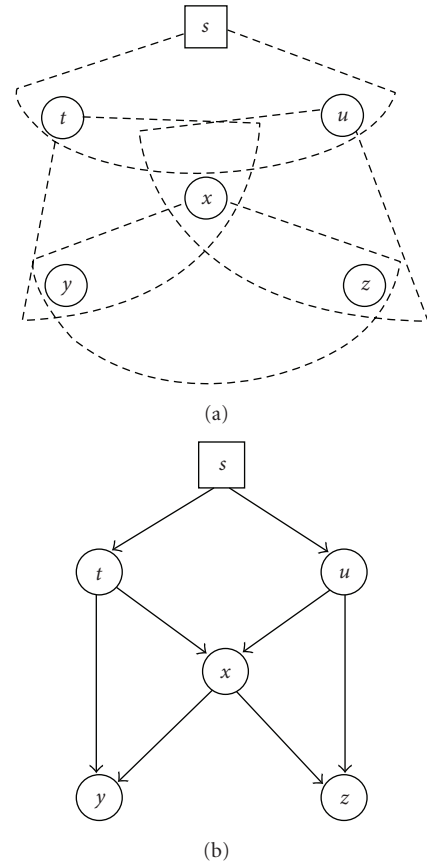

Figure 1: (a) Source s wants to send two messages to sink nodes y and z. t and u are within the coverage of source s, x is within the coverage of t and u, and terminals y and z are within the coverage of x. The region bounded by the dash lines denotes the signal coverage of the broadcast node. (b) The equivalent graph model of (a).

3. Main Schemes

In this section, we present some schemes that achieve different securities. Suppose that ω is the source rate. Each packet is represented by one vector in some linear space based on $\mathbb{F}_q$. The output packets of an AONT is called *pseudopackets*. In our schemes, AONT, the hash function and cryptosystem are public. The only shared secret is the key of the encryption when we use symmetric cryptsystem.

3.1. Against Wiretapping Attack. In wireless networks, because of the broadcast nature of the wireless interface, we canot determine which edges can be eavesdropped. So we canot obtain the same secure communication on the wireless networks if we made use of the scheme in [9] against wiretapping attacks. For example, consider the wireless network shown in Figure 1(a). From the presentation of the wireless network model in Section 2.1, we can get its equivalent graph model shown in Figure 1(b).

As for this wireless network, the scheme in [9] is not efficient and secure enough against wiretapping attack. In detail, the scheme in [9] for the network in Figure 1(b) is

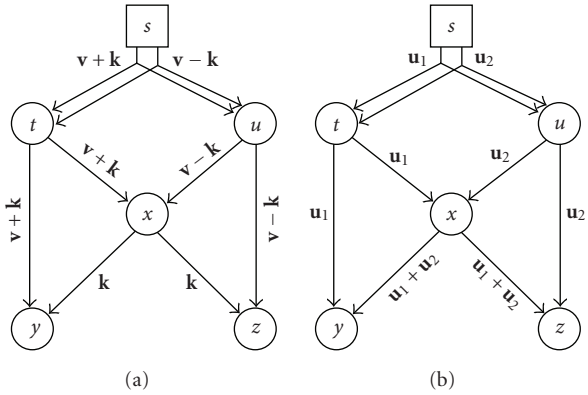

Figure 2: (a) The source node s sends $\mathbf{v}$ to sink nodes y and z. For security Cai et al. add an independent random packet $\mathbf{k}$ to $\mathbf{v}$. (b) At source node s we transform the packets $(\mathbf{v}_1, \mathbf{v}_2)$ into $(\mathbf{u}_1, \mathbf{u}_2)$ by a linear ANOT $\mathbf{T}_S$ then send the pseudopackets $\mathbf{u}_1$ and $\mathbf{u}_2$ to sink nodes.

shown in Figure 2(a). The collection of sets of wiretap edges is $\mathcal{A} = \{\{(t,y)\},\{(t,x)\}, \{(u,x)\},\{(x,y)\}, \{(x,z)\}, \{(u,z)\}\}$. For secure transmission, Cai et al. added some randomly chosen key vector $\mathbf{k}$ to the packet vector $\mathbf{v}$ at source node s. If the wiretapper only eavesdrops one channel in $\mathcal{A}$, he will get nothing about packet $\mathbf{v}$. However, the source only transmits one packet to the sink nodes in one transmission process. It means that one gets a secure transmission by giving up some capacity. In fact, if we choose an appropriate local encoding kernel for source node we can obtain the similarly secure transmission without capacity loss. In Figure 2(b), let a $(2,q)$-AONT matrix $\mathbf{T}_S$ defined in (2) be the local encoding kernel of the source node s, and $(\mathbf{u}_1, \mathbf{u}_2) = (\mathbf{v}_1, \mathbf{v}_2)\mathbf{T}_S$. The network codes of the intermediate nodes in Figures 2(a) and 2(b) are the same. If the wiretapper only wiretaps one channel, he can not get any meaningful information about $\mathbf{v}_1$ or $\mathbf{v}_2$ even if he knows $\mathbf{T}_S$. Therefore, we can get secure transmission without giving up any capacity.

