This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

An Efficient Privacy-Preserving Scheme against Traffic Analysis Attacks in Network Coding

Yanfei Fan, Yixin Jiang, Haojin Zhu, Xuemin (Sherman) Shen

Department of Electrical and Computer Engineering, University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{yfan, yixin, h9zhu, xshen}@bbcr.uwaterloo.ca

Abstract — **Privacy threat is one of the critical issues in network coding, where attacks such as traffic analysis can be easily launched by a malicious adversary once enough encoded packets are collected. Furthermore, the encoding/mixing nature of network coding precludes the feasibility of employing the existing privacy-preserving techniques, such as Onion Routing, in network coding enabled networks. In this paper, we propose a novel privacy-preserving scheme against traffic analysis in network coding. With homomorphic encryption operation on Global Encoding Vectors (GEVs), the proposed scheme offers two significant privacy-preserving features, packet flow untraceability and message content confidentiality, for efficiently thwarting the traffic analysis attacks. Moreover, the proposed scheme keeps the random coding feature, and each sink can recover the source packets by inverting the GEVs with a very high probability. Theoretical analysis and simulative evaluation demonstrate the validity and efficiency of the proposed scheme.**

Keywords — Network coding; homomorphic encryption; privacy preservation; traffic analysis

I. INTRODUCTION

Network coding has been widely recognized as a promising information dissemination approach to improve the network performance. Compared with conventional packet forwarding technologies, network coding offers, by allowing and encouraging coding/mixing operations on intermediate forwarders [1], several significant advantages such as the potential throughput improvement [3], transmission energy minimization [4], and delay minimization [5]. In addition, two key techniques, random coding [2] and linear coding [18], further promote the development of network coding technologies. The random coding makes network coding more practical, while the linear coding is proved to be sufficient and computationally efficient for network coding.

Primary applications of network coding include the file distribution and multimedia streaming on P2P networks [6], data transmission in sensor networks [7], tactical communications in military networks [8] and so on, where the application environment may be hostile or security/privacy-sensitive. However, network coding is susceptible to possible malicious attacks, which pose great threats to the security and privacy of a network system. Previous studies on securing network coding mainly focus on detecting and filtering out polluted messages [10]. Privacy issues, especially on how to protect encoded messages from tracking or traffic analysis, have received little attention.

Among all security threats, traffic analysis attacks, such as flow tracing, are of special interest in multi-hop wireless networks, where network coding is expected to play an important role in the near future. Consider a simple example of multicast communication in military ad hoc networks, where the nodes can communicate with each others. If an attacker can intercept the messages and even correlate the source and the destination by flow tracing, it may leak some sensitive information such as the location of some critical nodes (e.g. the commanders) and further impair the location privacy. Following this, the attacker can take some action to launch the so-called *Decapitation Strike* to destroy these critical nodes [23]. Another example is the event reporting in wireless sensor networks, where flow tracing can help attackers to identify the location of concerned events, e.g., the appearance of an endangered animal in a monitoring area, and then take subsequent actions to capture or kill the animals [22]. Such privacy threats also exist in network coding enabled networks since explicit Global Encoding Vectors (GEVs) prefixed in the encoded packets may provide a back door for adversaries to launch these attacks to compromise the privacy of users.

Preventing traffic analysis and provisioning anonymous networking is critical for securing network coding, especially in military related, such as sensor or tactical, networks. Anonymous networking refers to communicating on a network without revealing the source-destination pairs or the paths of traffic flow, which is equivalent to source (or destination) anonymity. Anonymous networking can be achieved by preventing flow tracing, which is one of the most severe traffic analysis attacks. The existing privacy-enhancing techniques designed for traditional packet-forwarding networks, such as proxy-based schemes [11], [12], Chaum's mix-based schemes [13], [14], and onion-based schemes [15], [16], are not suitable for network coding, since such traditional approaches may either require a series of trusted forwarding proxies to "confuse" adversaries about the data streams, or target at single-path transmission, which is not in line with the multi-path transmission principle in network coding.

