This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

P-Coding: Secure Network Coding against Eavesdropping Attacks

Peng Zhang, Yixin Jiang, Chuang Lin
Department of Computer Science and Technology
Tsinghua University
Beijing, China, 100084
{zhangpeng08, yxjiang, clin}@csnet1.cs.tsinghua.edu.cn

Yanfei Fan, Xuemin (Sherman) Shen
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, N2L, 3G1, Canada
{yfan, xshen}@bbcr.uwaterloo.ca

***Abstract* — Though providing an intrinsic secrecy, network coding is still vulnerable to eavesdropping attacks, by which an adversary may compromise the confidentiality of message content. Existing studies mainly deal with eavesdroppers that can intercept a limited number of packets. However, real scenarios often consist of more capable adversaries, e.g., global eavesdroppers, which can defeat these techniques. In this paper, we propose P-Coding, a novel security scheme against eavesdropping attacks in network coding. With the lightweight permutation encryption performed on each message and its coding vector, P-Coding can efficiently thwart global eavesdroppers in a transparent way. Moreover, P-Coding is also featured in scalability and robustness, which enable it to be integrated into practical network coded systems. Security analysis and simulation results demonstrate the efficacy and efficiency of the P-Coding scheme.**

Keywords* — *Network coding; eavesdropping attacks; permutation encryption

I. INTRODUCTION

Network coding, as an alternative to traditional store-and-forward mechanism, allows intermediate nodes to code/mix incoming data flows. This novel information dissemination approach, first introduced in [1] to maximize the multicast throughput, is also proved able to achieve smaller transmission-delay [14], lower energy-consumption [15], and enhanced robustness [16] in a variety of communicating systems. Random Linear Network Coding (RLNC), in which participating nodes linearly combine incoming packets using randomly chosen coefficients, is verified to be both sufficient and efficient for network coding paradigms [2], [3]. Thanks to the employment of RLNC, network coding is evolving into a practical and promising data communication technology [4]. It has already found applications in P2P content distribution networks [17], wireless mesh networks [18], delay tolerant networks [19], etc.

Due to the unreliable multi-hop transmission and willful intermediate packet-mixing, network coding is also susceptible to various types of security threats, e.g., eavesdropping attacks, Byzantine modifications, traffic analysis attacks, etc. Among all these threats, eavesdropping attacks are especially concerned, as they could seriously impair the confidentiality of network coded systems. Due to its versatility, network coding may be readily utilized in scenarios where sensitive information is communicated. Examples include military networks and banking systems, where confidentiality is always the top security concern, and should be assured by all means. However, the novelty of network coding greatly challenges the efficacy of most security techniques for the traditional packet-forwarding systems. Moreover, the passive nature of eavesdropping makes it rather difficult to be detected compared to active attacks. Thus, how to thwart eavesdropping attacks in the new context of network coding, especially RLNC, is quite a pressing and challenging issue.

Most of the existing schemes deal with adversaries that could only intercept a limited number of packets: In [5]- [8], it is attempted to design network codes that are *Shannon secure* (see definition in Section II-C.1) at the cost of decreased throughput; Bhattad et al. [9] propose to achieve *weak security* (see definition in Section II-C.2) without any loss in system capacity, by performing transformations on source messages. It is fair to say that these works are mainly of theoretical nature, due to the ideal assumption on limited eavesdropping capability of the adversary. In practice, by monitoring more network links, or collaborating with its malicious peers, a highly motivated adversary may intercept enough packets to defeat these schemes. To thwart such more motivated and/or capable adversaries, we believe some cryptographic approaches should be applied.

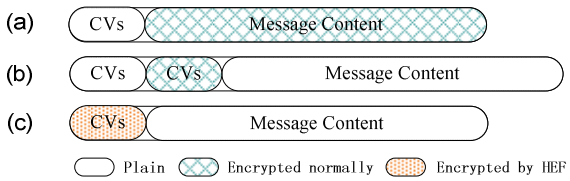

Fig. 1. Three cryptographic approaches for RLNC (CVs: Coding Vectors)

An intuitive approach based on cryptography is to employ link-to-link encryptions on coded packets. However, this method is not feasible as it will bring heavy computational overhead to each node, and result in significant performance degradation of the whole system. A modification is to perform encryptions on message content in an end-to-end manner, as shown in Fig. 1-(a). Though this method is much more efficient than the previous one, the requirement to encrypt the whole message content will require certain unnecessary computational overhead, as well.

The mixing feature of network coding allows us to assure confidentiality more efficiently by protecting the much shorter coding vector instead of the long message content. Vilela et al. [10] propose such a scheme, in which the set of coding vectors used at the source are encrypted, while another set of unencrypted ones are attached to maintain the standard coding processes at intermediate nodes, as shown in Fig. 1-(b). Ob-

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

viously, this scheme requires less data to be encrypted, but it actually needs two rounds of decoding processes, thus may not be as efficient as expected. Moreover, considerable space overhead will be incurred by the extra set of coding vectors.

In fact, the coding vectors could be more elegantly protected using Homomorphic Encryption Functions (HEFs) [21], needless of attaching an extra set of coding vectors, as shown in Fig. 1-(c). Due to the homomorphism of HEFs, random linear network coding could be performed directly on the encrypted coding vectors. However, the expensive HEF operations required at each participating node will certainly make this approach computational heavyweight and not scalable.

To sum the above three methods, they are inefficient in either computation or space, as the security feature provided by RLNC is not fully exploited. Considering that both the coding vectors and message content are necessary for packet-decoding, randomly mixing and reordering them together can bring considerable confusions to the adversary, thus provide an elegant security guarantee. Moreover, this holistic mixing and reordering can also hide coding vectors and bring some benefits to privacy preservation [11].

In this paper, we propose the P-Coding, a novel security scheme against eavesdropping attacks in network coding. Our objective is to efficiently achieve *computational security* (see definition in Section II-C.3) against global eavesdroppers for NC-based applications. In sum, P-Coding offers the following significant features: 1) **Security:** In P-Coding, it is rather computationally difficult for a global eavesdropper to recover any meaningful information. In addition, P-Coding also protects the coding vectors to resist flow tracing attacks, which clearly distinguishes itself from usual end-to-end encryptions. 2) **Efficiency**: The lightweight nature of permutation encryption makes P-Coding quite efficient to be carried out, which is of special benefit for resource-constraint systems, such as wireless sensor networks. 3) **Transparency**: P-Coding is transparent to intermediate coding processes due to the exchangeability of permutation encryption and symbol-level linear combination. 4) **Scalability**: With extra efforts required only at the source and sinks, the performance of P-Coding does not degrade with the increase of intermediate nodes. 5) **Robustness**: P-Coding could also be enhanced to address the single generation failure problem, which is to be defined later. This property enables its application in much hostile environments where the key may accidentally leak out.

As another important contribution, we also present a more extensive and accurate evaluation to verify the weak security [9] property of RLNC. Under some assumptions on the capability of the adversary and the size of finite field, we show that RLNC is inherently weakly secure with high probability. This fact may throw light on the benefits of coding randomly over a large finite field.

