A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks

Peng Zhang, Chuang Lin, *Senior Member, IEEE,* Yixin Jiang, Yanfei Fan, and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—Energy saving is an important issue in Mobile Ad Hoc Networks (MANETs). Recent studies show that network coding can help reduce the energy consumption in MANETs by using less transmissions. However, apart from transmission cost, there are other sources of energy consumption, e.g., data encryption/decryption. In this paper, we study how to leverage network coding to reduce the energy consumed by data encryption in MANETs. It is interesting that network coding has a nice property of intrinsic security, based on which encryption can be done quite efficiently. To this end, we propose P-Coding, a lightweight encryption scheme to provide confidentiality for network-coded MANETs in an energy-efficient way. The basic idea of P-Coding is to let the source randomly permutes the symbols of each packet (which is prefixed with its coding vector), before performing network coding operations. Without knowing the permutation, eavesdroppers cannot locate coding vectors for correct decoding, and thus cannot obtain any meaningful information. We demonstrate that due to its lightweight nature, P-Coding incurs minimal energy consumption compared to other encryption schemes.

Index Terms—Mobile ad hoc networks, energy saving, network coding, lightweight encryption.

1 Introduction

Mobile Ad Hoc Networks (MANETs) are important wireless communication paradigms. The mobile and infrastructureless nature of MANETs makes them suitable for collecting emergency data in disastrous areas and performing mission-critical communication in battle fields. A critical issue in MANETs is how to reduce energy consumption and maintain a longer life time for mobile nodes. Several energy-efficient schemes are proposed to resolve this issue [2]–[4].

Recent studies demonstrate that network coding [5] can help achieve a lower energy consumption in MANETs [6]–[8]. The energy saving comes from fact that less transmissions are required when in-network nodes are enabled to encode packets. The basic idea can be illustrated using the following example. Suppose there are six nodes forming a hexagon, and the transmission range of each node can only reach its left and right neighbor. Each node needs to broadcast one message to all other nodes. Without network coding, each message would require four broadcasts, as shown in Fig. 1(1). With network coding (Fig. 1(2)-(4)), a total number of nine transmissions are needed for three messages, i.e., three transmissions per message. If we would not consider the energy consumed by encoding and decoding operations, this means $1/4$ energy can be saved.

Besides basic transmissions, energy consumption can also come from encryption and decryption operations at each node, as most MANETs need some level of protection on their content. For example, in a battle field, the data communicated between soldiers with mobile devices can be very sensitive, and should be kept confidential during transmissions. The straightforward approach to provide confidentiality for network-coded MANETs is to encrypt the packet payload using symmetric-key encryption algorithms. While this method is not that efficient: reference [9] shows that on a Motorola's "DragonBall" embedded microprocessor, it consumes around $13.9\mu J$ to send a bit, while consumes another $7.9\mu J$ per bit when symmetric-key algorithms are used. In fact, the information mixing feature of network coding provides

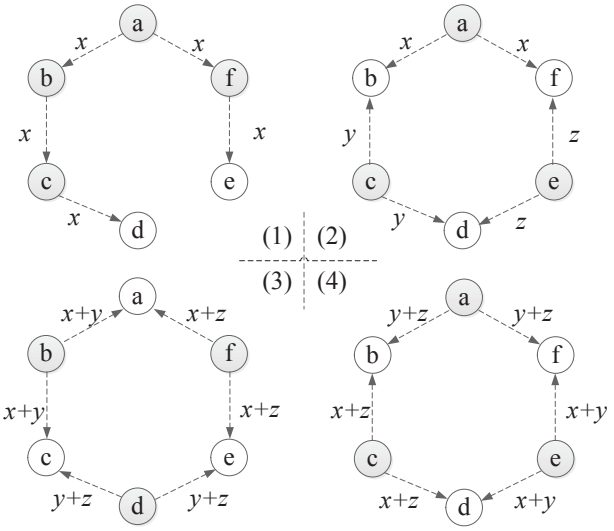


Fig. 1: Example illustrating how network coding reduces transmission times in MANETs. The shaded nodes are those involved in transmissions.

- *P. Zhang, C. Lin, and Y. Jiang are with Dept. of Computer Science and Technology, Tsinghua University, Beijing, China. Email:{pzhang, clin, yxjiang}@csnet1.cs.tsinghua.edu.cn*
- *Y. Fan and X. Shen are with Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada. Email: {yfan, xshen}@bbcr.uwaterloo.ca*

Part of this work was presented in IEEE INFOCOM'10 [1].

Digital Object Indentifier 10.1109/TPDS.2013.161

an intrinsic security, based on which a more efficient cryptographic scheme can be designed. Vilela et al. [10] propose such a scheme, in which the source performs random linear coding on the messages to be sent and locks/encrypts the coding vectors using the symmetric key shared between it and all sinks. Fan et al. [11] propose to encrypt coding vectors using Homomorphic Encryption Functions (HEFs) in an end-to-end manner. Due to the homomorphic nature of HEFs, network coding can be performed directly on the encrypted coding vectors, without impacting the standard network coding operations. However, the above two approaches have large overhead with respect to either computation or space, and may not be suitable for MANETs.

In this paper, we attempt to design a new encryption scheme that can fully exploit the security property of network coding. *Since both the coding vectors and message content are necessary for decoding, randomly reordering/mixing them will generate considerable confusion to the eavesdropping adversary.* In specific, we propose P-Coding, a lightweight encryption scheme to fight against eavesdroppers in network-coded MANETs. In a nutshell, P-Coding randomly mixes symbols of each coded packet (packet prefixed with its coding vector) using *permutation encryption*, to make it hard for eavesdroppers to locate coding vectors for packet decoding.

Our contribution is two-fold: (1) we propose a new encryption scheme which is lightweight in computation by leveraging network coding, which makes it very attractive in network-coded MANETs to further reduce energy consumption; and (2) we present an analysis on the intrinsic weak security provided by network coding, which is more accurate than [12]. We show that network coding is inherently weakly-secure with high probability, when the coding vectors are randomly chosen over a large finite field.

The remainder of this paper is organized as follows. Section 2 presents the system model and security model. Section 3 evaluates the intrinsic security provided by network coding. Section 4 introduces the P-Coding scheme and its enhanced version, and their security is analyzed in Section 5. Section 6 evaluates the performance of P-Coding with analysis and experiments. 7 surveys some related works on secure network coding, followed by a conclusion in Section 8.