Similar to Chaum's mix-based schemes [13], [14], network coding provides an intrinsic message mixing mechanism, which implies that privacy preservation may be efficiently performed in a distributed manner. Moreover, unlinkability between incoming packets and outgoing packets can be achieved by mixing the incoming packets at each intermediate node.

However, due to explicit GEVs (or called tags), the privacy offered by such a mixing feature is still vulnerable, since the earliest decoding is possible and linear dependence can be easily analyzed. A naïve solution to address the vulnerability is

employing link-to-link encryption. This method will certainly bring heavy computational overhead to nodes, and thus result in significant performance degradation of the whole system. Additionally, it can not preserve the privacy for network traffic, if some intermediate nodes are compromised by malicious adversaries. Such deficiencies motivate us to explore an efficient privacy-preserving scheme for random linear network coding.

In this paper, based on Homomorphic Encryption Functions (HEFs) [19], [20], we propose an efficient privacy-preserving scheme against traffic analysis in network coding. Our objective is to achieve anonymous networking in terms of source anonymity by preventing flow tracing. To the best of our knowledge, this is the first study on privacy issues in network coding. The proposed scheme offers the following significant features: 1) **Enhanced Privacy against packet tracing and traffic analysis**: With the employment of HEFs, the confidentiality of GEVs is effectively guaranteed, which makes it difficult for attackers to recover the encoded GEVs. Even when some intermediate nodes are compromised, the adversaries still can not decrypt the GEVs, since only the sinks know the decryption key. The confidentiality of GEVs brings the implicative benefit of the confidentiality of message content, because message decoding only relies on the GEVs. Moreover, with message recoding on encrypted GEVs, the coding/mixing feature of network coding can be exploited in a natural manner to satisfy the mixing requirements of privacy preservation against traffic analysis. 2) **Efficiency**: Due to the Homomorphism of HEFs, message recoding at intermediate nodes can be directly performed on encrypted messages, without knowing the decryption keys or performing expensive decryption operations on each incoming packet. The performance evaluation on computational complexity demonstrates the practicality of the proposed scheme. 3) **High Invertible GEVs**: Random network coding is feasible only if the prefixed GEVs are invertible with a high probability. Theoretical analysis has proved that the influence which the HEFs pose upon the invertible probability of GEVs is negligible. Thus random network coding can be preserved in the proposed scheme, since the encrypted GEVs is invertible with a very high probability.

The remainder of the paper is organized as follows. In Section II, preliminaries related to the proposed scheme are given, including network coding model, threat model, and homomorphic encryption concept. In Section III, the proposed privacy-preserving scheme is introduced in detail. In Section IV and V, the security analysis and performance metrics are presented, respectively. In Section VI, related works are surveyed, followed by the conclusions in Section VII.

II. PRELIMINARIES

A. Network Coding Model

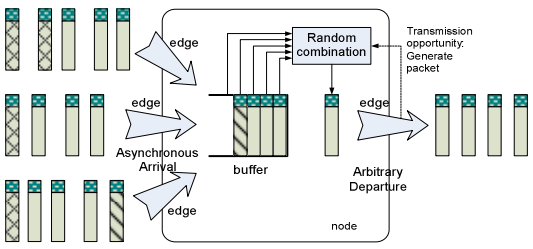

Fig. 1. Random coding (mixing) at intermediate nodes

Unlike traditional packet-forwarding systems, network coding allows intermediate nodes to perform computation on input messages, which makes the output messages be the mixture of input ones. This elegant principle implies a plethora of surprising results. One of the most exciting opportunities is random coding [2], as shown in Fig. 1.

