Secure Network Coding with Minimum Overhead Based on Hash Functions

Majid Adeli and Huaping Liu, *Senior Member, IEEE*

***Abstract*—In order to achieve complete security in networks with network coding, it has been shown that at least k original information symbols must be substituted with k uniformly distributed random symbols chosen from the same code field, where k is the maximum number of independent coding vectors accessible to the adversary. Thus, sender nodes must dedicate a portion of the bandwidth to transmit noise, which could cause a significant signaling overhead. In this letter, we propose a scheme that requires only one noisy symbol to be embedded in the original information symbol vector to achieve complete secrecy. This scheme utilizes hash functions to generate different random noisy symbols by using the only uniformly distributed random symbol and the information symbols. It has two main advantages: (a) complete security is guaranteed regardless of the number of independent coding vectors acquired by the adversary; and (b) signaling overhead to obtain complete security is minimized.**

***Index Terms*—Network coding, complete security, hash function, information rate.**

I. Introduction

NETWORK coding is a general form of the traditional routing schemes [1]. In network coding based networks, each intermediate node has the ability to process the data packets it receives and then transmit the processed symbols over its outgoing channels [2], [3]. If an intermediate node has n input and m output channels, then it should have m different functions of its n different input symbols. Network coding can be broadly categorized into two classes: linear network coding and nonlinear network coding [2]. Linear network coding, the focus of this paper, has been widely considered because it is simple and practical.

There have been research efforts on achieving secure network coding shortly after network coding was introduced in [1], [4]. In [5], a necessary and sufficient condition for the feasibility of constructing a secure linear network code is derived. In [6], [7], connections between secure network coding and secret sharing are established and trade-offs between security, code alphabet size, and multicast rate of secure linear network codes are derived. Two different types of security are considered in [8] and a code field size bound is obtained for each case. The scheme discussed in [9] is based on mixing the original raw information symbols with a vector of uniformly distributed random symbols of the same length by Xoring them, and then concatenating the resultant vector with the random symbol vector. The concatenated vector is finally sent through the network. Other works such as [10], [11] have addressed security issues when the adversary can manipulate the functionality of some intermediate nodes in the network. To deal with such type of attacks, called Byzantine attacks, it is suggested to use error-correcting codes at the source nodes. Coding at the source nodes is also considered in [12] as a way to make the network code secure. In this case, the source nodes apply channel coding on information symbols before network code is applied. Vulnerabilities of security schemes in network coding are briefly analyzed in [13].

Randomized network coding in mobile, unreliable, noisy networks is considered in [14]. To make the network code robust against these issues and at the same time keep the overhead low, training sequences are embedded in data packets and then channel coding is applied on the entire data packet.

In this paper, we propose a new scheme to achieve complete security for linear network coding with minimum information overhead and without changing the code field size or the functionality of intermediate nodes. Here, complete security means that the adversary will not gain any meaningful information by wiretapping the network links; where meaningful information implies any linear combination of the information symbols. The basic idea to achieve complete security with minimum overhead is to utilize hash functions [15], instead of by adding redundancy to the information message.

II. Problem Definition

We consider an interference-and-delay-free network with linear network coding where all channels act as noiseless, linear time-invariant filters. As commonly adopted, we denote this network as $G(V, E)$ [1], where G is an acyclic directed graph with unit capacity for each edge, V denotes the set of all nodes (vertices) in the network, and E represents the set of all edges (links or channels) that connect the network nodes.

Suppose that a set of source nodes, named S, send data over the network. All information symbols generated by the source nodes form an information vector of length n expressed as

$$\boldsymbol{m} = (m_1, m_2, \cdots, m_n)^T, \tag{1}$$

where $m_i \in GF(q)$ and $\|S\| \leq n$. Since we consider linear network coding, each edge is associated with a length-n vector, called coding vector. For $i = 1$ to $N \stackrel{\text{def}}{=} \|E\|$, we denote each coding vector as

$$\boldsymbol{v}_i = (\nu_{i1}, \nu_{i2}, \cdots, \nu_{in})^T. \tag{2}$$

Coding vectors are defined over the same code field to which the information symbols belong. At any time instant, the symbol transmitted on the ith edge, denoted by x_i, is the inner product of the information vector and the coding vector

Manuscript received August 6, 2009. The associate editor coordinating the review of this letter and approving it for publication was S. A. Jafar.