In reality, however, as for this wireless network, it is possible that the wiretapper can eavesdrop all the network linkages because of the broadcast nature of the wireless interface. Then the previous two schemes are not secure enough in practical applications. So some cryptographic approaches are required to address this problem. In fact, by combining ANOT with symmetrical cryptography, without constrictions of wiretapping sets, we can construct secure network coding in the sense of cryptographic security. That means the wiretapper cannot obtain any massage if he has not the secret key. Our secure network coding is presented as follows.

Scheme 1. Let $\mathbf{v}_1, \ldots, \mathbf{v}_\omega$ be ω packets, where $\mathbf{v}_i \in \mathbb{F}_q^\omega$. An (ω, q)-AONT matrix $\mathbf{T}_S$ is the local encoding kernel of source node s.

Step 1. Let $(\mathbf{u}_1, \ldots, \mathbf{u}_\omega) = (\mathbf{v}_1, \ldots, \mathbf{v}_\omega)\mathbf{T}_S$. The source node encrypts $\mathbf{u}_\omega$ using AES cryptsystem (the source can also choose other high speed asymmetric cryptsystem. And the only secret for this scheme is the private key owned by the sender and receiver.) and sends out $\mathbf{u}_1, \ldots, \mathbf{u}_{\omega-1}, \mathbf{c}$, where $\mathbf{c} = E_{\mathrm{AES}}(\mathbf{u}_\omega)$.

Step 2. Based on Jaggi's construction of network coding [19] for wired networks and Rajawat's [20] for wireless networks, we can construct the codes for the intermediate nodes in wired networks and wireless networks, respectively.

Step 3. Each sink node first decodes the received packets and gets $\mathbf{u}_1, \ldots, \mathbf{u}_{\omega-1}, \mathbf{c}$ then decrypts $\mathbf{c}$ and obtains $\mathbf{u}_\omega$. By the inverse of $\mathbf{T}_S$, they get the original packets $\mathbf{v}_1, \ldots, \mathbf{v}_\omega$.

Time Complexity Analysis. Since the orders of matrix $\mathbf{T}_S$ and its inverse are both ω, the time complexity of multiplying $\mathbf{T}_S$ or $\mathbf{T}_S^{-1}$ is at most $O(\omega^3)$. In addition, there are two operations, encryption and decryption. So the more time complexity of this construction than those of Jaggi's and Rajawat's is $O(\omega^3)$ and the time for encryption and decryption.

Security Analysis. Since the network coding in this paper is linear, all of the network coding operations in the network are linear. The packets in the network are linear combinations of $\mathbf{u}_1, \ldots, \mathbf{u}_{\omega-1}, \mathbf{c}$. On one hand, if the rank of the linear packets that an adversary eavesdrops is less than ω, he can only get some (not all) of the packets $\mathbf{u}_1, \ldots, \mathbf{u}_{\omega-1}, \mathbf{c}$. By the definition of AONT, he can not obtain any original packet $\mathbf{v}_i$. On the other hand, even if the rank of eavesdropped packets is equal to ω, the wiretapper can not get the pseudopacket $\mathbf{u}_\omega$ without the private key. So he can not obtain any original packet either by Theorem 1.

In this model, we do not need to encrypt all the transmitted packets (In [18], all pseudopackets are encrypted, because this requires an adversary to decrypt all the blocks of ciphertext to determine any block of plaintext by the definition of AONT. Then the attack will be slowed down without any change in the size of the secrete key. Therefore, AONT is used to afford a certain amount of additional security for a block cipher encryption.). Only one is enough by combing with AONT. By Theorem 1, each original packet is relative to all the pseudopackets. When we encrypt one of the pseudopackets, the wiretapper canot get all of the pseudopackets without the private key. So he canot obtain any original packet. For example in Figure 2(b), we only need to encrypt $\mathbf{u}_1$ or $\mathbf{u}_2$, then the wiretapper can not get any meaningful information about $\mathbf{v}_1$ and $\mathbf{v}_2$. The security here combines the information-theoretic security with cryptographic security. However, by the wooden barrel theory the whole security of this scheme is reduced to cryptographic security.