The remainder of this paper is organized as follows. Section II presents the system model, adversary model, and security levels, as foundations for the following discussions. Section III extensively evaluates the intrinsic security provided by RLNC. Section IV introduces the P-Coding scheme and its enhanced scheme, whose security and performance aspects are respectively studied in Section V. Section VI surveys some related works on secure network coding, followed by a brief conclusion in Section VII.

II. Problem Statement

A. System Model

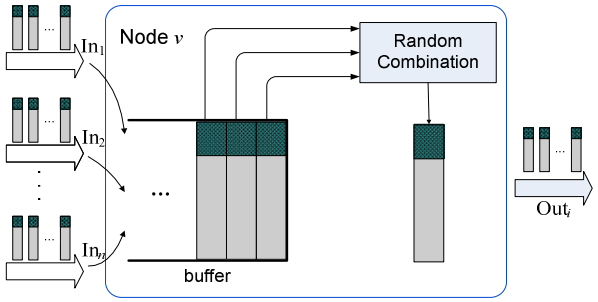

Fig. 2. Random linear network coding at intermediate nodes

We adopt the general random linear network coding model introduced in [4]. Formally, a network can be represented as an acyclic directed graph $G=(V,E)$. For each node $v\in V$, define the set of links terminating at it as $\Gamma^{-}(v)$, and the set of links originating from it as $\Gamma^{+}(v)$. For each link $e\in E$, it has the capacity of one packet per unit time, and let $y(e)$ be the packet carried on it. Here each packet is defined as a row vector $\boldsymbol{x}_i=[x_{i,1},...,x_{i,l}]$ of length l, defined on finite field F. Consider a general multicast case in which one source s needs to deliver a series of packets $\boldsymbol{x}_1,...,\boldsymbol{x}_h$ to a set of sinks, where h is the capacity of this multicast session.

For the sake of notations, we assume that $\Gamma^{-}(s)$ consists of h imaginary links, $e_1,...,e_h$, with $y(e_i)=\boldsymbol{x}_i$. Then for any $e\in\Gamma^{+}(v), v\notin T$, $y(e)$ is calculated by combining the incoming packets of v, linearly and on symbol-level:

$$y(e)=\sum_{e'\in\Gamma^{-}(v)}\beta_{e'}(e)y(e')=\boldsymbol{\beta}(e)Y(e') \qquad (1)$$

where the coefficients $\beta_{e'}$, randomly chosen from F, form the vector $\boldsymbol{\beta}(e)=[\beta_{e'}(e)]$ termed as the Local Encoding Vector (LEV). By induction, $y(e)$ could be represented as the linear combination of $\boldsymbol{x}_1,...,\boldsymbol{x}_h$:

$$y(e)=\sum_{i=1}^{h}g_i(e)\boldsymbol{x}_i=\boldsymbol{g}(e)X \qquad (2)$$

where $\boldsymbol{g}(e)=[g_1(e),...,g_h(e)]$, able to be calculated recursively using Eq. (1), is termed as the Global Encoding Vector (GEV). Assuming that h packets $y(e_1),\cdots,y(e_h)$ are received at a sink node v, we can apply Eq. (2) and write them in the matrix form:

$$Y=[y(e_1),\cdots,y(e_h)]^T=[\boldsymbol{g}(e_1),\cdots,\boldsymbol{g}(e_h)]^T X=GX \qquad (3)$$

If G, known as Global Encoding Matrix (GEM), is invertible, v can reconstruct source messages X by applying $X=G^{-1}Y$.

In reality, it is required that before transmission the source prefix each packet $\boldsymbol{x}_i$ with the i^{th} unit vector $\boldsymbol{u}_i$:

$$[\boldsymbol{u}_i,\boldsymbol{x}_i]=[\underbrace{0,..,0}_{i-1},1,\underbrace{0,..,0}_{h-i},x_{i,1},...,x_{i,l}] \qquad (4)$$

where each $\boldsymbol{u}_i$ is termed as a tag. With the same coding operations performed on the tag, we can achieve that each packet automatically contains its GEV, as Fig. 2 shows.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

B. Adversary Model

Informally, the adversary considered in this paper aims at intercepting packets and decoding them to harvest meaningful information. It could act as an external eavesdropper to monitor network links, and/or as an internal eavesdropper to compromise intermediate nodes and read their memories, as shown in Fig. 3. The primary difference of these two types of attack is that link-to-link cryptographic schemes capable of thwarting the former can be defeated by the latter. In this paper, we assume without loss of generality, that the source and sinks are always trusted and can never be compromised by an adversary.

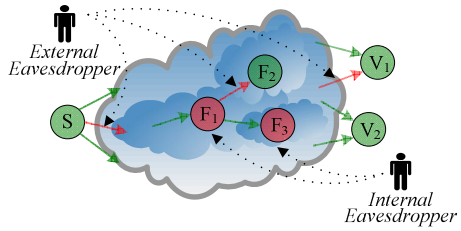

Fig. 3. The external and internal eavesdropper

Before giving the formal definition of adversary, let us first formalize the attack it could launch. For any eavesdropping attack W, let $E' \in E$ denote the set of links being monitored, and $V' \in V$ denote the set of nodes being compromised. Then W can be defined by the set of packets intercepted by the adversary:

$$W = \{y(e) : e \in E' \cup \Gamma_v^- \cup \Gamma_v^+, \forall v \in V'\} \quad (5)$$

We assume the adversary could launch any attack belonging to the set $A=\{W_i\}$ (i.e., access any packets in some $W_i \in A$). Let $\boldsymbol{W_i}$ be a matrix whose rows contain all linearly independent GEVs of packets in W_i. Let k_i be the number of rows in $\boldsymbol{W_i}$. The integer $k = \max_i k_i$ is defined as *capability* of the adversary. We say the adversary is *k-capable* if it has capability k, and we also say it is *global* if $k = h$.

C. Security Levels

With the adversary model given above, we could define three different levels of security for network coding systems:

1) *Shannon Security* [24]: The system is said to be Shannon secure or perfectly secure, if the adversary can not get any information about the source messages X from the intercepted packets, which could be formulated as:

$$H(X \mid W_i) = H(X), \quad \forall W_i \in A \quad (6)$$

2) *Weak security* [9]: If no *meaningful* information about the source messages X can be derived from the intercepted packets by adversary, the system is said to be weakly secure. This concept could be formally represented as:

$$H(\boldsymbol{x_i} \mid W_i) = H(\boldsymbol{x_i}), \quad \forall \boldsymbol{x_i} \in X; \forall W_i \in A \quad (7)$$

Remarks: The difference between Shannon security and weak security can be illustrated using the following example: Suppose the eavesdropper has intercepted $a \oplus b$, where a and b are two bits from some source. Then he can get one bit of information about the source, but this information is not meaningful. This system is weakly secure, but not Shannon secure.

3) *Computational Security* [20]: Computational security is based on the assumption that the adversary is resource-bounded. It is satisfied if the amount of effort to recover any meaningful information about $\forall \boldsymbol{x_i} \in X$ using the best currently-known methods exceeds computational resources of the adversary.