An overview of network coding and possible applications has been given in [5]. Another two techniques, packet tagging and buffering, are also key for network coding to be practical. Packet tagging will be introduced in the next subsection. In practical network coding, source information should be divided into blocks with h packets in each block. All coded packets related to the k^{th} block belong to generation k and random coding is only performed among the packets in the same block. Packets within a generation need to be synchronized by buffering for the purpose of network coding in intermediate nodes.

Consider an acyclic network (V, E, c) with unit capacity, i.e., $c(e)=1$ for all $e \in E$, meaning that each edge can carry one symbol per unit time. Assume that each symbol is an element of a finite field $\mathbb{F}_q$. Consider a network scenario with multicast sessions, where a session is comprised of one source $s \in V$ and a set of sinks $T \subseteq V$ (or one single sink $t \in V$). Let $h = MinCut(s,T)$ be the multicast capacity, and $x_1, \cdots, x_h$ be the h symbols to be delivered from s to T.

For each outgoing edge e of a node v, let $y(e) \in \mathbb{F}_q$ denote the symbol carried on e, which can be computed as a linear combination of the symbols $y(e')$ on the incoming edges e' of node v, i.e., $y(e) = \sum_{e'} \beta_{e'}(e) y(e')$. The coefficient vector $\boldsymbol{\beta}(e) = [\beta_{e'}(e)]$ is called as *Local Encoding Vector* (LEV).

By induction, the symbol $y(e)$ on any edge $e \in E$ can be computed as a linear combination of the source symbols $x_1, \cdots, x_h$, i.e., $y(e) = \sum_{i=1}^{h} g_i(e) x_i$. The coefficients form a *Global Encoding Vector* (GEV) $\boldsymbol{g}(e) = [g_1(e), ..., g_h(e)]$, which can be computed recursively as $\boldsymbol{g}(e) = \sum_{e'} \beta_{e'}(e) \boldsymbol{g}(e')$, using the LEVs $\boldsymbol{\beta}(e)$. The GEV $\boldsymbol{g}(e)$ represents the code symbol $y(e)$ in terms of the source symbols $x_1, \cdots, x_h$.

Suppose that a sink $t \in T$ receives symbols $y(e_1), \cdots, y(e_h)$, which can be expressed in terms of the source symbols as

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} g_1(e_1) & \cdots & g_h(e_1) \\ \vdots & \ddots & \vdots \\ g_1(e_h) & \cdots & g_h(e_h) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G_t \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix}, \quad (1)$$

where G_t is called *Global Encoding Matrix* (GEM) and the i^{th} row of the matrix G_t is the GEV associated with $y(e_i)$. Sink t can further recover the h source symbols by inverting the matrix G_t and applying the inverse on $y(e_1), \cdots, y(e_h)$.

In a real network, each packet can be considered as a vector of symbols $\boldsymbol{y}(e) = [y_1(e), \cdots, y_N(e)]$. By likewise grouping the source symbols into packets $\boldsymbol{x}_i = [x_{i,1}, \cdots, x_{i,N}]$, the above algebraic relationships carry over to packets. To facilitate the decoding at the sinks, each message is tagged with its GEV $\boldsymbol{g}(e)$, which can be easily achieved by prefixing the i^{th} packet $\boldsymbol{x}_i$ with the i^{th} unit vector $\boldsymbol{u}_i$, and each packet is automatically

tagged with the corresponding GEV, since

$$[\boldsymbol{g}(e), \boldsymbol{y}(e)] = \sum_{e'} \beta_{e'}(e)[\boldsymbol{g}(e'), \boldsymbol{y}(e')] = \sum_{i=1}^{h} g_i(e)[\boldsymbol{u}_i, \boldsymbol{x}_i] \quad (2)$$

The benefit of tags is that the GEVs can be found within the packets themselves, so that the sinks can compute G_t without knowing the network topology or packet-forwarding paths. Nor is a side channel required for the communication of G_t. Actually, the network can be dynamic, with nodes and edges being added or removed in an ad hoc way. The coding arguments can be time varying and random.