Now, we show the advantages of AONT as the local encoding kernel of the source node. Firstly, from the information-theoretic point: we not only increase the achievable throughput, but also get secure transmission. Secondly, from the cryptographic point: we only need to encrypt one packet out from the source node instead of encrypting all the packets which will be sent to sink nodes. Moreover, we can save lots of time consumption, explained from Table 1, where ω denotes the source rate, the length of each packet is 2 bytes and "clk" is the abbreviated clock.

3.2. Byzantine Modification Detection. Since some important packets that are modified by an adversary will mislead

Table 1: The time consumptions of different encryption models.

ω	AES (clk)	AES with Parallel Computing (clk)	AONT+AES (clk)
5	9380	2516	1888
10	18760	3316	1898
20	37520	4916	1918
40	75040	8116	1958

the receivers and may cause the receivers to make wrong decisions, the modification detection of packets transmitted is also very important compared with the modification correction in both wireless and wired networks. In this subsection, we present a scheme to detect the Byzantine modification combining AONT with a simple polynomial hash function.

By the definition of AONT, we find that if one of the pseudopackets from source node is damaged, then it is likely that every packet will be damaged. This is the error-propagation property of AONT. So we can append a suitable block of redundancy to the packet before applying an AONT. And this redundancy can be used to verify the integrity of the packets and also can be removed after decode.

Suppose source s multicast ω vector packets $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\omega$ to the sink nodes. For convenience, each packet in the network is represented by a column vector $\hat{\mathbf{v}}$ of $d+1$ ($d \geq \omega$) symbols over a finite field $\mathbb{F}_q$, where the first d entries are data symbols and the last one is a redundant hash symbol. The hash symbol in each augmented packet is given by a hash function $\psi : \mathbb{F}_q^d \to \mathbb{F}_q$ of the data symbols. Of course, we can choose any nonlinear hash function. In fact, we find that the security can be ensured by a simple nonlinear function. In detail, we take the following simple nonlinear function as the secure hash function in this scheme.

Let $\psi : \mathbb{F}_q^d \to \mathbb{F}_q$ be the function mapping $(x_1, \ldots, x_d)^T$ to

$$\psi(x_1, \ldots, x_\omega) = x_1^2 + \cdots + x_d^2. \tag{5}$$

Denote $\mathbf{v}_i = (v_{i,1}\ v_{i,2} \cdots v_{i,d})^T, 1 \leq i \leq \omega$. Denote the augmented packets by

$$\hat{\mathbf{v}}_i = \begin{pmatrix} \mathbf{v}_i \\ h_i \end{pmatrix}, \tag{6}$$

where $\binom{\mathbf{x}}{\mathbf{y}}$ denotes the concatenation of two vectors $\mathbf{x}$ and $\mathbf{y}$, and h_i is the hash symbol satisfying

$$h_i = \psi(v_{i,1}, v_{i,2}, \ldots, v_{i,d}) = \sum_{j=1}^{d} v_{i,j}^2, \quad 1 \leq i \leq \omega. \tag{7}$$

Now we give a brief description of our scheme as follows.

Scheme 2. **Initialization**: For each original packets $\mathbf{v}_i, 1 \leq i \leq \omega$, the source s calculates the hash values $h_i, 1 \leq i \leq \omega$, and obtains the augmented packets $\hat{\mathbf{v}}_i, 1 \leq i \leq \omega$, by concatenating the hash value h_i to each original packet $\mathbf{v}_i$.

Step 1. The source s takes the AONT matrix $\mathbf{T}_S$ as its local encoding kernel and computes $(\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_\omega) = (\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_\omega)\mathbf{T}_S$. Then sends out $\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_\omega$.

Step 2. Based on the Jaggi's construction of network coding for wired networks and Rajawat's for wireless networks, we can construct the codes for the intermediate nodes.

Step 3. Each sink node first decodes the received packets and gets $\hat{\mathbf{v}}'_1, \ldots, \hat{\mathbf{v}}'_\omega$. (Since in this scheme, we donot consider the wiretapping but the integrity of the packets, it does not need to encrypt any pseudopackets from the source. We note that $\mathbf{T}_S$ is the local encoding kernel of the source s, and thus we can decode directly and get the packets $\hat{\mathbf{v}}'_1, \ldots, \hat{\mathbf{v}}'_\omega$.) Then it verifies whether $h'_i = \sum_{j=1}^{d} {v'_{i,j}}^2, 1 \leq i \leq \omega$. If $h'_i = \sum_{j=1}^{d} {v'_{i,j}}^2$ for all i, then there does not exist modification on the transmission and $\hat{\mathbf{v}}_i = \hat{\mathbf{v}}'_i, 1 \leq i \leq \omega$. Finally, they remove the hash values and obtain the original packets $\mathbf{v}_1, \ldots, \mathbf{v}_\omega$.