2 Problem Statement

2.1 System Model

We consider a typical MANET consisting of N nodes, each of which can be a source. The MANET can be modeled as an acyclic directed graph $G = (V, E)$. For each node $v \in V$, there is a link from v to u if u is within v's transmission range. Let $\Gamma^-(v)$ be the set of links terminating at v, and $\Gamma^+(v)$ be the set of links originating from v. We assume that each link $e \in E$ has the capacity of one packet per unit time, and $y(e)$ is the packet carried on it. Here a packet is defined as a row vector of l elements from finite field $\mathbb{F}_q$. We also assume that linear network coding is enabled in this network. To illustrate how network coding works, let us consider the case that one node s needs to deliver a series of packets $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_h$ to a set of sinks $T \subset V$. Define the matrix of source packets as $X = [\boldsymbol{x}_i^T, \ldots, \boldsymbol{x}_h^T]^T$, i.e., X consists of all source packets as its rows. For simplicity, let $\Gamma^-(s)$ consists of h imaginary links, $\tilde{e}_1, \ldots, \tilde{e}_h$, with $y(\tilde{e}_i) = \boldsymbol{x_i}$. Then for any $e \in \Gamma^+(v), v \notin T$, $y(e)$ is calculated by linearly combining the incoming packets of v as:

$$y(e) = \sum_{e' \in \Gamma^-(v)} \beta_{e'}(e) y(e') = \boldsymbol{\beta}(e)[y^T(e')]^T_{e' \in \Gamma^-(v)} \quad (1)$$

where the coefficients $\beta_{e'}$ are chosen over $\mathbb{F}_q$, and the row vector $\boldsymbol{\beta}(e) = [\beta_{e'}]_{e' \in \Gamma^-(v)}$ is termed as the *Local Encoding Vector (LEV)* of link e. By induction, $y(e)$ can be represented as the linear combination of source packets:

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

where $\boldsymbol{g}(e) = [g_1(e), \ldots, g_h(e)]$ can be calculated recursively using Eq. (1), and is termed as the *Global Encoding Vector (GEV)* of link e. Assume that h packets $y(e_1), \ldots, y(e_h)$ are received by a sink node v from links $e_1, \ldots, e_h$. Then, by applying Eq. (2), we have:

$$Y = \begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}(e_1) \\ \vdots \\ \boldsymbol{g}(e_h) \end{bmatrix} X = GX \quad (3)$$

where G is termed as the *Global Encoding Matrix (GEM)* of node v. Since G is invertible with high probability when q is sufficiently large [13], v can reconstruct source messages X by calculating $X = G^{-1}Y$.

In practice, the source prefixes each packet x_i with the ith unit vector $\boldsymbol{u}_i$:

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

where each $\boldsymbol{u}_i$ is termed as a *tag*. With the same coding operations performed on these tags, each packet will automatically contain its GEV.

2.2 Security Model

Informally, the adversary considered in this paper aims at intercepting packets and decoding them to harvest meaningful information. It can act as an *external eavesdropper* to monitor network links, and/or as an *internal eavesdropper* to compromise intermediate nodes and read their memories. For any eavesdropping attack W, let $E' \subset E$ denote the set of links being monitored, and $V' \subset V$ denote the set of nodes being compromised. We characterize the attack W as the set of packets intercepted by the adversary:

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

Then, an adversary can be defined as a set $A = \{W_i\}$, i.e., it can launch any attack belonging to 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 $\boldsymbol{W}_i$. Then the *capability* of the adversary can be defined as $k = \max_i k_i$. We say A is *k-capable* if it has capability k, and we say A is *global* if $k = h$. With the adversary model given above, we consider the following three different levels of security for network-coded systems:

1) *Shannon Security* [14]: The system is said to be Shannon secure (perfectly secure), if the adversary cannot get any information about the source messages X from the intercepted packets, which can be formulated as:

$$H(X|W_i) = H(X), \forall W_i \in A \tag{6}$$

2) *Weak security* [12]: If no meaningful information about the source messages X can be derived from the packets intercepted by adversary, the system is said to be weakly secure, which can be formally stated as:

$$H(\boldsymbol{x}_i|W_i) = H(\boldsymbol{x}_i), \forall \boldsymbol{x}_i \in X; \forall W_i \in A \tag{7}$$

The difference between Shannon security and weak security can be illustrated using the following simple example: Suppose the eavesdropper Eav has intercepted one bit $a \oplus b$, where a and b are two i.i.d bits from the source. Then Eav obtains one bit of information about a and b, and the system is clearly not Shannon secure. However, Eav cannot recover either a or b, i.e., no meaningful information is leaked about either a or b. The system is said to be weakly secure.

3) *Computational Security* [15]: 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.

Remarks: In this paper, we will not consider Shannon security, as it is only achievable under ideal assumption that the adversary can only monitor a limited number of links [16]. In other words, Shannon security cannot be achieved when there are global eavesdroppers. As for weak security, we will show that given that the finite field size is sufficiently large and the adversary is less than h-capable, network coding is inherently weakly secure with high probability. Computational security will be our main focus, as it can be achieved using cryptographic approaches.

3 Intrinsic Security of Network Coding

In this section, we will demonstrate the weak security property of network coding through two theorems. The first one states that under certain assumptions, network coding is inherently weakly secure with high probability; while the second considers the smart adversary which can guess some combinations of the source messages.

We consider the Random Linear Network Coding (RLNC) model [13], where coding coefficients (i.e., elements of LEVs) are chosen randomly from finite field $\mathbb{F}_q$.

Theorem 1. For sufficiently large value of q, the probability that the adversary of capability $k < h$ will not get any meaningful information can be approximated as:

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

Proof: See Appendix A. □

We show our approximate result (i.e., Theorem 1) and that of Bhattad's [12] in Fig. 2. For comparison, we also include the accurate $P_{ws}(k)$ using the value of $M(0, k)$ calculated by Eq. (18). From Fig. 2, we can see that *if the size of finite field is sufficiently large, the probability of weak security can be made arbitrarily high*. Moreover, our approximate result is closer to the accurate one, compared to the result given by Bhattad's.

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 network coding to resist this guessing threat.