B. Homomorphic Encryption Function

Homomorphic Encryption Functions (HEFs) have the property of homomorphism, which means operations on plaintext can be performed by operating on corresponding ciphertext. For example, suppose $E(\cdot)$ is a HEF. It is easy to compute $E(x+y)$ from $E(x)$ and $E(y)$ without knowing the corresponding plaintext x and y. To be applicable in the proposed scheme, a HEF $E(\cdot)$ needs to satisfy the following properties:

1) **Additivity**: Given the ciphertext, $E(x)$ and $E(y)$, there exists a computationally efficient algorithm $Add(\cdot,\cdot)$ such that $E(x+y) = Add(E(x), E(y))$.

2) **Scalar Multiplicativity**: Given $E(x)$ and a scalar t, there exists a computationally efficient algorithm $Mul(\cdot,\cdot)$ such that $E(t \cdot x) = Mul(E(x), t)$.

Actually, the scalar multiplicativity can be deduced from the additivity, since $E(t \cdot x) = E(\sum_{i=1}^{t} x)$. Benaloh [19] and Paillier [20] cryptosystems are such two additive HEFs, where the addition on plaintext can be achieved by performing a multiplicative operation on the corresponding ciphertext, i.e., $E(x_1 + x_2) = E(x_1) \cdot E(x_2)$. Further, the following two equations can be easily derived.

$$E(t \cdot x) = E^t(x)$$
$$E\left(\sum_i t_i \cdot x_i\right) = \prod_i E^{t_i}(x_i) \quad (3)$$

C. Threats Model and Security Assumptions

We consider the following two attack models:

Outside Attacker: An outside attacker can be considered as a global passive eavesdropper who has the ability to observe all network links, as shown in Fig. 2. An outside attacker can examine the tags and message content, and thus link outgoing packets with incoming packets. Further, even if messages are encrypted in an end-to-end manner, it is still possible for a global outside attacker to trace packets by analyzing and comparing the message ciphertext.

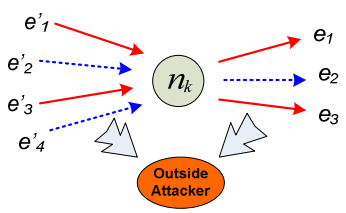

Fig. 2. Attack Model: Outside Attacker

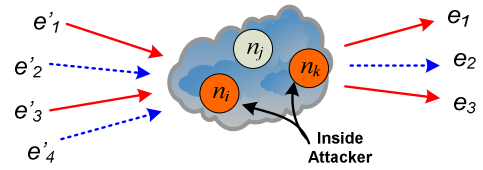

Fig. 3. Attack Model: Inside Attacker

Inside attacker: An inside attacker may compromise several intermediate nodes. Link-to-link encryption is vulnerable to inside attackers since they may already obtain the decryption keys and reveal message plaintext.

Both inside and outside attackers may perform more advanced traffic analysis techniques, including size correlation, time-order correlation, and message content correlation [21]. Thus, adversaries can further explore these correlations to deduce the forwarding paths [22].

Without loss of generality, we assume that a secure, anonymous routing protocol is deployed to assist network nodes to determine the forwarding paths. Many solutions have been proposed to deal with this issue [23]. We also assume that the generation number of a packet is hidden by the secure routing scheme. Under this assumption, an outside attacker can not determine the generation of a packet for its further analysis.

III. The Proposed Privacy-Preserving Scheme for Network Coding

In this section, we propose a novel privacy-preserving scheme for network coding, which can efficiently thwart traffic analysis attacks such as flow tracing, followed by theoretical analysis on the invertibility of GEMs.

A. The Proposed Privacy-Preserving Scheme

Though providing an inherent mixing mechanism, which is similar to Chaum's mix-based schemes [13], [14], the original network coding cannot provide privacy guarantee due to explicit GEVs, since an adversary can recover the original messages as long as enough packets are collected. Link-to-link encryption is also vulnerable to inside attackers since they may already compromise several intermediate nodes and obtain the corresponding decryption keys.