In the following of this paper, we will not consider Shannon security, as it is only achievable under ideal assumption that the adversary is limited in capability. Moreover, the cost to achieve Shannon security may well offset the throughput benefit provided by network coding. As for weak security, we will show that given the filed size is sufficiently large and the adversary less than h-capable, RLNC is inherently weakly secure. Computational security, however, is our main focus, as it can assume the global adversary model. Moreover, it can be achieved in real practice, by utilizing cryptographic approaches.

III. Intrinsic Security of RLNC

In this section, we will demonstrate the weak security property provided by RLNC, by giving two theorems. The first one shows that under some assumptions, RLNC is inherently weakly secure with high probability; while the second considers the smart adversary who can guess some combinations of the source messages.

We consider the random linear network coding model, where coding coefficients are chosen randomly from finite field F of size q, and assume that the adversary has capability $k<h$.

Theorem 1: For sufficiently large value of q, the probability that the adversary cannot get any meaningful information can be approximated as:

$$P_{ws}(k) = \prod_{i=1}^{k} (1 - hq^{i-h} + hq^{i-h-1}) \quad (8)$$

Proof: Consider the adversary launches an attack $W_i \in A$. There are two necessary conditions for the adversary not to get any meaningful information: 1) k_i is less than the number h of source messages; otherwise all information could be recovered; 2) No unit vector could be derived by performing Gaussian eliminations on $\boldsymbol{W_i}$, since a unit vector indicates that one message could be directly obtained. Since $k<h$, we have $k_i \le k < h$, meaning that condition (1) holds. Then we only need to evaluate the probability that condition (2) holds under the strongest adversary assumption, i.e., $k_i = k$.

Let $M(i, j)$, $0 \le i \le j$, denote the number of all $(i+j)\times h$ matrices $\boldsymbol{\Lambda}$ defined on F satisfying: a) The first i rows of $\boldsymbol{\Lambda}$ are linearly independent and already fixed; b) No unit vector could be derived by performing Gaussian eliminations on $\boldsymbol{\Lambda}$. Then it easily follows that $M(0,k)$ is the number of all $k\times h$ matrices satisfying condition (2).

From analysis of linear dependence, we get the following recurrence relation about $M(i, j)$:

$$M(i, j) = q^i \cdot M(i, j-1) + (N_{i+1} - q^i) \cdot M(i+1, j) \quad (9)$$

where $N_i = q^h - q^{i-1}(q-1)h$ is the number of choices for the i^{th} row satisfying that no unit vector could be derived, given that the first $i-1$ rows are linearly independent.

As q is sufficiently large, we have $M(i, j) \approx N_{i+1} M(i+1, j)$ which leads to the approximation:

$$M(0,k) \approx \prod_{i=1}^{k} N_i = [q^h - (q-1)h] \cdots [q^h - q^{k-1}(q-1)h] \quad (10)$$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

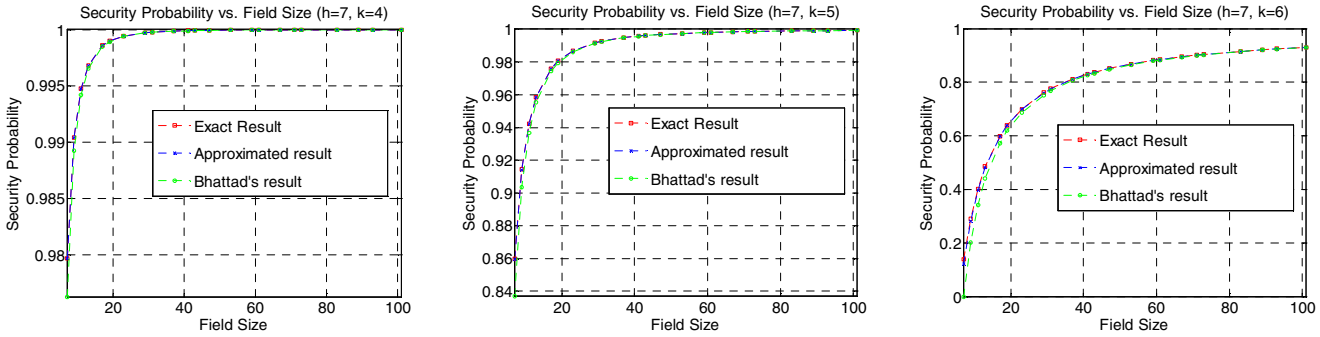

Fig. 4. Security probability vs. field size: a) h=7, k=4; b) h=7, k=5; c) h=7, k=6

The probability that condition (2) holds is then calculated as:

$$P_{ws}(k)=\frac{M(0,k)}{q^{hk}}\approx\prod_{i=1}^{k}(1-hq^{i-h}+hq^{i-h-1}) \qquad (11)$$

The theorem is thus proven. ■

Eq. (11) is a more accurate expression compared to the result given by Bhattad et al. [9]. We can see that if the size of finite field is sufficiently large, the probability can be made arbitrarily high. Moreover, the rate of convergence decreases drastically as the capability of adversary increases. These facts can be seen in Fig. 4.

Next we consider a more general case where a smart adversary can accurately guess some linear combinations of source messages. The adversary is also allowed to choose the linear coefficients for the combinations. In a successful case for the adversary, it can solve more than g messages with only g guesses. We then evaluate the probability for RLNC to resist this guessing threat.

Theorem 2: The probability that the smart adversary can only solve g messages by g guesses is:

$$P_{gws}(k,g)=1-|\cup_{1\le i\le h-g}G_i|/q^{hk},\quad (1\le g<h-k) \qquad (12)$$

where $G_i=\{\boldsymbol{W}_i:\exists\{r_1,...,r_i\}\subseteq(span(I_1,...,I_{g+i})\cap span(\boldsymbol{W}_i))\}$, I_j is a unit row vector of dimension h, and $I_i\neq I_j$ for $i\neq j$.

Proof: From the analysis of linear dependence, we know that the smart adversary could recover $g+i\le h$ messages by g guesses if and only if the matrix $\boldsymbol{W_i}$ contains i row vectors that are linear combinations of some $g+i$ different unit row vectors. If we define the set of all such $\boldsymbol{W_i}$ as G_i, then the probability that only g messages can be recovered is calculated by eliminating the probability contributed by all G_i, $(1\le i\le h-g)$. ■

IV. P-CODING: THE PROPOSED SCHEME

This section presents the concept of permutation encryption, based on which we propose the P-Coding scheme to thwart eavesdropping attacks. Following that an enhanced scheme is introduced to enhance the robustness of P-Coding.

A. Permutation Encryption

We formalize the concept of permutation encryption as a special case of the classic *transposition cipher* [20].

Notations: We term a sequence π containing each element of set $\{1,2,...,n\}$ once and only once as a *permutation with length n*. Let $\pi(i)$ be the i^{th} element of π, then the *product* of two permutations π_1 and π_2, defined by $\pi_1\circ\pi_2$, or $\pi_1\pi_2$ is calculated using $\pi_1\pi_2(i)=\pi_1(\pi_2(i))$. Let π^{-1} be the inverse of π with respect to product operation.