Time Complexity Analysis. This scheme is similar to Scheme 1. The differences are additional calculations for hash symbols in Step 1 and verifications for hash symbols in Step 3. The time complexity of these two operations is $O(\omega^2)$. So the time complexity of Steps 1 and 3 is polynomial on the length ω of the packet vector and equal to $O(\omega^3)$. So the total time complexity of this secure network coding construction is only $O(\omega^3)$ more than that of Jaggi's for wired networks and Rajawat's for wireless networks.

Security Analysis. Based on the model above, an adversary successfully modifies the packet that he can construct the logical hash symbol after modifying the data symbols (actually here he modifies the pseudopackets). From the following theorem we will find that an adversary can construct a logical hash symbol after modifying the data symbols with a very low probability.

Theorem 2. *In Scheme 2, the probability of not detecting an error is at most* $(1/q)^\omega$*, where* ω *is the source rate.*

To prove this theorem, we first prove the following two lemmas.

Lemma 3. *Given the vector* $\mathbf{a}$ *and scalar value* c*, the probability of randomly choosing a vector* $\mathbf{v} \in \mathbb{F}_q^n$ *such that the inner product* $\mathbf{v} \cdot \mathbf{a} = c$ *is* $1/q$*.*

Proof. The number of points on the hyperplane $\{\mathbf{v} \in \mathbb{F}_q^n \mid \mathbf{v} \cdot \mathbf{a} = c\}$ is q^{n-1}. And the cardinality of the field $\mathbb{F}_q^n$ is q^n. So the probability of choosing a vector $\mathbf{v}$ such that $\mathbf{v} \cdot \mathbf{a} = c$ is $q^{n-1}/q^n = 1/q$. □

Lemma 4. *The probability of randomly choosing a vector* $\mathbf{v} \in \mathbb{F}_q^n$ *such that*

$$\begin{aligned} \mathbf{v}\cdot\mathbf{a}_1 &= c_1 \\ \mathbf{v}\cdot\mathbf{a}_2 &= c_2 \\ &\vdots\ \vdots \\ \mathbf{v}\cdot\mathbf{a}_n &= c_n \end{aligned} \tag{8}$$

is at most $(1/q)^n$*, where the vectors* $\mathbf{a}_i, 1 \le i \le n$ *and scalar values* $c_i, 1 \le i \le n$ *are fixed and independent.*

Proof. By Lemma 3, we randomly choose a vector $\mathbf{v} \in \mathbb{F}_q^n$ such that $\mathbf{v}\cdot\mathbf{a}_i = c_i$ with probability $1/q$. Then the probability that choosing an appropriate vector $\mathbf{v} \in \mathbb{F}_q^n$ such that $\mathbf{v}$ satisfies (8) is at most $(1/q)^n$. □

Now we prove Theorem 2.

Let $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_\omega$ be the ω packet vectors transmitted, each of which consists of $d+1$ symbols from a finite field $\mathbb{F}_q$ and is a column vector. The first d entries are data symbols and the rest one is the redundant hash symbol. It can be represented as

$$\hat{\mathbf{v}}_i = \begin{pmatrix} \mathbf{v}_i \\ h_i \end{pmatrix} = \begin{pmatrix} v_{i,1} \\ \vdots \\ v_{i,d} \\ h_i \end{pmatrix}, \quad i = 1, \ldots, \omega, \tag{9}$$

where $\mathbf{v}_i$ denotes the data. The hash symbol $h_i = \psi(v_{i,1}, v_{i,2}, \ldots, v_{i,d})$, $i = 1, \ldots, \omega$.

The matrix $\mathbf{T}_S$ is the local encoding kernel of source node, and let

$$(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_\omega) = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\omega)\mathbf{T}_S, \tag{10}$$

$$(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_\omega) = (\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_\omega)\mathbf{T}_S. \tag{11}$$

So the hash symbols satisfy

$$(h_{\mathbf{u}_1}, h_{\mathbf{u}_2}, \ldots, h_{\mathbf{u}_\omega}) = (h_1, h_2, \ldots, h_\omega)\mathbf{T}_S, \tag{12}$$

where $h_{\mathbf{u}_i}$ denotes the hash symbol of $\mathbf{u}_i$. Therefore, $h_{\mathbf{u}_i} = h_i + h_\omega$ because $\mathbf{u}_i = \mathbf{v}_i + \mathbf{v}_\omega$ from (3). Notice that the hash function is not a linear function, that is, $h_{\mathbf{u}_i} \neq h_{\mathbf{v}_i+\mathbf{v}_\omega}$. When the adversary modifies some pseudopacket $\mathbf{u}_i$, he has to modify the hash symbol $h_{\mathbf{u}_i}$ such that the sink nodes can not detect the modification. The proof can be completed by two steps.