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

$$P_{gws}(k, g) = 1 - |\cup_{1 \le t \le h-g} G_t|/q^{hk}, (1 \le g < h - k) \tag{9}$$

where $G_t = \{\boldsymbol{W}_i : \exists\{r_1, \ldots, r_t\} \subset (span(I_1, \ldots, I_{g+t}) \cap span(\boldsymbol{W}_i))\}$, each $I_i, (1 \le i \le h)$ is a unit row vector of dimension h, and $I_i \neq I_j$ for $i \neq j$.

Proof: See Appendix B. □

4 P-Coding: The Proposed Scheme

This section defines *permutation encryption*, based on which we introduce P-Coding, a lightweight encryption scheme. Then, we introduce an enhanced scheme to further improve the security of P-Coding.

4.1 Permutation Encryption

We formalize the concept of permutation encryption as a special case of the classic transposition cipher [15].

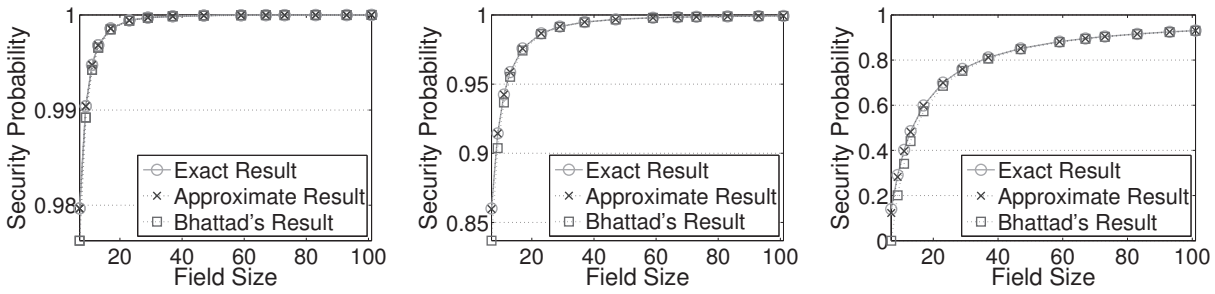
Notations: We term a sequence π containing each element of set $1, \ldots, n$ once and only once as a *permutation with length* n. Let $\pi(i)$ be the ith element of π, then the product of two permutations π_1 and π_2, defined by $\pi_1 \circ \pi_2$, or $\pi_1\pi_2$ is calculated using $\pi_1\pi_2(i)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

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

$$E_k(\boldsymbol{m}) = [m_{k(1)}, m_{k(2)}, \ldots, m_{k(n)}] \tag{10}$$

Similarly, we can define the *Permutation Decryption Function* on $\boldsymbol{c}$ using key k as $D_k(\boldsymbol{c})$, satisfying $D_k(E_k(\boldsymbol{m})) = \boldsymbol{m}$. Here, the permutation k is termed as the PEF key.

Remarks: Note that permutation encryption is quite simple and vulnerable to cryptographic analysis [17].



(a) $h = 7, k = 4$. (b) $h = 7, k = 5$. (c) $h = 7, k = 6$.

Fig. 2: Security probability vs. field size.

However, we try to use it on top of network coding to generate considerable confusion to the adversary. It may work in the context of network coding, as packets in network coding are linear combinations of original packets. To decode it, we need GEVs. Randomly permuting the packet symbols can make the eavesdropper unable to locate the GEVs and thus fail to decode the packets. Detailed proof will be given in Section 5.

4.2 The P-Coding Scheme

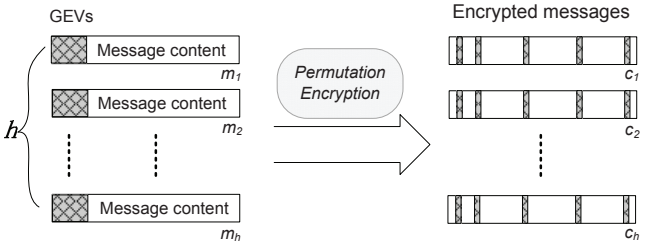


Fig. 3: Permutation encryption on coded messages.

The basic idea of P-Coding is to perform permutation encryptions on coded messages, as shown in Fig. 3. After PEF operations, symbols of the messages and corresponding GEVs can be mixed and reordered together. We will show in Section 5 that such PEF operations can generate 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, so that the source and sinks can share a PEF key k at the bootstrap stage of P-Coding.

Source Encoding: Consider the situation that a source s has h messages, denoted by column vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_h$, 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 $\boldsymbol{\beta}(e_i)$ of output link e_i, we can get the coded message $y(e_i) = [\boldsymbol{\beta}(e_i), \boldsymbol{\beta}(e_i)X]$, where $X = [\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_h^T]^T$. 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 rearranged 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 linear combinations, intermediate recoding can be transparently performed on the encrypted messages:

$$c[y(e_i)] = c[\sum_{e' \in \Gamma^-(v)} \beta_{e'}(e) y(e')] = \sum_{e' \in \Gamma^-(v)} \beta_{e'}(e) c[y(e')]$$

Note that this transparency property makes P-Coding rather efficient, since no extra effort is needed at any intermediate node.

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:

$$D_k\{c[y(e_i)]\} = E_k^{-1}\{E_k[y(e_i)]\} = y(e_i) \tag{11}$$

Once h linearly independent messages $y(e_1), \ldots, y(e_h)$ are collected, the sink derives the following matrix representation similar to 3:

$$Y = \begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}(e_1), \boldsymbol{g}(e_1)X \\ \vdots \\ \boldsymbol{g}(e_h), \boldsymbol{g}(e_h)X \end{bmatrix} = [G, GX] \tag{12}$$

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

$$Y = [G, GX] \xrightarrow[elimination]{Guassian} [I, X] \tag{13}$$

4.3 The Enhanced P-Coding Scheme

In practical network coding applications (e.g., distributed content distribution [18]), the source may need to transmit a large volume of data D. In this case, the source should first divide D into generations:

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

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 PEF key is used throughout the transmission, single generation failure may occur, 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*. More specifically, for each generation G_i, let the PEF key be used in G_i as k_i. Before each generation of data transmission, the source S conducts the following three steps: (1) S chooses a random permutation ω_i of length n, termed as *the perturbing key*; (2) S updates k_i, using the equation $k_i = \omega_i \circ k_{i-1}$, where $\circ$ denotes the product of two permutations; (3) S encrypts ω_i using another cryptographic approach (e.g., AES [19]), and sends the ciphertext of ω_i to all sinks who can similarly update k_i.