Definition 1: Let $\boldsymbol{m}=[m_1,m_2,...,m_n]$ be a sequence of symbols from a finite field F, and k be a permutation with length n, then the *Permutation Encryption Function* (*PEF*) on $\boldsymbol{m}$ using key k is defined as:

$$E_k(\boldsymbol{m})=E_k([m_1,m_2,...,m_n])=[m_{k(1)},m_{k(2)},...,m_{k(n)}] \qquad (13)$$

Correspondingly, the *Permutation Decryption Function* on $\boldsymbol{c}$ using key k is defined as:

$$D_k(\boldsymbol{c})=D_k([c_1,c_2,...,c_n])=[c_{k^{-1}(1)},c_{k^{-1}(2)},...,c_{k^{-1}(n)}] \qquad (14)$$

Property: Since the permutation encryption only involves reordering the sequences to be encrypted, without altering any of their symbols, it is exchangeable with the symbol-level linear operations over finite field F, as shown below:

1) **Addition:** $E_k(\boldsymbol{m}+\boldsymbol{n})=E_k(\boldsymbol{m})+E_k(\boldsymbol{n})$

2) **Scalar Multiplication**: $E_k(t\cdot\boldsymbol{m})=t\cdot E_k(\boldsymbol{m})$

To utilize the permutation encryption in real applications, we observe that two technical issues should be considered:

1) The plaintext $\boldsymbol{m}$ must be protected; otherwise it is easy to deduce the key k by correlating it with the ciphertext $\boldsymbol{c}$. This fact could be formalized as: $I(K,M\,|\,C)=0$, where $I(\cdot,\cdot\,|\,\cdot)$ denotes conditional mutual information.

2) The encryption key should be chosen randomly, which is intuitively necessary for PEFs to be effective. One possible approach is given in Algorithm 1.

Algorithm 1. Random Key Generation

```
00: Function Random_Key_Generation ( n )
01:     for each i∈[1,n]   perm(i)←i;   /* Initialization */
02:     for each i∈[1,n-1]   /* Key generation loop */
03:         select a random integer r uniformly distributed over [i,n];
04:         perm(i) ↔ perm(r);
05:     return perm;
```

B. The P-Coding Scheme

The basic idea of P-Coding is to perform the permutation encryption on coded messages, as shown in Fig. 5. After PEF operations, symbols of the messages and corresponding GEVs can be mixed and reordered together. We will show in Section

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

V that such holistic encryption operations could bring considerable confusions to the adversary.

The P-Coding scheme primarily consists of three stages: source encoding, intermediate recoding, and sink decoding. Without loss of generality, we assume that there is a Key Distribution Center (KDC) responsible for symmetric key establishment. Then the source and sinks can acquire the PEF key k offline.

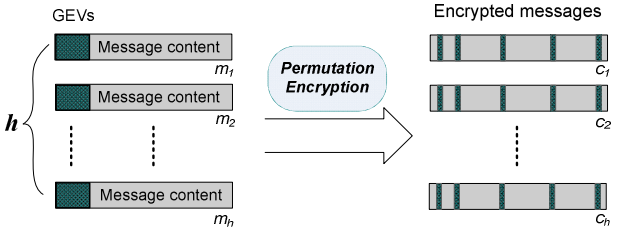

Fig. 5. Permutation encryption on coded messages

Source Encoding: Consider the situation that a source s has h messages, say $\boldsymbol{x}_1,...,\boldsymbol{x}_h$, to be sent out. It first prefixes these h messages with their corresponding unit vectors, according to Eq. (4). Then the source performs linear combinations on these messages with randomly chosen LEVs. For instance, with LEV $\boldsymbol{\beta}(e_i)$, we could get the coded message $y(e_i)=[\boldsymbol{\beta}(e_i),\boldsymbol{\beta}(e_i)X]$, where $X=[\boldsymbol{x}_1,...,\boldsymbol{x}_h]^T$. Finally, the source performs permutation encryption on each message $y(e_i)$ to get its ciphertext $c[y(e_i)]=E_k[y(e_i)]$.

Intermediate Recoding: Since the symbols of messages and corresponding GEVs are rearrange via PEF, and the intermediate nodes have no knowledge of the key being used, it is rather difficult for them to reconstruct source messages. On the other hand, as permutation encryptions are exchangeable with symbol-level linear combinations, intermediate recoding can be transparently performed on the encrypted messages:

$$\begin{aligned} c[y(e)] &= c[\textstyle\sum_{e'\in\Gamma^-(v)}\beta_{e'}(e)\cdot y(e')] \\ &= \textstyle\sum_{e'\in\Gamma^-(v)}\beta_{e'}(e)\cdot c[y(e')] \end{aligned} \tag{15}$$

Evidently, this transparency property can significantly improve the system efficiency, since no extra effort is needed at any intermediate node. In addition, it can also facilitate the real deployment of the P-Coding scheme in network coding based applications.

Sink Decoding: For each sink node, on receiving a message $c[y(e_i)]$ from its incoming link $e_i\in\Gamma^-(v)$, it decrypts the message by performing permutation decryption on it:

$$D_k\{c[y(e_i)]\}=E_{k^{-1}}\{E_k[y(e_i)]\}=y(e_i) \tag{16}$$

Once h linearly independent messages $y(e_1),...,y(e_h)$ are collected, the sink derives the following matrix representation:

$$Y=\begin{bmatrix} y(e_1)\\ \vdots \\ y(e_h)\end{bmatrix}=\begin{bmatrix} \boldsymbol{g}(e_1),\boldsymbol{g}(e_1)X\\ \vdots \\ \boldsymbol{g}(e_h),\boldsymbol{g}(e_h)X\end{bmatrix}=[G,GX] \tag{17}$$

Finally, the source messages could be recovered by applying Gaussian eliminations on Y:

$$Y=[G,GX]\xrightarrow[\text{elminations}]{\text{Guassian}}[I,X] \tag{18}$$

C. The Enhanced P-Coding Scheme

In practical network coding scenarios, such as distributed content distribution, the source may need to transmit a large volume of data D. In this case, the source should first divide D into generations:

$$D=[\underbrace{\boldsymbol{x}_1,...,\boldsymbol{x}_h}_{G_1},\underbrace{\boldsymbol{x}_{h+1},...,\boldsymbol{x}_{2h}}_{G_2},...,\underbrace{\boldsymbol{x}_{(n-1)h+1},...,\boldsymbol{x}_{nh}}_{G_i},...]$$

Then D is sent as a stream of generations, with network coding only performed among messages belonging to the same generation. In P-Coding, if the same key is used throughout the transmission, the problem of single generation failure may happen, in which an accidental key disclosure in one generation will compromise the secrecy of the following transmission.

We address this problem by randomly perturbing the key used in each generation. Let k_i denote the key used in the i^{th} generation, then k_i is calculated by $k_i=\omega_i\circ k_{i-1}$, where ω_i is a random permutation with length n, termed as the *perturbing key*. If ω_i is changed each generation and only shared by the source and sinks, this approach could effectively prevent the single generation failure.