Step 1. We suppose that only the first pseudopacket $\mathbf{u}_1$ is modified and the new pseudopacket is denoted by $\mathbf{u}_1'$. Let $\mathbf{u}_1' = \mathbf{u}_1 + \Delta\mathbf{u}_1$ and $h_{\mathbf{u}_1'} = h_{\mathbf{u}_1} + \Delta h_{\mathbf{u}_1}$. $\Delta\mathbf{u}_1$ is known to the adversary.

The adversary wants to know $\Delta h_{\mathbf{u}_1}$. By the representation of $\mathbf{T}_S^{-1}$ in (4), we have

$$\mathbf{v}_i' = \begin{cases} (1-\gamma)\mathbf{u}_1 - \cdots - \gamma\mathbf{u}_{\omega-1} \\ \quad +\gamma\mathbf{u}_\omega + (1-\gamma)\Delta\mathbf{u}_1, & i = 1, \\ -\gamma\mathbf{u}_1 - \cdots - \gamma\mathbf{u}_{i-1} - \gamma\mathbf{u}_{i+1} - \cdots - \gamma\mathbf{u}_{\omega-1} \\ \quad +(1-\gamma)\mathbf{u}_i + \gamma\mathbf{u}_\omega - \gamma\Delta\mathbf{u}_1, & i = 2, \ldots, \omega-1, \\ \gamma(\mathbf{u}_1 + \mathbf{u}_2 + \cdots + \mathbf{u}_{\omega-1} - \mathbf{u}_\omega) \\ \quad +\gamma\Delta\mathbf{u}_1, & i = \omega. \end{cases} \tag{13}$$

So even only one of $\mathbf{u}_1$ is modified, all the packets $\hat{\mathbf{v}}_i, i = 1, \ldots, \omega$ will be changed. From (13), we have

$$\Delta\mathbf{v}_i = \begin{cases} (1-\gamma)\Delta\mathbf{u}_1, & i = 1, \\ -\gamma\Delta\mathbf{u}_1, & i = 2, \ldots, \omega-1, \\ \gamma\Delta\mathbf{u}_1, & i = \omega. \end{cases} \tag{14}$$

Firstly, suppose that $\Delta u_{1,1} \neq 0$ and $\Delta u_{1,j} = 0, j = 2, \ldots, d$. Then, by the definition of hash function in (5),

$$\begin{aligned} \Delta h_i = h_i' - h_i &= \psi\left(v_{i,1}', v_{i,2}, \ldots, v_{i,d}\right) - \psi(v_{i,1}, v_{i,2}, \ldots, v_{i,d}) \\ &= \begin{cases} 2(1-\gamma)v_{1,1}\Delta u_{1,1} \\ \quad +(1-\gamma)^2\Delta u_{1,1}^2, & i = 1, \\ -2\gamma v_{i,1}\Delta u_{1,1} + \gamma^2\Delta u_{1,1}^2, & i = 2, \cdots, \omega-1, \\ 2\gamma v_{\omega,1}\Delta u_{1,1} + \gamma^2\Delta u_{1,1}^2, & i = \omega. \end{cases} \end{aligned} \tag{15}$$

In (15), $v_{i,1}$, $i = 1, \ldots, \omega$, are unknown to the adversary except for γ and $\Delta u_{1,1}$. Moreover, Δh_i is unknown to the adversary, but fixed. By Lemma 4, the probability of constructing an appropriate Δh_i satisfying (15) is at most $(1/q)^\omega$.

Secondly, suppose that $u_{1,1}$ and $u_{1,2}$ are modified. Let $u_{1,1}' = u_{1,1} + \Delta u_{1,1}$ and $u_{1,2}' = u_{1,2} + \Delta u_{1,2}$. Then,

$$\begin{aligned} \Delta h_i = h_i' - h_i &= \phi\left(v_{i,1}', v_{i,2}', v_{i,3} \ldots, v_{i,d}\right) - \phi(v_{i,1}, v_{i,2}, \ldots, v_{i,d}) \\ &= \begin{cases} 2(1-\gamma)\left(v_{1,1}\Delta u_{1,1} + v_{1,2}\Delta u_{1,2}\right) \\ \quad +(1-\gamma)^2\left(\Delta u_{1,1}^2 + \Delta u_{1,2}^2\right), & i = 1, \\ -2\gamma\left(v_{i,1}\Delta u_{1,1} + v_{i,2}\Delta u_{1,2}\right) + \gamma^2\left(\Delta u_{1,1}^2 + \Delta u_{1,2}^2\right), & \\ \qquad\qquad i = 2, \ldots, \omega-1, & \\ 2\gamma\left(v_{\omega,1}\Delta u_{1,1} + v_{\omega,2}\Delta u_{1,2}\right) + \gamma^2\left(\Delta u_{1,1}^2 + \Delta u_{1,2}^2\right), & \\ \qquad\qquad i = \omega. & \end{cases} \end{aligned} \tag{16}$$