If the perturbing key ω_i is randomly chosen each generation and communicated securely between the source and sinks, this scheme can effectively prevent the single generation failure. However, the scheme will also 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 ith generation with the ciphertext of ω_i. Considering that each perturbing key is of length n, the same with a tagged packet, this scheme will incur 100% space overhead if no extra measure is taken, clearly not feasible. In the following, we will show how to make this scheme 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 can be seen 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 can decrease the length of the 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, 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. Therefore, we introduce the following proposition.

Proposition 1. There is a one-to-one correspondence between integers $n \in [0, m!-1]$ and permutations π with length m.

Proof: From basic combinatorics, any $n \in [0, m!-1]$ can be uniquely represented as:

$$n = a_{m-1}(m-1)! + a_{m-2}(m-2)! + , \ldots , + a_1 \cdot 1! \quad (14)$$

where $a_i \in [0, i]$ can be calculated using two recursive formulas: $a_i = n_i \% (i+1)$ and $n_{i+1} = \lfloor n_i/(i+1) \rfloor$, with initial condition $n_1 = n$. Construct a sequence $b_1, \ldots, b_{m-1}$ from $a_1, \ldots, a_{m-1}$, with $b_i = m - a_{m-i}$, and we have $b_i \in [i, m]$. Define a permutation $\omega = (1, 2, \ldots, m)$, and perform m rounds of operations: in the ith round, exchange the elements of $\omega(i)$ and $\omega(b_i)$. Then, the resultant ω is the corresponding permutation of length m. Since the above construction is a one-to-one correspondence, the proposition is proven. □

Based on this proposition, we propose Algorithm 1, which aims to perturb the key using five parameters, of which k is the current PEF key; n denotes the length of the tagged packet; m denotes the partiality of the perturbing key; s and d are chosen randomly from their respective domains to represent the perturbing key.

Algorithm 1: Key Pertrubing Function

Input: a permutation k of length n, integers n, m, s, d with $1 \le m \le n$, $s \in [0, n-m+1]$ and $d \in [0, m!-1]$
Output: a perturbed permutation $\tilde{k}$ of length n

```
// to generate the sequence (a_1, ..., a_{m-1})
foreach i ∈ [1, m-1] do
    a(i) ← d%(i+1) ;
    d = ⌊d/i+1⌋ ;
end
// to generate the sequence (b_1, ..., b_{m-1})
foreach i ∈ [1, m-1] do
    b(i) ← m − a(m−i) ;
end
// Initialization
foreach i ∈ [1, n] do
    ω(i) ← i ;
end
// to calculate the partial permutation
foreach i ∈ [1, m-1] do
    ω(s−1+i) ↔ ω(s−1+b(i)) ;
end
// to perturb the current key k using ω
foreach i ∈ [1, n] do
    k̃(i) ← ω(k(i)) ;
end
return k̃ ;
```

In the enhanced P-Coding scheme, we can employ symmetric encryptions to secure the transmission of perturbing key (s, d) from the source to sinks. Another possible approach is to let the source and sinks share a common Pseudo Random Number Generator (PRNG), so that the perturbing key (s, d) can be generated by the source and sinks in a distributed manner.

5 Security Analysis

In this section, we will analyze the security property of the proposed P-Coding scheme, both theoretically and with experimental validation. We will show the condition for permutation encryption to be secure based on

elementary probability models, and demonstrate that P-Coding provides a relatively high level of confidentiality. The experimental validation confirms our analysis that P-Coding is much more secure than the naive transposition cipher.

5.1 Theoretical Model

We represent the message to be encrypted as a random vector $M = [M(1), \ldots, M(n)]$ over finite field $\mathbb{F}_q$. Similarly, we represent the PEF key and corresponding ciphertext as $K = [K(1), \ldots, K(n)]$ and $C = [C(1), \ldots, C(n)]$, respectively. For sake of notations, we define equivalent events $\{K = k\} = \bigcap_{i=1}^{n}\{K(i) = k(i)\}$, $\{M = \boldsymbol{x}\} = \bigcap_{i=1}^{n}\{M(i) = x_i\}$, and $\{C = \boldsymbol{x}\} = \bigcap_{i=1}^{n}\{C(i) = x_i\}$. To simplify our analysis, assume the field size q is sufficiently large, so that the sequence M will not include duplicate symbols, i.e., $P(M(I) = M(J)) = P(I = J)$, 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:

$$P(\bigcap_{i=1}^{n}\{C(i) = x_{k(i)}\}|M = \boldsymbol{x}) = 1/n!, \ (\forall k, \forall \boldsymbol{x}) \qquad (15)$$

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

$$P(\bigcap_{i=1}^{n}\{M(i) = x_{k(i)}\}|C = \boldsymbol{x}) = 1/n!, \ (\forall k, \forall \boldsymbol{x}) \qquad (16)$$

Of these two random properties of permutation encryption, backward randomness means that the plaintext *could have been* any possible order/sequence of the ciphertext with equal probability. This can 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: $P(K = k) = 1/n!$ for each k, and K is distributed independent of M.

Proof: See Appendix C. □

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: See Appendix D. □

Is P-Coding backward random? We claim that P-Coding is forward random since the PEF key k is generated randomly and uniformly, and chosen independently of the messages to be encrypted. Then, packet in network coding undergoes rounds of random linear combinations, thus the dependence among its elements has been largely eliminated and the distribution tends to be uniform. This fact makes P-Coding backward random to some extent according to Theorem 4.

Exhaustive search is rather expensive. Recall that each generation contains h messages, and each message has length n. To carry out exhaustive search, the adversary needs to try $\boldsymbol{O}(n!)$ rounds to guess the plaintext or PEF key. In each round, it should test its guess by performing Gaussian eliminations according to Eq. (13), with computational complexity to be $\boldsymbol{O}(h^3)$ in terms of multiplication operations. Therefore, the computational complexity for exhaustive search is $\boldsymbol{O}(n! \cdot h^3)$.