Discussion: This approach will inevitably incur some space overhead as the perturbing key should be transmitted in each generation. One possible implementation is to prefix each packet of the i^{th} generation with the ciphertext of ω_i. This will incur 100% space overhead if no extra measure is taken, clearly not feasible. In the following part, we will study how to make this approach more efficient.

Definition 2: Suppose π is a permutation with length n, if $\pi(i)=i$ holds for each $i\notin[s,s+m-1]\subseteq[1,n]$, we say that π is *m-partial*.

For a partial permutation, some elements of it are in their original positions. It is easy to see that an m-partial permutation with length n can be represented by an integer $s\in[0,n-m+1]$ and a permutation with length m. Thus, we could decrease the length of key to m, by using an m-partial permutation as the perturbing key.

Next, we consider compressing the m-partial permutation to an integer $d\in[0,m!-1]$ for efficient transmission. To achieve this goal, we must find a one-to-one correspondence between integers and permutations, so that given an integer it is efficient to calculate the corresponding permutation. We will show that Proposition 1 and Proposition 2 together can do the job.

Proposition 1: There is a one-to-one correspondence between integers $n\in[0,m!-1]$ and sequences $A_{m-1}=[a_1,...,a_{m-1}]$, where $a_i\in[0,i]$.

Proof: First rewrite $m!-1$ in the following form:

$$\begin{aligned} m!-1 &= (m-1)(m-1)!+(m-1)!-1 \\ &= (m-1)(m-1)!+(m-2)(m-2)!+...+2\cdot 2!+2!-1 \\ &= (m-1)(m-1)!+(m-2)(m-2)!+...+2\cdot 2!+1\cdot 1! \end{aligned}$$

Then any $n\in[0,m!-1]$ could be uniquely represented as:

$$\mathrm{n}=a_{m-1}(m-1)!+a_{m-2}(m-2)!+...+a_2\cdot 2!+a_1\cdot 1!\ \ (a_i\in[0,i])$$

where a_i is calculated using three formulas: $a_i=n_i\%(i+1)$, $n_{i+1}=n_i/(i+1)$, and $n_1=n$. ■

Proposition 2: There is a one-to-one correspondence between sequences $A_{m-1}=[a_1,...,a_{m-1}]$, where $a_i\in[0,i]$, and permutations with length m.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Proof: For each $A_{m-1}=[a_1,...,a_{m-1}]$, we could construct a corresponding sequence $B_{m-1}=[b_1,...,b_{m-1}]$ using $b_i=m-a_{m-i}$. As $b_i\in[i,m]$ follows from $a_i\in[0,i]$, we could obtain a unique permutation with length *n*, by replacing the random integer in the i^{th} loop of Algorithm 1 with b_i. Similarly, it is easy to verify that the reverse construction is also unique, thus a one-to-one correspondence is established. ■

Based on the above two propositions, we give Algorithm 2, which aims to perturb the key using five parameters, of which *k* is the current key to be perturbed; *n* and *m* are initially agreed upon by the source and sinks; *s* and *d*, already defined above, are chosen randomly from their respective domains to represent the perturbing key.

Algorithm 2 Key Perturbing

```
00: Function Key_Perturbing ( k, n, m, s, d )
01:   for each i∈[1,m-1]   /* to generate the sequence [a_1,...,a_{m-1}] */
02:     a(i)←d % (i+1);   d←d / (i+1);
03:   for each i∈[1,m-1]   /* to generate the sequence [b_1,...,b_{m-1}] */
04:     b(i)←m−a(m−i);
05:   for each i∈[1,n]   /* Initialization */
06:     π(i)←i;
07:   for each i∈[1,m-1]   /* to calculte the m-partial permutation */
08:     π(s−1+i)↔π(s−1+b(i));
09:   for each i∈[1,n]   /* to perturb the current key k using π */
10:     k̃(i)←π(k(i));
11:   return k̃(i);
```

In real implementation, we require that in each generation, the source node encrypt the key pair (*s*, *d*) of this generation using symmetric keys shared between itself and the corresponding sink. This symmetric key infrastructure can be established at the start of system. The implementation will not be difficult, and we would not go into details in this paper due to the limitation of space.

V. Security Analysis and Performance Evaluation

In the following discussion, we still assume each generation contains *h* source messages of length *l*. According to our system model, each of the coded messages has length *n*=*h*+*l*. These notations will be used in this section without extra explanations.

A. Probability Model of Permutation Encryption

In this model, we represent the sequence to be encrypted as a vector $M=[M(1),...,M(n)]$, where each $M(i)$ is a random variable distributed over finite field *F*. Similarly, we represent the key and corresponding ciphertext as $K=[K(1),...,K(n)]$ and $C=[M(K(1)),...,M(K(n))]$, respectively. For the sake of notations, define equivalent events $\{K=\boldsymbol{k}\}=\bigcap_{i=1}^{n}\{K(i)=k(i)\}$, $\{M=\boldsymbol{x}\}=\bigcap_{i=1}^{n}\{M(i)=x(i)\}$, and $\{C=\boldsymbol{x}\}=\bigcap_{i=1}^{n}\{C(i)=x(i)\}$. To simplify our analysis, assume the size of field is sufficiently large, so that the sequence *M* will not include duplicate symbols. As $|F|\to\infty$, we have $P(M(I)=M(J))=P(I=J)$, where *I* and *J* are random variables distributed over $[1,n]$.

Definition 3: We say a permutation encryption is *forward random*, or has the property of *forward randomness*, if and only if it satisfies:

$$P(\bigcap_{i=1}^{n}\{C(i)=x_{k(i)}\}\mid M=\boldsymbol{x})=1/n!,\quad(\forall k,\forall\boldsymbol{x})\qquad(19)$$

Similarly, we say a permutation is *backward random*, or has the property of *backward randomness*, if and only if it satisfies:

$$P(\bigcap_{i=1}^{n}\{M(i)=x_{k(i)}\}\mid C=\boldsymbol{x})=1/n!,\quad(\forall k,\forall\boldsymbol{x})\qquad(20)$$

Of these two random properties of permutation encryption, backward randomness property means that the plaintext *could have been* any possible order/sequence of the ciphertext with equal probability. This fact could make the cryptanalysis on permutation encryption degrade into exhaustive search, which promises a very strong security for permutation encryption. In the following, we will give sufficient conditions for the property of forward and backward randomness, respectively.

Theorem 3: A sufficient condition for the permutation encryption to be have forward randomness is: $P(K=\boldsymbol{k})=1/n!$ and K is distributed independent of M.