From (16), the probability of randomly choosing the right $v_{i,1}$ and $v_{i,2}$ for constructing logical hash symbols is also at most $(1/q)^\omega$ by Lemma 4.

Thirdly, when the adversary modifies more data symbols of $\mathbf{u}_1$, by the similar method, we can prove that the probability of constructing the logical hash symbols is at most $(1/q)^{\omega}$.

Step 2. When the adversary modifies more pseudopackets $\mathbf{u}_i$ at one time, from the similar method above, the probability of constructing the logical hash symbols is no more than $(1/q)^{\omega}$.

From the proof of Theorem 2, we have the following two corollaries.

Corollary 1. *The probability of not detecting an error is not related to both the number of modified packets $\Delta\mathbf{u}_i$ and the symbols of one pseudopacket, but the cardinality of $\mathbb{F}_q$ and the source rate.*

Corollary 2. *If the redundant hash symbol in the packet is a constant or a linear function of the data symbols, then the scheme can not defend the Byzantine modification.*

Proof. First, when the hash symbol is a constant. The adversary only modifies the data symbols and keeps the hash symbols unchanged. Then the receivers can not detect the modification.

Second, when the hash symbol is a linear function of the data symbols, we have

$$\begin{aligned} h_{\mathbf{u}_i} &= h_i + h_\omega = \phi(\mathbf{v}_i) + \phi(\mathbf{v}_\omega) \\ &= \phi(\mathbf{v}_i + \mathbf{v}_\omega), \quad i = 1, \ldots, \omega - 1, \\ h_{\mathbf{u}_\omega} &= \phi(\mathbf{v}_1 + \cdots + \mathbf{v}_{\omega-1} + \lambda\mathbf{v}_\omega). \end{aligned} \tag{17}$$

By (15), $\Delta h_i = h_i' - h_i = h_{\mathbf{v}_i' - \mathbf{v}_i} = \phi(\mathbf{v}_i' - \mathbf{v}_i) = \phi(\Delta\mathbf{v}_i)$. $\Delta\mathbf{u}_j$ are known by attackers. So, by the relationship between $\Delta\mathbf{v}_i$ and $\Delta\mathbf{u}_j$, Δh_i are easily calculated and this scheme canot defend Byzantine modification. □

3.3. Against Wiretapping and Byzantine Attacks. Further, by combining with Scheme 1, we can improve Scheme 2 to against wiretapping attack. Before sending out $\hat{\mathbf{u}}_i$, $i = 1, \ldots, \omega$, the source encrypts the last packet $\hat{\mathbf{u}}_\omega$ and denotes the encrypted packet by $\hat{\mathbf{c}}$. The aim is to prevent wiretapper from recovering any original packets.

Scheme 3 provides not only security but also authenticity.

Scheme 3. **Initialization**: For each packet $\mathbf{v}_i, 1 \le i \le \omega$, the source calculates the hash values $h_i, 1 \le i \le \omega$, and obtains the augmented packets $\hat{\mathbf{v}}_i, 1 \le i \le \omega$, by concatenating the hash value h_i to each original packets $\mathbf{v}_i$.

Step 1. The source takes $\mathbf{T}_S$ as its local encoding kernel. Computes $(\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_\omega) = (\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_\omega)\mathbf{T}_S$ and encrypts $\hat{\mathbf{u}}_\omega$ using AES cryptsystem (Here we use symmetry cryptsystem. Because if we use asymmetry cryptsystem, by the public key the adversary may successfully modify all the pseudopackets at the same time when he controls ω edge disjoint paths.) to get $\hat{\mathbf{c}} = E_{\mathrm{AES}}(\hat{\mathbf{u}}_\omega)$. Then sends out $\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_{\omega-1}, \hat{\mathbf{c}}$.

Step 2. Based on the Jaggi's construction of network coding for wired networks and Rajawat's for wireless network, we can construct the codes for the intermediate nodes.