An intuitive way to resolve this issue is to keep GEVs confidential to intermediate nodes by encrypting the GEVs in an end-to-end manner, which can prevent compromised intermediate nodes from analyzing GEVs or recovering the original messages. Such an intuitive method, however, cannot prevent the adversaries from tracking the message ciphertext if the "mixing" feature is disabled by the end-to-end encryption.

To address this issue, in this paper, we employ the Paillier cryptosystem [20] as the HEF to apply encryption to GEVs. HEF can not only keep the confidentiality of GEVs, but also leave intermediate nodes the capability for efficiently mixing the coded messages. In the Paillier cryptosystem, given a message m and the public key (n, g), the encryption function can be described as follows,

$$E(m) = g^m \cdot r^n \pmod{n^2}, \quad (4)$$

where r is a random factor in the Paillier cryptosystem. It satisfies the following homomorphic property:

$$E(m_1) \cdot E(m_2) = g^{m_1+m_2} \cdot (r_1 r_2)^n \pmod{n^2} = E(m_1 + m_2). \quad (5)$$

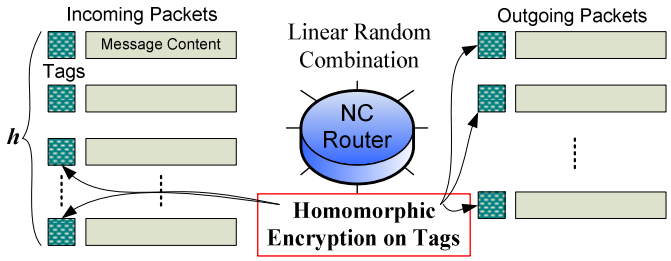

Fig. 4. Homomorphic encryption on packet tags

With HEFs, intermediate nodes are allowed to directly perform random coding/mixing on the coded messages and encrypted tags, as shown in Fig. 4. In other words, due to the homomorphism of the HEF, linear network coding can be achieved by operating on the encoded messages and the ciphertext of GEVs, without knowing the decryption keys or performing the decryption operations.

There are three parties in the proposed scheme: the source, intermediate nodes (also called forwarders), and sinks. The proposed scheme primarily consists of three phases: source encoding, intermediate random linear recoding, and sink decoding. Without loss of generality, we assume that each sink acquires two keys, the encryption key *ek* and the decryption key *dk*, from an offline Trust Authority (TA) and the encryption key *ek* is published to all the other nodes. For supporting multicast, a group of sinks is required to negotiate the key pair in advance [26]. During message transmission, we also assume that the encryption key *ek* and the generation number of a packet are hidden by the secure routing scheme [23], and only authenticated intermediate nodes can obtain the information.

Source Encoding: Consider the situation that a source has h messages, say $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_h$, to be sent out. The source first prefixes h unit vectors to the h messages, as illustrated in Fig. 5. After tagging, the source can choose a random LEV and then perform a normal linear encoding operation on these messages. In this way, finally, one LEV will generate an encoded message with the GEV (which is equal to the LEV temporarily) tagged.

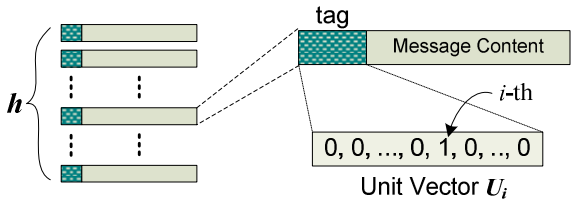

Fig. 5. Packet Tagging before Source Coding

To offer confidentiality for the tags, homomorphic encryption operations are employed on these tags as follows:

$$\begin{aligned} c_i(e) &= E_{ek}(g_i(e)), (1 \le i \le h) \\ \boldsymbol{c}(e) &= [c_1(e), c_2(e), \cdots, c_h(e)] \end{aligned}, \tag{6}$$

where the notation *ek* denotes the encryption key.