The authors are with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon 97331 U.S.A. (e-mail: {adeli, hliu}@eecs.oregonstate.edu).

Digital Object Identifier 10.1109/LCOMM.2009.12.091648

associated with that edge. So x_i, which is a linear combination of all information symbols can be expressed as

$$x_i = \boldsymbol{v}_i^T \boldsymbol{m}, \quad \text{for } i = 1, \cdots, N. \tag{3}$$

Assume that the adversary has access to at most k $(k < n)$ independent edges, where "independent edges" means their corresponding coding vectors are linearly independent. In order to prevent the adversary from acquiring any meaningful information, we should substitute some of information symbols (say r) in the original information vector with uniformly distributed random symbols, z, selected from the same code field and then run a linear transformation on the modified information vector, $\tilde{\boldsymbol{m}}$, which is expressed as

$$\tilde{\boldsymbol{m}} = (m_1, m_2, \cdots, m_{n-r}, z_1, \cdots, z_r)^T. \tag{4}$$

A final transformation is applied on $\tilde{\boldsymbol{m}}$ because if we simply place the components of $\tilde{\boldsymbol{m}}$ on the output edges of the source nodes, then on some of the outgoing edges we will have plain, disclosed information symbols, which are not secure. This linear transformation can be modeled by a simple matrix multiplication at source nodes. This way, at the output edges of each source node we have a linear combination of meaningful information symbols and noisy symbols. Under some constraints that will be discussed in Sec. III, each edge in the network carries only noise, yielding complete security.

The noisy symbols embedded to ensure security cause a reduction in information rate. It has been shown that with existing approaches, the number of noisy symbols, r, must be at least equal to k [5]–[9]. This substitution results in significant reduction in information rate.

III. Existing Schemes

We review existing approaches to achieve completely secure linear network coding which has been proposed in literatures such as [5]–[9]. The basic idea is to make sure that the adversary will not be able to neutralize the effect of the noisy symbols on meaningful data by linearly combining available x_i's. To achieve this goal, we consider

$$\boldsymbol{V}\tilde{\boldsymbol{m}} = [\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \cdots \ \boldsymbol{v}_k]^T \tilde{\boldsymbol{m}} = [x_1 \ x_2 \ \cdots \ x_k]^T, \tag{5}$$

where $\boldsymbol{v}_i$ denotes an n-dimensional coding vector corresponding to the ith edge available to the adversary, and $\boldsymbol{V}$ is the matrix obtained by putting all these coding vectors together. If we partition $\boldsymbol{V}_{k\times n}$ into two submatrices, $\boldsymbol{A}_{k\times(n-r)}$ and $\boldsymbol{B}_{k\times r}$, corresponding to the parts of $\tilde{\boldsymbol{m}}$ that contain information and noisy symbols respectively, then we have

$$\left[\boldsymbol{A}_{k\times(n-r)} | \boldsymbol{B}_{k\times r}\right] \tilde{\boldsymbol{m}} = (x_1, \cdots, x_k)^T. \tag{6}$$

In order to maintain complete security, any linear combination of the rows of matrix $\boldsymbol{V}$ should not yield to a length-n vector with all zeros for its last r components, because otherwise the adversary would be able to gain meaningful information (i.e., a linear combination of the original information symbols) by linearly combining his wiretapped symbols. This requires the submatrix $\boldsymbol{B}$ to have independent rows, which necessitates $r \geq k$. Therefore, the number of substituted noisy symbols in $\tilde{\boldsymbol{m}}$ must not be less than k.

IV. Proposed Scheme

From Sec. III, we see that existing schemes are based on embedding uniformly distributed random symbols in the information vector and mixing them with information symbols through linear combinations. This approach makes linear network code secure at the expense of potentially a significantly increased overhead. Also, for any given number of embedded noisy symbols, the adversary could break the security by acquiring the symbols on more independent edges.