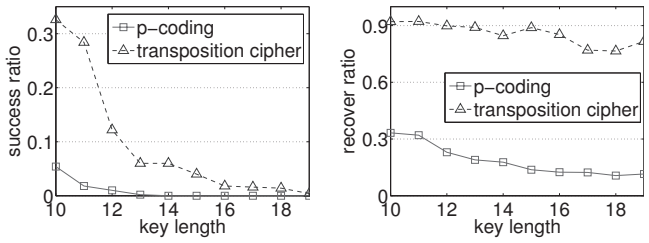
5.2 Experimental Validation

We consider a typical cryptanalysis on transposition cipher, and evaluate its effectiveness in breaking P-Coding. This cryptanalysis is based on the non-uniform frequencies of n-letter combinations, known as n-grams [20], in natural languages. For example, bigram 'TH' has a much higher frequency than bigram 'QZ' in English. Using frequency statistics of n-grams, the fitness of a guessed permutation p can be easily accessed: first decrypt a large number of ciphertexts by permuting them with the inverse of p, and then evaluate how close the n-gram statistics of the decrypted messages are to those of the underlying language. After that, by searching in the neighborhood of ps' with good fitness, we are expected to find other permutations with better fitness. This searching process continues until the key k is finally found. Recent studies show that optimization heuristics, e.g., Genetic Algorithms [17], Simulated Annealing [21], and Ant Colony [22], can be used to automate this searching process.

Though the above cryptanalysis is quite effective in breaking transposition ciphers, we argue that it does not work well for P-Coding, for the following reasons. (1) In P-Coding, before we can access the fitness of a permutation p, we should first decrypt generations of packets with p, and then decode them with the GEVs. The decoding process requires $\boldsymbol{O}(nh^2l)$ multiplications, where n is the number of generations, h is the generation size, and l is the length of a message. Thus, it is much more time-consuming to access the fitness of p in P-Coding than in transposition cipher. (2) Even a small change in the permutation p, say an exchange of two positions, will result in different GEVs, which may decode messages into quite different content. This means that even p has a good fitness, we cannot expect to find permutations with better fitness by searching in p's neighborhood.

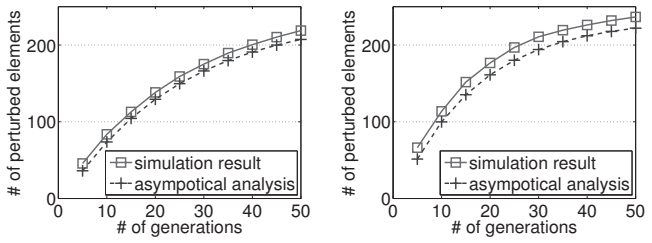
To justify the above argument, we implement the genetic algorithm proposed in [17], and evaluate its feasibility to break our P-Coding scheme (please refer to Appendix F for the algorithm). For comparison, we also include the performance of the algorithm to defeat traditional transposition ciphers. The metrics to compare include: (1) *Success Ratio*, the ratio of the number of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.



(a) Success ratio (b) Recover ratio

Fig. 4: The effectiveness of Genetic algorithm on transposition cipher and P-Coding.



(a) $n = 255, m = 10$. (b) $n = 255, m = 15$.

Fig. 5: Number of perturbed positions vs. number of generations.

rounds in which the key is recovered, to the total number of rounds, and (2) *Recovery Ratio*, the ratio of the average number of recovered positions of the key, to the total length of the key.

In our simulation, we chose a readable passage of length 1000 (in words), divide it into multiple messages, and perform P-Coding and transposition encryption on them, respectively. Note that the primary difference between P-Coding scheme and transposition ciphers is the former permutes messages after they are randomly and linearly coded, while the latter permute messages directly. For the parameters of genetic algorithm, we set the group size to 12, and the maximum round of mating and mutation to 100. We experiment by varying the length of permutation key from 10 to 19. For each case, we run the genetic algorithm 500 rounds for both P-Coding and transposition cipher. We report the results in Fig. 4.

From Fig. 4(a), we can see that genetic algorithm is effective to break transposition ciphers on passage we chose, particularly when the key length is 10. For longer key length, the success attack ratio stays above 0, meaning that this attack is till feasible. However, with P-Coding, the permutation encryption is rather resistant to this attack. This is justified by observing that the probability of successful attack becomes 0 when the key length increase to 14.

Fig. 4(b) shows that even the genetic algorithm succeeds in recovering the whole key at a low probability, it can actually recover the majority of the key. This indicates the effectiveness of genetic algorithm (partially recovered key can also disclose very critical information about the plaintext). On the other hand, the recovery ratio for our P-Coding is only about 10% (when the key length is 19), which is very low, considering that even a randomly generated sequence can has some positions that coincide with the key.

5.3 Security Analysis for Enhanced P-Coding

If the PEF key does not leak in any generation, 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 appealing properties.

Security: After the compromise of security in current generation, the security level in following ones will be strong enough to resist 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 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 PEF 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 in P-Coding. Finally, the adversary should decode the messages by performing Eq. (13), which requires $\boldsymbol{O}(h^3)$ multiplication operations. Thus, the computational complexity in terms of multiplication is $\boldsymbol{O}(n \cdot m! \cdot h^3)$, which can be made sufficiently large by choosing m properly.

Recovery: As the PEF key is perturbed randomly and incrementally, it will become more and more irrelevant to its original value with the iterations of generations. Thus, even if the current key is disclosed, its randomness to the adversary will gradually recover after several generations. Theorem 5 gives the numerical result to justify this argument.

Theorem 5. After i generations, the expected number of all perturbed positions in the PEF key is approximately:

$$EX_i = \frac{i-1}{i+1}(n-m)[1-(1-\frac{m}{n-m})^{i+1}] + m \qquad (17)$$

when $n \to \infty$.

Proof: See Appendix E. □

Fig. 5 shows the approximated results from Theorem 5. For comparisons, we also include the exact results obtained from simulation. It can be seen that the number of perturbed positions in PEF key increases with the number of generations, meaning that its randomness will gradually recover after accidental disclosure.

6 Performance Evaluation

6.1 Analysis

The P-Coding Scheme: As the PEF key can be predistributed at the bootstrap stage, the only online computation overhead of P-Coding comes from the permutation encryption operations at the source and decryption operations at sinks. According to Eq. (10), 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 computation overhead is then $\boldsymbol{O}(n \cdot h)$ in terms of memory copy operations. Since the inherent overhead of network coding is at least $\boldsymbol{O}(h^3)$ in terms of multiplication operations (due to the necessity of Gaussian eliminations), P-Coding is quite lightweight in computation. In addition, P-Coding does not cause any space overhead either.