Intermediate Random Linear Recoding: After receiving a number of packets belonging to the same generation, an intermediate node can perform random linear coding on these packets. To generate an outgoing packet, firstly, a random LEV $[\beta_1,\cdots,\beta_h]$ is chosen independently of the other random LEVs. Then, a linear combination of message content of the incoming packets is computed as the message content of the outgoing packet, as shown in Fig. 1.

Since the tags of the h incoming packets are in ciphertext format, and the intermediate node has no knowledge of the corresponding decryption keys, it is difficult for the intermediate node to perform functions such as earliest decoding to get the message content. However, due to the homomorphism of the encryption function, a linear transformation can be directly performed on the encrypted tags of the incoming packets to generate a new tag for the outgoing packet, namely,

$$g(e) = \sum_{i=1}^{h} \beta_i(e) g(e'_i). \tag{7}$$

The GEV of a new outgoing packet can be calculated according to equation (7). By utilizing the homomorphic characteristic of the encryption on GEVs of incoming packets, the ciphertext of the new GEVs for outgoing packets can be calculated through the following equation:

$$\begin{aligned} E_{ek}(g(e)) &= E_{ek}(\sum_{i=1}^{h} \beta_i(e) g(e'_i)) \\ &= \prod_{i=1}^{h} E_{ek}(\beta_i(e) g(e'_i)). \\ &= \prod_{i=1}^{h} E_{ek}^{\beta_i(e)}(g(e'_i)) \end{aligned} \tag{8}$$

The ciphertext of new GEVs can be computed based on the ciphertext of GEVs of incoming packets without knowing the decryption key. Finally, the ciphertext of the new GEV is prefixed to the corresponding message content to form a new outgoing packet, which is sent out to the downstream nodes.

Sink Decoding: After receiving a packet, the sink first decrypt the packet tag using the corresponding decryption key *dk*.

$$\begin{aligned} g_i(e) &= D_{dk}(c_i(e)) \quad (1 \le i \le h) \\ \boldsymbol{g}(e) &= [g_1(e), g_2(e), \cdots, g_h(e)] \end{aligned} \tag{9}$$

Once enough packets are received, a sink can decode the packets to get the original message. Then, the sink derives the decoding vector, which is the inverse of the GEM, as shown in the following equations.

$$\begin{aligned} \boldsymbol{G}^{-1} \cdot \boldsymbol{G} &= \boldsymbol{U} \pmod n \\ \boldsymbol{G} &= [\boldsymbol{g}(e_1), \boldsymbol{g}(e_2), \cdots, \boldsymbol{g}(e_h)]^T \end{aligned} \tag{10}$$

Finally, the sink can use the inverse matrix to recover the original messages, shown as follows.

$$\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_h \end{bmatrix} = \boldsymbol{G}^{-1} \begin{bmatrix} \boldsymbol{y}(e_1) \\ \vdots \\ \boldsymbol{y}(e_h) \end{bmatrix}. \tag{11}$$

A key issue in sink decoding is the invertibility of the GEM. We will further discuss the invertibility of a GEM, which is comprised of h GEVs with h elements in each GEV.

B. Invertibility of a GEM

Suppose we have a GEM A, which is comprised of h GEVs with h elements each GEV. $|A|$ and A^* are the determinant and the adjoint of the matrix A, respectively. According to the theory of linear algebra, finding the inverse of a square matrix A is equivalent to solving the corresponding system of linear equation with A being the coefficient matrix.