Proof: $P(\bigcap_{i=1}^{n}\{C(i)=x_{k(i)}\}\mid M=\boldsymbol{x})$

$$=\frac{P(\bigcap_{i=1}^{n}\{M(K(k^{-1}(i)))=M(i)\},M=\boldsymbol{x})}{P(M=\boldsymbol{x})}$$

$$=\frac{P(\bigcap_{i=1}^{n}\{K(k^{-1}(i))=i\},M=\boldsymbol{x})}{P(M=\boldsymbol{x})}\quad(|F|\to\infty)$$

$$=\frac{P(\bigcap_{i=1}^{n}\{K(k^{-1}(i))=i\})P(M=\boldsymbol{x})}{P(M=\boldsymbol{x})}\quad(\textit{Independence})$$

$$=P(K=\boldsymbol{k})=1/n!$$

The theorem can be proven according to Definition 3. ■

Theorem 4: A sufficient condition for the permutation encryption to have backward randomness is: the permutation encryption has forward randomness, and each $M(i)\in M$ is independently and uniformly distributed.

Proof: $P(\bigcap_{i=1}^{n}\{M(i)=x_{k(i)}\}\mid C=\boldsymbol{x})$

$$=\frac{P(C=\boldsymbol{x}\mid\bigcap_{i=1}^{n}\{M(i)=x_{k(i)}\})P(\bigcap_{i=1}^{n}\{M(i)=x_{k(i)}\})}{\sum_{\pi}P(C=\boldsymbol{x}\mid\bigcap_{i=1}^{n}\{M(i)=x_{\pi(i)}\})P(\bigcap_{i=1}^{n}\{M(i)=x_{\pi(i)}\})}$$

$$=\frac{P(\bigcap_{i=1}^{n}\{C(i)=x'_{k^{-1}(i)}\}\mid M=\boldsymbol{x}')P(\bigcap_{i=1}^{n}\{M(i)=x_{k(i)}\})}{\sum_{\pi}P(\bigcap_{i=1}^{n}\{C(i)=x''_{\pi^{-1}(i)}\}\mid M=\boldsymbol{x}'')P(\bigcap_{i=1}^{n}\{M(i)=x_{\pi(i)}\})}$$

$$=\frac{(1/n!)P(\bigcap_{i=1}^{n}\{M(i)=x_{k(i)}\})}{\sum_{\pi}(1/n!)P(\bigcap_{i=1}^{n}\{M(i)=x_{\pi(i)}\})}\quad(\textit{Theorem }3)$$

$$=\frac{\prod_{i=1}^{n}P(M(i)=x_{k(i)})}{\sum_{\pi}\prod_{i=1}^{n}P(M(i)=x_{\pi(i)})}\quad(\textit{Independece})$$

$$=|F|^{n}/(n!\cdot|F|^{n})=1/n!$$

The theorem can be proven according to Definition 3. ■

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

B. *Security Analysis: The P-Coding Scheme*

We will demonstrate that the P-Coding scheme provides a high level of computational security, by showing that to defeat it the adversary has to carry out exhaustive search, which is rather difficult in computation.

First, we demonstrate the necessity of exhaustive search by verifying that the permutation encryption in P-Coding has the property of backward randomness:

1) In P-Coding, the encryption key k is generated randomly and uniformly using Algorithm 1, and chosen independently of the message to be encrypted. Thus the encryption has forward randomness according to Theorem 3.

2) Each packet in network coding undergoes rounds of random linear combinations, thus it is fair to assume that the dependence among its symbols has been eliminated and the distribution tends to be uniform. As we have already shown the encryption is forward random, it is also backward random according to Theorem 4.

Secondly, we need to consider some existing cryptanalysis techniques, and demonstrate their infeasibility to defeat the permutation encryption utilized in our P-Coding scheme.

1) *Traditional Cryptanalysis*: Traditional cryptanalysis approaches such as differential analysis, linear analysis, etc, are based on either known-plaintext or selected-plaintext. As we assume that the source and sinks will not be compromised, the plaintexts cannot be accessed by the adversary. Thus these approaches can not defeat our P-Coding scheme.

2) *Optimization Heuristics*: As a relatively new innovation, optimization heuristics, including genetic algorithms [22], simulated annealing [23], etc, are believed to be the best currently-known approaches to break the classic transposition ciphers. They achieve their goals by optimizing some fitness functions which are designed to assess the suitability of a given key. For details of this approach, readers could refer to [22].

Though capable of breaking transposition ciphers on written languages, optimizations heuristics are not applicable to our permutation encryptions on coded messages, for two reasons: First, there are no n-gram statistics in coded messages, since random linear mixing will make their elements approximately independent. Secondly, these approaches assume the continuity of fitness functions, so that it could search for better suitable keys in the neighborhood of the current one. In P-Coding, however, a little change in the PEF key will result in different GEVs, which will decode messages into quite different content.

Finally, we evaluate the cost of exhaustive search. Assume each generation contains h message, and each coded message has length n. To carry out exhaustive search, the adversary needs to try $\boldsymbol{O}(n!)$ rounds to guess the plaintext. In each round, it should test its guess by performing Gaussian eliminations according to Eq. (18), whose computational complexity is $\boldsymbol{O}(h^3)$ in terms of multiplication operations. In sum, the computational complexity for exhaustive search is $\boldsymbol{O}(n!\cdot h^3)$.

Table. I. Magnitude of some factorials

n	30	40	50	60
$n!$	2.65×10^{32}	8.16×10^{47}	3.04×10^{64}	8.32×10^{81}

Table. I sketches the difficulty of carrying out exhaustive search, by giving some values of factorials. More specifically, let us consider a normal case n=50: If the adversary could guess and test as many as 10^{20} possible plaintexts per second, the average time to obtain the right one is:

$$T = \frac{50!/2}{365\times24\times3600\times10^{20}} = 4.8221\times10^{36} \text{ (year)}$$

In addition, P-Coding can also efficiently thwart the flow tracing attacks, one of the most severe traffic analysis approaches to compromise the privacy of networks, as shown in Fig. 6. Here we consider three primary flow tracing methods: size correlation, time-order correlation, and message content correlation. As RLNC has messages trimmed into equal size, and buffered at intermediate nodes [4], it could inherently resist the first two methods. To launch message content correlation, the adversary must intercept packets of the same generation and determine if an intercepted packet in some downstream link is a linear combination of some known packets.

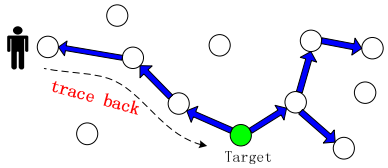

Fig. 6. An example of flow tracing attack

According to our previous work [11], the computational complexity is $\boldsymbol{O}(h^3+h\cdot l)$ in terms of multiplication operations, and if some anonymous routing protocol is already in place to protect the generation number, this complexity will increase to $\boldsymbol{O}(C_{wh}^{h}(h^3+h\cdot l))$, where w is the total number of generations. Compared to the existing network coding with explicit GEVs, P-Coding could significantly increase the difficulty of flow tracing, and thus effectively preserve the privacy of networks. This feature remarkably distinguishes P-Coding from the intuitive end-to-end encryption scheme.

C. *Security Analysis: The Enhanced P-Coding Scheme*

Evidently, if the key does not leak out in any generation, which indicates the normal case, the security level of enhanced scheme is as high as that of the P-Coding scheme. When single generation failure occurs, the enhanced scheme can provide two extra properties.