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 can be done within a 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 1, the extra computation overhead incurred by the enhanced scheme is just $\boldsymbol{O}(n)$.

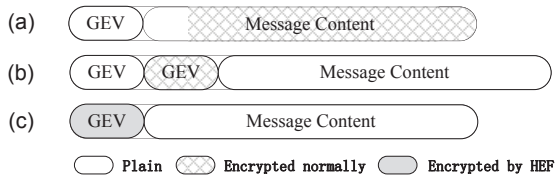


Fig. 6: Three cryptographic approaches for network-coded MANETs. Here GEV refers to Global Encoding Vector.

Comparisons: We compare the computation overhead of P-Coding and three other cryptographic schemes [10], [11] for network-coding based systems. These three schemes are depicted in Fig. 6: (a) the intuitive approach of directly encrypting the message content; (b) the approach proposed in [10] to encrypt only coding vectors; (c) the approach proposed in [11] to encrypt coding vectors using HEF.

For scheme (a), source messages will be encrypted using symmetric-key cryptographic algorithms (e.g., AES [19]), which cost around $\boldsymbol{O}(h \cdot l)$ multiplicative operations. For scheme (b), symmetric-key encryptions are only performed on GEVs, with the overhead of $\boldsymbol{O}(h^2)$ multiplicative operations. In addition, this scheme also requires operations of source encoding, intermediate recoding and sink decoding, which will cost $\boldsymbol{O}(h^2)$, $\boldsymbol{O}(M \cdot N \cdot h)$ and $\boldsymbol{O}(h^3)$ multiplicative operations, respectively (M denotes the average number packet recoding performed by all intermediate nodes; N is the average number of combined packets for each coding operation). It also requires a space overhead of ratio $h/(n+h)$, as it inserts a duplicate GEV of length h in each packet of length n. For scheme (c), it encrypts GEVs using the public-key based Paillier cryptosystem [23], which will incur a heavy computation overhead at both source and sinks. Moreover, the linear combinations performed by intermediate nodes on GEVs will also require multiplicative and exponential operations, which are even more expensive.

Since the computation overhead of P-Coding scheme is only $\boldsymbol{O}(n)$ in terms of memory copy operations, and there is no space overhead, it is fair to conclude that our P-Coding scheme outperforms the other three schemes on thwarting eavesdropping attacks.

6.2 Experiments

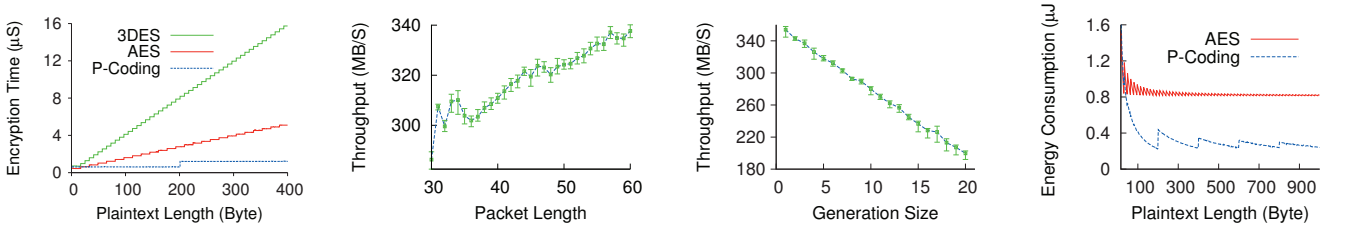
In the following, we will evaluate the performance of P-Coding through experiments. For implementation of P-Coding, we first split plaintext into multiple generations of packets, then let each generation go through a random linear coding process, and perform permutation encryption on each coded packet. For comparisons, we also implement AES and 3DES (both with CBC mode) using cryptography libraries of OpenSSL [24]. Our experiment environment is a Linux desktop with 3.30GHz Intel Core i3 CPU and 4GB memory. The performance metrics we consider include encryption time, throughput, and energy consumption.

Encryption Time: We let P-Coding, AES and 3DES encrypt a given plaintext with length up to 1K bytes, and measure the time they use. The experiment setting is as follows. First, since the key size of 3DES is 192 bits (64 bits for each round), we also choose a 192-bit key for AES. Second, since 192 bits can represent a permutation of length 46, we let the P-Coding key be a random permutation of length 45. The generation size is set to be 5, meaning that each generation has 5 packets. Note that the block size of P-Coding is $(45-5) \times 5 = 200$ bytes.

The results are shown in Fig. 7(a). It can be seen the encryption time of 3DES, AES, and P-Coding increases with steps of 8 bytes, 16 bytes, and 200 bytes, respectively. In addition, the encryption time of P-Coding is around $1/3$ that of AES. Considering OpenSSL's highly-optimized implementation of AES, and our limited time in optimizing P-Coding encryption algorithms, we expect this ratio to be even lower than that.

Throughput: Less encryption time means larger throughput. We will show that the throughput of P-Coding is affected by packet length (which equals the length of permutation key) and generation size (which equals the length of a GEVs). First, given a fixed generation size, the larger the packet length is, the lower the overhead of GEVs is. This is shown in Fig. 7(b), where the generation size is set to be 5. Second, given a fixed packet length, the throughput of P-Coding decreases linearly with respect to the generation size, as shown in Fig. 7(c). This is because when the packet length is fixed (here set to be 45), the increase of generation size means the increase of GEV length, and thus the increase of encryption overhead.

Energy Consumption: Less encryption time also means fewer CPU cycles, and less energy consumptions. To evaluate the energy efficiency of P-Coding in MANETs, we will estimate the energy consumption of P-Coding encryption on mobile nodes. Here, we choose the Motorola's "DragonBall MC68328", a common embedded



(a) Encryption Time (b) Throughput vs. Packet Legnth (c) Throughput vs. Generation Size (d) Per-Byte Energy Consumption

Fig. 7: Experimental results of P-Coding's performance (encryption time, throughput, and energy consumption).

microprocessor deployed in millions of wireless PDAs. Carman et al. [25] estimated that it took around $0.013mJ$ energy for AES to encrypt a 128-bit block with 128-bit key on DragonBall. Their estimation is based on the fact that it takes around 400 CPU cycles on 32-bit Intel microprocessors [26], and is obtained by scaling this result by some factor for the DragonBall microprocessors.