The goal of this paper is to minimize overhead while guaranteeing complete security in network coding. Our scheme is based on using hash functions. A hash function h maps an input x of arbitrary (but finite) bit-length to an output $h(x)$ of fixed, pre-specified bit-length [15]. In addition, given h and an input x, $h(x)$ can be calculated with polynomial time complexity but for a given output, it is computationally infeasible to find the corresponding input. Also for a hash function, it is computationally infeasible to find two distinct inputs x_1 and x_2 such that $h(x_1) = h(x_2)$. The last property can be interpreted as a pseudo one-to-one input-output relationship, which means that for different inputs, the outputs will be different with high probability. In other words [15],

$$\forall x_1, x_2 \text{ if } x_1 \neq x_2 \overset{\text{with high probability}}{\Longrightarrow} f(x_1) \neq f(x_2). \tag{7}$$

Stronger hash functions have lower collision probability. Since hash function domain is bigger than its range, theoretically the collision probability is greater than zero. However, for a well designed hash function, finding collisions in polynomial time is practically impossible. Hence, with a good approximation, we can assume that (7) is always true. More details about hash function properties are available in [15].

Now suppose that $f(\cdot)$ is a hash function with outputs defined over $GF(q)$ and $a \in GF(q)$ is a uniformly distributed random symbol. We build the vector $\tilde{\boldsymbol{m}}$ as

$$\begin{aligned}\tilde{\boldsymbol{m}} = (&x_1 + f(a), \ x_2 + f(a, x_1), \ x_3 + f(a, x_1, x_2), \cdots, \\ &x_{n-1} + f(a, x_1, x_2, \cdots, x_{n-2}), a)^T,\end{aligned} \tag{8}$$

where the comma-separation of independent variables in the hash function denotes concatenation, which means the individual components are put next to each other in a bit-wise manner and then applied to the hash function as a single input. Since the arguments of the hash functions in (8) are different, and considering (7), we find that each component in (8) has a different noisy term (similar to one-time-pad stream ciphers [15]). So, this way, by using only one random symbol, we have concealed all the information symbols in such a way that the adversary cannot recover them via linear combination. The rest of the process is the same as current schemes: running a linear transformation on $\tilde{\boldsymbol{m}}$ and then sending the resultant vector through the network.

Note that in this scheme $f(\cdot)$ is known to all parties including the adversary, and we have not used any cryptographic process. Furthermore, there is no need for senders and receivers to do any kind of handshaking before starting data transmission. Since the receivers know $f(\cdot)$, after recovering $\tilde{\boldsymbol{m}}$ (feasibility of recovery depends on the design of the network code), all information symbols can be recovered by having a, which is the last component of $\tilde{\boldsymbol{m}}$.

The main reason for using hash function is to generate different random symbols out of the only noisy symbol, a. Although the statement (7) is probabilistic, as long as the probability of the event $f(a, x_1, \cdots, x_i) = f(a, x_1, \cdots, x_j)$ for any distinct i and j is not greater than the probability of having the same outcomes for two independent uniformly distributed random variables selected from $GF(q)$ (i.e., $\frac{1}{q}$), then the security is as strong as using $n-1$ independent uniformly distributed random symbols for $n-1$ information symbols in $\tilde{\boldsymbol{m}}$ (i.e. One-time-pad stream cipher). Typical hash functions and code fields satisfy this condition easily.

A. Discussion and Comparison with Existing Schemes

Since $\boldsymbol{B}_{k\times r}$ in (6) has to be full row rank, all its rows must be linearly independent. This requirement imposes some limitations in assigning coding vectors to the network edges and makes the design and management of network code more complex, especially for random linear network codes. In our proposed scheme, we use only one random symbol and then spread out the randomness evenly over the information vector so that each component in $\tilde{\boldsymbol{m}}$ has a noisy term. Therefore, for the adversary, neither any individual component of $\tilde{\boldsymbol{m}}$ nor any linear combination of them has meaningful information. As a result, there is virtually no limitation on the coding vector design. For example, we could even have coding vectors like $e_i = (0, \cdots, 0, 1, 0, \cdots, 0)$, which has only one nonzero component located at the ith position. However, assigning code vector $e_n = (0, \cdots, 0, 1)$ to any edge of the network should be avoided, since in this case, if $e_1 = (1, 0, \cdots, 0)$ is another coding vector, and if the corresponding channels of both of these two coding vectors are reached by the adversary, then x_1 can be detected; if e_2 as well as e_1 and e_n is a coding vector and acquired by the adversary, then x_2 can be detected, and so on. Therefore, it is recommended to avoid assigning e_n to any of the network edges and that would be the only restriction. Since each coding vector has n components, excluding the all-zero vector, roughly we have $q^n - 1$ possible ways for selecting coding vectors. Note that this is an upper bound for the coding vector space. With the proposed scheme, this bound will be reduced from $q^n - 1$ to $q^n - 2$. Considering that n and q have large values (e.g., $q = 2^8$ and $n = 10$), this reduction in the coding vector choices would be negligible.