1) **Security**: After the compromise of security in current generation, the security level in following ones will be strong enough to thwart further attacks. We show this by evaluating the computational complexity for the adversary to guess the next PEF key based on the current one. First, it should first locate the start point of key perturbing operation, which has $\boldsymbol{O}(n)$ different choices. Then it should fix the correct sequence of the perturbed section of key, which has $\boldsymbol{O}(m!)$ different choices. It is fair to assume that these choices are equally possible, according to the randomness property of permutation encryption. Finally, it should decode the messages by performing Eq. (18), which costs $\boldsymbol{O}(h^3)$ multiplication operations. Thus, the computational complexity is $\boldsymbol{O}(n\cdot m!\cdot h^3)$ in terms of multiplication operations,

2) **Recovery**: As the PEF key is perturbed randomly and incrementally, it will become more and more irrelevant to its original form with iterations of generations. Consequently, even if the current key is disclosed, its randomness to the adversary will gradually recover after some generations. The numerical result is given in Theorem 5.

Theorem 5: After i generations, the expected number of all perturbed symbols (distinct) in the PEF key is approximately:

$$EX_i = \frac{i-1}{i+1}\cdot(n-m)\cdot[1-(1-\frac{m}{n-m})^{i+1}]+m \quad (n\to\infty) \quad (21)$$

Proof: See the Appendix. ■

Fig. 7 shows that the number of perturbed symbols in PEF key increases with the number of generations, meaning that its randomness will gradually recover after accidental disclosure. Moreover, the closeness between asymptotical analysis and simulative result well justifies our approximation.

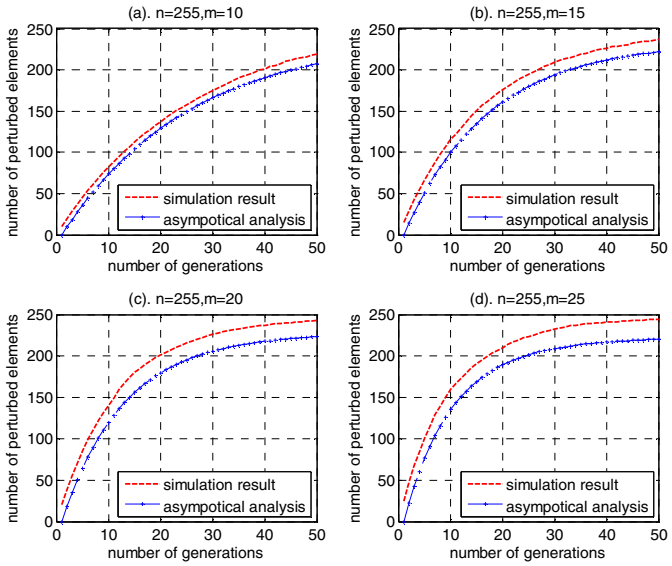

Fig. 7. Number of perturbed symbols vs. number of generations: a) n=255, m=10; b) n=255, m=15; c) n=255, m=20; d) n=255, m=25

D. Performance Evaluations

The P-Coding Scheme: As the PEF key could be acquired offline, the only computational overhead of P-Coding comes from the permutation encryption operations at the source and decryptions operations at sinks. According to Eq. (13) and Eq. (14), the encryption and decryption processes only involve reordering the symbols of messages, thus require $\boldsymbol{O}(n)$ memory copy operations. As there are h messages in each generation, the computational overhead is then $\boldsymbol{O}(n\cdot h)$ in terms of memory copy operations. Considering the inherent overhead of RLNC is at lease $\boldsymbol{O}(h^3)$ in terms of multiplication operations, due to the necessity of Gaussian eliminations, P-Coding is quite light-weight in computation. In addition, P-Coding does not cause any space overhead, as well.

The Enhanced P-Coding Scheme: In the enhanced scheme, the source should generate two integers s and d to represent the perturbing key in each generation. It is fair to assume that the generation of these two integers could be done within constant time. So it is the same with the encryption and decryption of them. As the computational complexity of key perturbing processes is $\boldsymbol{O}(n)$ according to Algorithm 2, the extra computational overhead incurred by the enhanced scheme is just $\boldsymbol{O}(n)$.

E. Comparisons

As a retrospect, in the following we compare P-Coding with the other three schemes mentioned at the beginning of this paper, and then present the results in Table. II.

First we claim that though all these schemes can provide some degree of confidentiality, only P-Coding and scheme (c) can achieve privacy, since (a) and (b) fail to protect the explicit GEVs, which provide backdoors for traffic analysis. Then, we compare their respective computational overhead in terms of multiplications. As P-Coding achieves security by performing only memory copy operations, whose cost is negligible compared to that of multiplication, it is reasonable to approximate its computational cost to be zero. On the other hand, the other three schemes will incur nonnegligible overhead. Specifically, the overhead of scheme (a) can be derived straightforward, while that of (b) and (c) has been derived in [10] and [11], respectively. Moreover, (b) also incurs space overhead since it inserts a duplicate GEV of length h in each packet of length n.

From Table. II, it is fair to conclude that our P-Coding scheme outperforms the other three candidates on thwarting eavesdropping attacks in network coding.

Table. II. Comparisons among P-Coding and (a)-(c) in Section I

	P-Coding	(a)	(b)	(c)
Confidentiality	√	√	√	√
Privacy against Flow Tracing	√	×	×	√
Computational Overhead	≈ 0	$\boldsymbol{O}(h\cdot l)$	$\boldsymbol{O}(h^3)$	$\boldsymbol{O}(h^3 logn)$
Space Overhead	*None*	*None*	$h/(n+h)$	*None*

VI. Related Works

On securing network coding against eavesdropping attacks, several valuable works have appeared, which could be divided into three categories according to security levels they provide.

Some of these works attempt to provide *Shannon security* by designing linear secure codes for network coding. Cai et al. [5] first identify the "wire-tap" adversary who can monitor a limited number of links, and present an approach to transform any linear network code to be secure in the presence of this kind of adversary. The same problem is further treated in [6], where the authors generalize the method used in [5] and find a tradeoff between the multicast capacity and the field size. An interesting observation made in [6] is that making network coding secure is equivalent to finding codes with some distance properties. Rouayheb et al. [7] formalize the problem as a generalization of Wiretap Channel II model, and propose a secure scheme by implementing cosset coding at the source. They show that this coding scheme could be implemented without affecting the underlying network coding architecture. This model is further studied by Silva et al. in [8], where another source coding scheme, named Maximum Rank Distance (MRD), is designed to replace that used in [7]. Both [7] and [8] stress the fact that the designing of secure codes and the optimizing of network transport could be treated independently.

As a generalization of Shannon security, Bhattad et al. [9] innovatively introduce the concept of *weak security*, by which the system is said to be secure if the adversary can not recover any meaningful information. They show that under this relaxed security requirement, the multicast capacity could be achieved

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

by performing linear transformations at the source.