Here we use a similar method: we first measure the ratio of running time of P-Coding and 128-bit AES, and then estimate the per-byte energy consumption of P-Coding by scaling that of AES by this ratio. Fig. 7(d) gives the per-byte energy consumption of DragonBall microprocessor when using 128-bit AES and P-Coding, respectively. As the block size of P-Coding is 200 bytes, it can be seen that its per-byte energy consumption increases shapely when the plaintext length reaches multiples of 200 bytes, while drops gradually with further increase of plaintext length. As the plaintext length increases, the per-byte energy consumption of AES and P-Coding converges to $0.8\mu J$ and $0.25\mu J$, respectively.

7 Related Work

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 is proved to maximize the multicast throughput [5]. 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 [13].

The application of network coding in achieving minimum energy transmissions has received significant attention. In [6], Wu et al. show that by allow intermediate nodes encode packet, the problem of finding the minimum-energy multicast tree can be formulated as a linear program, which can be solved in polynomial time. This is in contrast with the fact that the same problem is NP-complete if traditional routing is used [27]. Fragouli et al. [7] studied the problem of energy-efficient broadcasting in MANETs using network coding, and propose some probabilistic algorithms. The same problem is treated in [8], in which the authors propose deterministic algorithms based on partial dominant pruning (PDP). Their algorithms relies on the information of two-hop neighbors and opportunistic listening to encode packets.

Besides reducing energy consumption of transmission in MANETs, network coding also bears a free security property, which has been researched in [12], [28], [29]. Bhattad et al. [12] 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 random linear network coding is inherently weakly secure with a high probability if coding is performed over a large finite field. Lima et al. [28] consider the threats posed by "nice but curious" intermediate nodes and develop an algebraic security criterion to access the intrinsic security provided by network coding. They derive the relationship between field size and the security level, and observe that the security is dependent on network topology. The algebraic security criterion is essentially weak security. Based on the weak security model, Wang et al. [29] design a polynomial-time deterministic code to secure linear network coding. They show that by using this scheme, optimal throughput for multiple streams between a single source-destination pair can be achieved.

By leveraging the intrinsic security of network coding, some cryptographic approaches have been proposed to secure network-coding-based applications. One scheme is SPOC [10], proposed by Vilela et al, in which the source encrypts/locks the GEV of each message after random linear coding, and attach another set of GEVs to enable standard network coding. Recivers can recover the source messages by following a decode-decrypt-decode procedure. This scheme is essentially an end-to-end cryptographic approach, and is lightweight in computation. Another scheme proposed by Fan et al. [11] is based on Homomorphic Encryption Function (HEF) [30]. This scheme has the coding coefficients encrypted using HEF. Due to the homomorphic property of HEF, linearly combination operations can be directly performed on the encrypted coding coefficients. As a result, no extra coding coefficients are needed as by SPOC. As another difference from SPOC, Fan's HEF-based scheme can achieve both content secrecy (i.e., confidentiality), and contextual secrecy (i.e., privacy) at the same time. However, both of these two schemes fail to fully exploit the mixing nature of network coding.

8 Discussion and Conclusion

8.1 Discussion

Node Mobility: In the above, we have not specified how to handle node mobility. Actually, node mobility poses a challenge for key management: since network topology is constantly changing, there is no pre-established route for key establishment [31]. However, once keys are established, mobility has little impact on encryptions/decryptions. In this paper, we view key management as an orthogonal problem, which has been studied in many previous works [32].

Extension: We believe the applications of P-Coding are beyond MANETs. Any system that enables random linear network coding, like P2P live streaming [33], distributed storage [34], and file distribution [18] may use P-Coding for confidentiality. While the values of applying P-Coding in these applications are not as high as in MANETs, since these applications are generally not energy-constrained and any symmetric cryptographic algorithms would function well. We will extend our scheme to other scenarios where encryption efficiency is critical.

8.2 Conclusion

This paper studied the problem of energy saving in MANETs based on the technique of network coding. Previous studies demonstrated that network coding can reduce energy consumption with less transmissions in MANETs. We proposed P-Coding, a lightweight encryption scheme on top of network coding, to further reduce energy consumption in MANETs by cutting the security cost. P-Coding exploits the intrinsic security property of network coding, and uses simple permutation encryptions to generate considerable confusion to eavesdropping adversaries. We showed that P-Coding is efficient in computation, and incurs less energy consumption for encryptions/decryptions. Our future work includes extending the application of P-Coding to other communication networks, e.g., vehicular ad hoc networks.

References

[1] P. Zhang, Y. Jiang, C. Lin, Y. Fan, and X. Shen, "P-Coding: Secure network coding against eavesdropping attacks," in *Proceedings of IEEE INFOCOM*, Mar. 2010.

[2] S. Singh, C. Raghavendra, and J. Stepanek, "Power-aware broadcasting in mobile ad hoc networks," in *Proceedings of IEEE PIMRC*, 1999.

[3] J. Wieselthier, G. Nguyen, and A. Ephremides, "Algorithms for energy-efficient multicasting in static ad hoc wireless networks," *Mobile Networks and Applications*, vol. 6, no. 3, pp. 251–263, 2001.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

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

[6] Y. Wu, P. Chou, and S. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1906–1918, 2005.

[7] C. Fragouli, J. Widmer, and J. Boudec, "A network coding approach to energy efficient broadcasting: from theory to practice," in *Proceedings of IEEE INFOCOM*, 2006.

[8] L. Li, R. Ramjee, M. Buddhikot, and S. Miller, "Network coding-based broadcast in mobile ad-hoc networks," in *Proceedings of IEEE INFOCOM*, 2007.

[9] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols," *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 128–143, 2006.

[10] J. P. Vilela, L. Lima, and J. Barros, "Lightweight security for network coding," in *Proceedings of IEEE ICC*, May 2008.

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

[12] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," in *Proceedings of NetCod*, Apr. 2005.

[13] 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 Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

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

[15] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of applied cryptography*. CRC Press, Oct. 1996.

[16] N. Cai and R. W. Yeung, "Secure network coding," in *Proceedings of IEEE ISIT*, Jun 2002.

[17] A. Dimovski and D. Gligoroski, "Attacks on the transposition ciphers using optimization heuristics," in *Proceedings of International Scientific Conference on Information, Communication and Energy Systems and Technologies*, Oct. 2003.

[18] C. Gkantsidis and P. Rodriguez, "Network coding for large scale file distribution," in *Proceedings of IEEE INFOCOM*, Mar. 2005.