Step 3. Each sink node first decodes the received packets and gets $\hat{\mathbf{u}}_1', \ldots, \hat{\mathbf{u}}_{\omega-1}', \hat{\mathbf{c}}'$, then gets $\hat{\mathbf{u}}_\omega' = D_{\mathrm{AES}}(\hat{\mathbf{c}}')$ by decrypting $\hat{\mathbf{c}}'$. Verify whether $h_i' = \sum_{j=1}^{d} {v_{i,j}'}^2$, $1 \le i \le \omega$. If $h_i' = \sum_{j=1}^{d} {v_{i,j}'}^2$ for all i, there does not exist modification on the transmission. They get the original packets $\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_\omega$ by $\mathbf{T}_S^{-1}$.

These three schemes are based on Jaggi's construction for wired networks and Rajawat's construction for wireless networks. Actually, we can also use Ho's random network coding [21]. In Scheme 2, the only change is to randomly choose the local encoding kernels from a large finite field. Except for the change in Scheme 2, in Schemes 1 and 3 the packets from the source will be appended with an ω-dimensional identity vector, the global encoding kernel, before being sent out. However, random network coding for wireless networks requires a large alphabet size to render networks robust to link failures.

Example 5. We construct a secure network code on the wireless network in Figure 1(a) to detect Byzantine modifications. Suppose the base field is $\mathbb{F}_5$. Let $\lambda = 3 \notin \{0, 1 \bmod 5\}$. Then $\gamma = (2 - 1 - 3)^{-1} = 2$. So the AONT matrix $\mathbf{T}_S$, which is used to encode the two original packets at the source node s, is

$$\mathbf{T}_S = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix}. \tag{18}$$

Suppose the packets $\mathbf{v}_1$ and $\mathbf{v}_2$ will be sent to the sink nodes y and z. The two encoded packets (pseudopackets) from the source are

$$(\mathbf{u}_1\ \mathbf{u}_2) = (\mathbf{v}_1\ \mathbf{v}_2)\mathbf{T}_S = (\mathbf{v}_1 + \mathbf{v}_2\ \mathbf{v}_1 + 3\mathbf{v}_2). \tag{19}$$

Then the pseudopackets transmitted on the edges are shown in Figure 3. When an adversary wiretaps any one of $\mathcal{A} = \{\{(t, y)\}, \{(t, x)\}, \{(u, x)\}, \{(x, y), \{(x, z)\}\}, \{(u, z)\}\}$, he can not get any meaningful information about packet $\mathbf{v}_1$ or $\mathbf{v}_2$. If we encrypt the packet $\mathbf{u}_2$, then the adversary can get nothing even when he wiretaps all the channels.

Let $\hat{\mathbf{v}}_1 = (1\ 2\ 3\ 4)^T, \hat{\mathbf{v}}_2 = (3\ 2\ 4\ 4)^T$. The last symbol is the hash symbol and calculated using the hash function in (4). The two augmented pseudopackets are

$$(\hat{\mathbf{u}}_1\ \hat{\mathbf{u}}_2) = \begin{pmatrix} 1 & 3 \\ 2 & 2 \\ 3 & 4 \\ 4 & 4 \end{pmatrix}\mathbf{T}_S = \begin{pmatrix} 4 & 0 \\ 4 & 3 \\ 2 & 0 \\ 3 & 1 \end{pmatrix}. \tag{20}$$

Suppose that an adversary modifies the data packet $\hat{\mathbf{u}}_1$ and let $\hat{\mathbf{u}}_1' = (3\ 4\ 2\ 3)^T$. Then, the receiver can decode the packet vectors and get $\hat{\mathbf{v}}_1' = (1\ 2\ 4\ 4)^T, \hat{\mathbf{v}}_2' = (2\ 2\ 3\ 4)^T$. It is

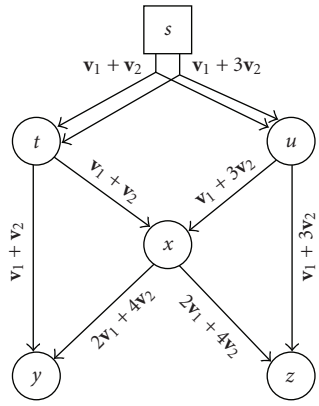

FIGURE 3: The source node s transforms the packets $\mathbf{v}_1$ and $\mathbf{v}_2$ by $\mathbf{T}_S$, and sends the outputs $\mathbf{u}_1 = \mathbf{v}_1 + \mathbf{v}_2$ and $\mathbf{u}_2 = \mathbf{v}_1 + 3\mathbf{v}_2$ to sink nodes y and z.

easy to verify that $4 \neq 1 + 4 + 16 \mod 5 = 1$ and $4 \neq 4 + 4 + 9 \mod 5 = 2$. So the receivers can find that the packets are modified.