Computational security, as another relaxation to Shannon security, receives relatively less attention. Existing schemes to provide computational security include the lightweight security mechanism proposed by Vilela et al. [10] and the homomorphic encryption technique contributed by Fan et al. [11]. Details of these two schemes have already been covered in Section I.

Though without giving practical schemes, some works still contribute to our understanding of this security issue. Jain et al. [12] give a necessary and sufficient condition for the unicast in cyclic networks to be secure, by exploiting the topological feature of communicating systems; Lima et al. [13] consider the threats posed by "nice but curious" intermediate nodes and develop an algebraic security criterion to access the inherent security provided by the RLNC paradigm.

VII. CONCLUSION

In this paper, we first formalized an adversary model which incorporates both the external and internal eavesdroppers in Random Linear Network Coding (RLNC). Then we presented a more extensive and accurate evaluation for the weak security provided by RLNC. Since this security is unreliable for practical scenarios consisting of global eavesdroppers, we proposed the P-Coding scheme. With analysis and simulation, we demonstrated that it can efficiently thwart global eavesdropping attacks in a transparent way.

ACKNOWLEDGEMENT

This work is supported by the National Grand Fundamental Research 973 Program of China (No. 2010CB328105), and the National Natural Science Foundation of China (No. 60970101).

REFERENCES

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

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

[3] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. on Information Theory*, vol. 52, no. 10, pp. 4413-4430, Oct. 2006.

[4] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," *Proc. of 41st Allerton Conference on Communication, Control and Computing*, 2003.

[5] N. Cai and R. W. Yeung, "Secure network coding," *Proc. of the IEEE International Symposium on Information Theory (ISIT)*, pp. 323, 2002.

[6] J. Feldman, T. Malkin, C. Stein, and R. A Servedio, "On the capacity of secure network coding," *Proc. of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, pp. 30-39, Sept. 2004.

[7] S. Y. E. Rouayheb and E. Soljanin, "On wiretap networks II," *Proc. of IEEE ISIT*, pp. 551–555, Jun. 2007.

[8] D. Silva and F. R, "Security for wiretap networks via rank-metric codes," *Proc. of IEEE ISIT*, pp. 176-180, Jul. 2008.

[9] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," *Proc. of the First Workshop on Network Coding, Theory, & Applications*, 2005.

[10] J. P. Vilela, L. Lima, J. Barros, "Lightweight security for network coding," *Proc. of IEEE ICC'08*, pp. 1750-1754, May. 2008.

[11] Y. Fan, Y. Jiang, H. Zhu, and X. Shen "An efficient privacy-preserving scheme against traffic analysis in network coding," *Proc. of IEEE INFOCOM'09*, pp. 2213-2221, Apr. 2009.

[12] K. Jain, "Security based on network topology against the wiretapping attack," *IEEE Wireless Communications*, pp. 68-71, Feb. 2004.

[13] L. Lima, M. Médard, and J. Barros, "Random linear network coding: A free cypher?" *Proc. of IEEE ISIT*, pp. 546–550, Jun. 2007.

[14] P. A. Chou and Y. Wu. "Network coding for the internet and wireless networks," *MSR-TR-2007-70,* Microsoft Research, Jun. 2007.

[15] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile Ad hoc networks using network coding," *IEEE Trans. on Communications*, vol. 53, no. 11, pp. 1906-1918, Nov. 2005.

[16] D. Lun, M. Médard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," *Proc. of IEEE ISIT*, pp. 1848–1852, Sept. 2005.

[17] C. Gkantsidis and P. Rodriguez, "Network coding for large scale file distribution," *Proc. of IEEE INFOCOM'*05, pp. 2235-2245, Mar. 2005.

[18] S. Katti, H. Rahul, D. Katabi, W. Hu, M. Médard, and J. Crowcroft. "Xors in the air: practical wireless network coding," *IEEE/ACM Trans. on Networking*, vol. 16, no. 3, pp. 497-510, Jun. 2008.

[19] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," *Proc. of the 2005 ACM SIGCOMM Workshop on Delay-tolerant networking*, pp. 284-251, Aug. 2005.

[20] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Oct. 1996.

[21] J. Benaloh, "Dense probabilistic encryption," *Proc. of the Workshop on Selected Areas in Cryptography*, pp. 120-128, Aug. 1994.

[22] A. Dimovski and D. Gligoroski, "Attacks on the transposition ciphers using optimization heuristics," *Proc. of the International Scientific Conf. on Information, Comm. & Energy Systems & Technologies*, Oct. 2003.

[23] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. "Optimization by simulated annealing," *Science,* vol. 220, no. 4598, pp. 671-680, 1983.

[24] C. E. Shannon, "*Communication theory of secrecy systems*," *Bell Systems Technical Journal*, vol. 28, pp. 656-715, Oct. 1949.

APPENDIX: PROOF OF THEOREM 5

Proof: Suppose there is a segment L with length n. In each round, a randomly chosen segment with length $m < n$ in L is colored. Let X_i be the total length which has been colored after i rounds, then we are supposed to evaluate EX_i.

As *n* is sufficiently large, each start point of the i^{th} colored segments, defined by S_i satisfies an independent continuous uniform distribution over $(0,\lambda)$, where $\lambda = n - m$. Define a series of random variables δ_j as $\delta_0 = S_{(1)}$, $\delta_1 = S_{(2)} - S_{(1)}$, ..., $\delta_{i-1} = S_{(i)} - S_{(i-1)}$, where $S_{(j)}$ is the j^{th} order statistics of $S_1,...,S_i$. Then it follows that $\delta_0, \delta_1, ..., \delta_{i-1}$ are identically distributed. Define another series of random variables $\tilde{\delta}_j, (1 \le j \le i-1)$ as: $\tilde{\delta}_j = \delta_j$ if $\delta_j < m$; $\tilde{\delta}_j = m$ if $\delta_j \ge m$. It is evident that $\tilde{\delta}_0, \tilde{\delta}_1, ..., \tilde{\delta}_{i-1}$ are also identically distributed. Actually, $\tilde{\delta}_j$ is the length of segment being colored from start point $S_{(j)}$, without overlapping with that form $S_{(j+1)}$ (if there is). Thus we have $EX_i = \sum_{j=1}^{i-1} E\tilde{\delta}_j + m = (i-1)E\tilde{\delta}_0 + m$. As the probability density function of δ_0 is $f(x) = \frac{i}{\lambda}(1-\frac{x}{\lambda})^{i-1} I_{(x \le \lambda)}$, we also have:

$$E\tilde{\delta}_0 = \int_0^m x d\,\mathrm{P}(\tilde{\delta}_0 \le x) = \int_0^m x f(x)dx + m\int_m^\lambda f(x)dx$$

$$= \int_0^m (1-\frac{x}{\lambda})^i dx = \frac{\lambda}{i+1}[1-(1-\frac{m}{\lambda})^{i+1}]$$

Finally, the expectation of X_i can be calculated as:

$$EX_i = \frac{i-1}{i+1} \cdot \lambda \cdot [1-(1-\frac{m}{\lambda})^{i+1}] + m$$

Eq. (21) can be obtained by substituting λ with $n - m$. ■