[19] J. Daemen and V. Rijmen, *The design of Rijndael: AES–the advanced encryption standard*. Springer Verlag, 2002.

[20] L. C. Washington and W. Trappe, *Introduction to cryptography: with coding theory*. Prentice Hall PTR, 2002.

[21] J. Giddy and R. Safavi-Naini, "Automated cryptanalysis of transposition ciphers," *The Computer Journal*, vol. 37, no. 5, pp. 429–436, 1994.

[22] M. D. Russell, J. A. Clark, and S. Stepney, "Making the most of two heuristics: Breaking transposition ciphers with ants," in *The 2003 Congress on Evolutionary Computation*, 2003.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuocity classes," in *Proceedings of EUROCRYPT*, May 1999.

[24] "The OpenSSL project," http://www.openssl.org/.

[25] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and approaches for distributed sensor network security (final)," *DARPA Project report*, 2000.

[26] K. Aoki and H. Lipmaa, "Fast implementations of aes candidates," in *Third AES Candidate Conference*, 2000, pp. 13–14.

[27] M. Čagalj, J. Hubaux, and C. Enz, "Minimum-energy broadcast in all-wireless networks: Np-completeness and distribution issues," in *Proceedings of ACM Mobicom*, 2002.

[28] L. Lima, M. Médard, and J. Barros, "Random linear network coding: A free cypher?" in *Proceedings of IEEE ISIT*, Jun. 2007.

[29] J. Wang, J. Wang, K. Lu, B. Xiao, and N. Gu, "Optimal linear network coding design for secure unicast with multiple streams," in *Proceedings of IEEE INFOCOM*, Mar. 2010.

[30] J. Benaloh, "Dense probabilistic encryption," in *Proceedings of the Workshop on Selected Areas in Cryptography*, Aug. 1994.

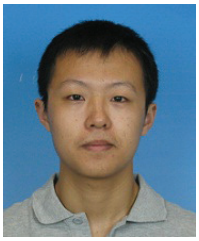
[31] A.-F. Chan, "Distributed symmetric key management for mobile ad hoc networks," in *Proceedings of IEEE INFOCOM*, 2004.

[32] J. V. D. Merwe, D. Dawoud, and S. McDonald, "A survey on peer-to-peer key management for mobile ad hoc networks," *ACM computing surveys*, vol. 39, no. 1, p. 1, 2007.

[33] M. Wang and B. Li, "R^2: Random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1655–1666, Dec. 2007.

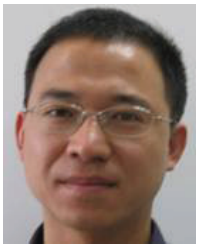
[34] Y. Hu, H. C. Chen, P. P. Lee, and Y. Tang, "Nccloud: applying network coding for the storage repair in a cloud-of-clouds," in *Proceedings of USENIX FAST*, 2012.

[35] S. Ross, *Introduction to probability models*, 9th ed. Academic Press, 2007.

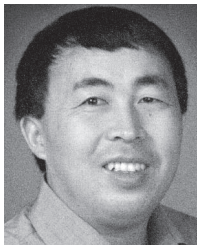


Peng Zhang received the B.Eng. degree in Computer Science from Beijing University of Posts and Telecommunications in 2008. He is now a Ph.D. candidate in the Department of Computer Science and Technology at Tsinghua University. His research interests include network coding, network security, and information-centric networking.

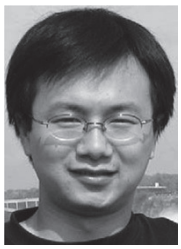
Chuang Lin received the Ph.D. degree in Computer Science from Tsinghua University in 1994. He is now a professor of the Department of Computer Science and Technology, Tsinghua University, China. He is a Honorary Visiting Professor, University of Bradford, UK. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 300 papers in research journals and IEEE conferences, and four monographs in these areas.



Yixin Jiang received the Ph.D. degree in Computer Science from Tsinghua University in 2006. He is now an associate professor at Tsinghua University. In 2005, he was a Visiting Scholar with the Department of Computer Science, Hong Kong Baptist University. From 2007 to 2009, he was a Post Doctorial Fellow with University of Waterloo. His research interests include wireless network security, trusted computing and network coding.



Xuemin (Sherman) Shen (IEEE M'97-SM'02-F09) received the B.Sc.(1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. He is a Professor and University Research Chair, Department of Electrical and Computer Engineering, University of Waterloo, Canada. He was the Associate Chair for Graduate Studies from 2004 to 2008. Dr. Shen's research focuses on resource management in interconnected wireless/wired networks, wireless network security, wireless body area networks, vehicular ad hoc and sensor networks. He is a co-author/editor of six books, and has published more than 600 papers and book chapters in wireless communications and networks, control and filtering. Dr. Shen served as the Technical Program Committee Chair for IEEE VTC'10 Fall, the Symposia Chair for IEEE ICC'10, the Tutorial Chair for IEEE VTC'11 Spring and IEEE ICC'08, the Technical Program Committee Chair for IEEE Globecom'07, the General Co-Chair for Chinacom'07 and QShine'06, the Chair for IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking. He also serves/served as the Editor-in-Chief for IEEE Network, Peer-to-Peer Networking and Application, and IET Communications; a Founding Area Editor for IEEE Transactions on Wireless Communications; an Associate Editor for IEEE Transactions on Vehicular Technology, Computer Networks, and ACM/Wireless Networks, etc.; and the Guest Editor for IEEE JSAC, IEEE Wireless Communications, IEEE Communications Magazine, and ACM Mobile Networks and Applications, etc. Dr. Shen received the Excellent Graduate Supervision Award in 2006, and the Outstanding Performance Award in 2004, 2007 and 2010 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada, and the Distinguished Performance Award in 2002 and 2007 from the Faculty of Engineering, University of Waterloo. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an IEEE Fellow, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, and a Distinguished Lecturer of IEEE Vehicular Technology Society and Communications Society.



Yanfei Fan received the M.Eng. degree (2005) from Tsinghua University, China, and the B.Eng. degree (2002) from Beijing University of Posts and Telecommunications, China, all in Computer Science. He is currently pursuing his Ph.D. degree in the Department of Electrical and Computer Engineering at University of Waterloo, Canada. His research interests include network coding, security in wireless communication and mobile computing.