4. Conclusion

For secure transmission, if only the information-theoretic approach is used, some bandwidth has to be given up or a high computation complexity is necessary. As to cryptographic approach, all the packets have to be encrypted against wiretapping. Even if the data is hashed and appended with its hash value, one may not detect the modifications when the adversary modifies the data and its hash value simultaneously. To address these problems, we combine the information-theoretic approach with cryptographic approach to design secure network coding. On one hand, we do not give up any network capacity to achieve the same security as that of Cai and Yeung. More importantly, our Scheme 1 does not require any restrictions on the wiretapping sets compared with that of Cai and Yeung. It means that our secure network coding is suitable for both wired networks and wireless networks. On the other hand, we decrease the resource consumptions of encryption and decryption. Furthermore, based on some simple hash function, our Scheme 2 is designed to detect the Byzantine modification. It can achieve a high detection probability with only one hash symbol and low computation complexity. In the end, combining the two schemes above we propose Scheme 3 which provides not only security but also authenticity.

Acknowledgments

The authors would like to thank editor and all the anonymous reviewers for their helpful advices. This paper was supported by the National Natural Science Foundation of China and the Research Grants Council of Hong Kong Joint Research Scheme (no. 60731160626), the National Natural Science Foundation of China (no. 60821001), the Specialized the Foundation for the Author of National Excellent Doctoral Dissertation of PR China (FANEDD) (Grant no. 200951), the 111 Project (no. B08004).

References

[1] 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.

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

[3] P. A. Chou and Y. Wu, "Network coding for the internet and wireless networks," *IEEE Signal Processing Magazine*, vol. 24, no. 5, pp. 77–85, 2007.

[4] S. Deb, M. Effros, T. Ho, et al., "Network coding for wireless applications: a brief tutorial," in *Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN '05)*, London, UK, May 2005.

[5] K. Jain, "Security based on network topology against the wiretapping attack," *IEEE Wireless Communications*, vol. 11, no. 1, pp. 68–71, 2004.

[6] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS '06)*, pp. 857–863, Princeton, NJ, USA, January 2006.

[7] F. Zhao, T. Kalker, M. Médard, and K. J. Han, "Signatures for content distribution with network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '07)*, pp. 556–560, June 2007.

[8] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: signature schemes for network coding," in *Proceedings of the Public Key Cryptography (PKC '09)*, vol. 5443 of *Lecture Notes in Computer Science*, pp. 68–87, 2009.

[9] N. Cai and R. W. Yeung, "Secure network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '02)*, June 2002.

[10] J. Feldman, T. Malkin, R. A. Servedio, and C. Stein, "On the capacity of secure network coding," in *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, October 2004.

[11] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," in *Proceedings of the 1st Workshop on Network Coding, Theory, and Applications (NETCOD '05)*, 2005.

[12] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Karger, "Byzantine modification detection in multicast networks with random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2798–2803, 2008.

[13] S.-H. Lee, U. Lee, K.-W. Lee, and M. Gerla, "Content distribution in VANETs using network coding: the effect of disk I/O and processing O/H," in *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08)*, pp. 117–125, San Francisco, Calif, USA, July 2008.

[14] Z. Guo, P. Xie, J.-H. Cui, and B. Wang, "On applying network coding to underwater sensor networks," in *Proceedings of the 1st ACM International Workshop on Underwater Networks (WUWNet '06)*, pp. 109–112, ACM Press, New York, NY, USA, 2006.

[15] A. Al Hamra, C. Barakat, and T. Turletti, "Network coding for wireless mesh networks: a case study," in *Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '06)*, 2006.

[16] R. W. Yeung, *Information Theory and Network Coding*, Springer, New York, NY, USA, 2008.

[17] R. L. Rivest, "All-or-nothing encryption and the packege transform," in *Proceedings of the Fast Software Encryption (FSE '97)*, vol. 1267 of *Lecture Notes in Computer Science*, pp. 210–218, 1997.

[18] D. R. Stinson, "Something about all or nothing (transforms)," *Designs, Codes, and Cryptography*, vol. 22, no. 2, pp. 133–138, 2001.

[19] S. Jaggi, P. Sanders, P. A. Chou, et al., "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.

[20] K. Rajawat and G. B. Giannakis, "Non-random wireless network coding," in *Proceedings of the 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops, (SECON '09)*, pp. 1–6, Rome, Italy, June 2009.

[21] T. Ho, M. Médard, R. Köetter, et al., "A random linear network coding approach to mul ticast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.