One-way functions are also used in [16] to make a network code secure. The main idea in [16] is to obtain security based on the topology of the network. This scheme requires establishing a spanning tree throughout the network and each network node needs to distinguish its output links not included in the spanning tree from the one included. It assumes that there is only one receiver and the sender wants to transmit one information symbol in each time interval, and each node has a random number generator. For cases that the above scheme does not work properly, one-way functions with the same domain and range (i.e., both are defined over the same field) are suggested to generate random symbols. It is shown that by using such functions, security is achievable. Hash functions are used in our proposed scheme with a different goal: achieving complete security while minimizing information overhead in such a way that intermediate nodes do not need a random number generator and there are no restrictions on the number of receivers and no need for spanning trees.

V. Conclusion

We have presented a new approach to achieve complete security in networks with linear network coding. The basic idea of the proposed scheme is to introduce noisy terms using hash functions. Advantages of the proposed scheme include (a) sender nodes need to substitute *only one* information symbol with a uniformly distributed random symbol, which minimizes transmission overhead; (b) no changes are needed for all the coding and routing processes at intermediate nodes; (c) no cryptographic schemes are needed and no handshaking is required before data transmission starts; (d) greatly relaxed requirements on coding vector selection space; and (e) neither any linear combinations nor any individual component in the message vector gives the adversary any information. A disadvantage of proposed approach might be the use of hash functions, which increases the transmission complexity. However, an appropriately chosen, simple and efficient hash function will reduce the added complexity.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W.-H. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. IT-46, pp. 1204–1216, Apr. 2000.

[2] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, *Network Coding Theory*. Now Publishers Inc., 2006.

[3] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. IT-49, pp. 371–381, Feb. 2003.

[4] R. W.-H. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *IEEE Trans. Inf. Theory*, vol. IT-45, pp. 1111–1120, May 1999.

[5] N. Cai and R. W. Yeung, "Secure network coding," in *Proc. IEEE ISIT'02*, Lausanne, Switzerland, July 2002, p. 323.

[6] J. Feldman, T. Malkin, C. Stein, and R. A. Servedio, "On the capacity of secure network coding," in *Proc. 42nd Annual Allerton Conf. Commun., Control and Comput.*, Sep. 2004.

[7] J. Feldman, T. Malkin, C. Stein, and R. A. Servedio, "Secure network coding via filtered secret sharing," in *Proc. 42nd Annual Allerton Conf. Commun., Control and Comput.*, Sep. 2004.

[8] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," in *Proc. NETCOD'05*, Riva del Garda, Italy, Apr. 2005.

[9] Y. Zhang, C. Xu, and F. Wang, "A novel scheme for secure network coding using one-time pad," in *Proc. Int. Conf. Networks Security, Wireless Commun. and Trusted Comput.*, China, Apr. 2009, pp. 92–98.

[10] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Proc. ISIT'04*, June 2004, p. 143.

[11] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of Byzantine adversaries," *IEEE Trans. Inf. Theory*, vol. 54, pp. 2596–2603, June 2008.

[12] S. Y. El Rouayheb and E. Soljanin, "On wiretap networks II," in *Proc. ISIT'07*, Nice, France, June 2007, pp. 551–555.

[13] J. Dong, R. Curtmola, R. Sethi, and C. Nita-Rotaru, "Toward secure network coding in wireless networks: threats and challenges," in *Proc. IEEE NPSEC*, Orlando, Florida, Oct. 2008, pp. 33–38.

[14] M. Riemensberger, Y. E. Sagduyu, M. L. Honig, and W. Utschick, "Training overhead for decoding random linear network codes in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 729–737, June 2009.

[15] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, chs. 6 and 9. CRC Press, 1996.

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