Similar to solving a system of linear equations, Gaussian elimination can also be used to solve a system of linear congruence equations. Due to the homomorphism of the modulo congruence in terms of the addition, subtraction, and multiplication operations, a system of linear congruence equations can

be separated into several single equations with one unknown in each equation, shown as follows:

$$|A|x_i = \sum_{j=1}^{h}(-1)^{i+j} y_j M_{ij} \pmod n . \quad (12)$$

A system of linear congruence equations is solvable if and only if every independent equation is solvable. The only difference of solving a system of linear equations and solving a system of linear congruence equations lies in finding the inverse of $|A|$ modulo n. In order to further discuss the solutions, we formulate a system of linear congruence equations as:

$$|A|x_i = \left|\tilde{A}_i\right| \pmod n \ \ (i = 1, \cdots, h), \quad (13)$$

where $\left|\tilde{A}_i\right| = \sum_{j=1}^{h}(-1)^{i+j} y_j M_{ij} \pmod n$.

Theorem 1: A system of linear congruence equations has a unique solution only if $|A| \neq 0$.

Proof: See Appendix A. ■

However, this condition is not a sufficient condition for a system of linear congruence equations to have a unique solution, because a solution for $|A|x_i = \left|\tilde{A}_i\right|$ does not imply a corresponding solution for $|A|x_i = \left|\tilde{A}_i\right| \pmod n$.

Theorem 2: A system of linear congruence equations has d^h solutions if:

$$\begin{cases} |A| \neq 0 \\ \left|\tilde{A}_i\right| \equiv 0 \pmod d \quad (i = 1, 2, \cdots, h) \end{cases} \quad (14)$$

where $d = \gcd(|A|, n)$.

Proof: See Appendix B. ■

Corollary 1: A system of linear congruence equations has a unique solution if and only if:

$$\begin{cases} |A| \neq 0 \\ \gcd(|A|, n) = 1 \end{cases} \quad (15)$$

and $x_i = |A|^{-1}\left|\tilde{A}_i\right| \pmod n \quad (i = 1, 2, \cdots, h)$.

Proof: See Appendix C. ■

Theorem 2 and **Corollary 1** show that a solvable system of linear congruence equations does not imply the invertibility of the corresponding coefficient matrix. A stronger condition, i.e., $\gcd(|A|, n) = 1$, is required for the invertibility of a coefficient matrix A modulo n. The above theorems and corollary generally hold no matter if n takes the value of a prime number q or the product of two prime numbers p and q. In section V, we will further give a quantitative analysis on the invertible probability of a coefficient matrix.

IV. Security Analysis

In this section, we analyze the privacy preservation from the proposed scheme, as well as an extra privacy-enhancing policy.

A. Privacy Enhancement

The allowance of earliest decoding conflicts with the basic principle of privacy preservation. Therefore, the proposed scheme inherently precludes the feasibility of earliest decoding. In the following, we further discuss what protection the proposed scheme can provide and how well it can provide.

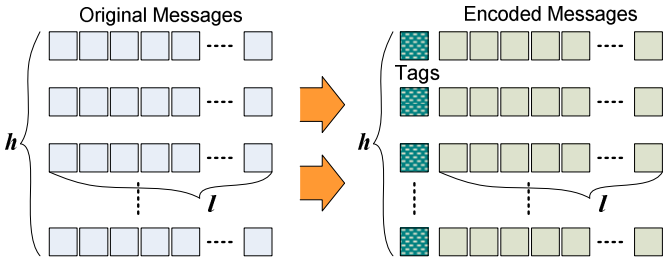

Fig. 6. Message Content Encoding

The first protection is on generation numbers. If an adversary attempts to launch a traffic-analysis attack, a better method is to identify the packet generations first. However, in our scheme, the generation number is hidden in the secure routing scheme.

The second protection is to resist size correlation and time-order correlation attacks, which are two widely-used techniques in traffic analysis. In the proposed scheme, every message is trimmed to be of the same size, which makes size correlation useless to adversaries. The inherent buffering technique [5] of network coding can greatly improve the resistance to time-order correlation.

The third protection is to eliminate message content correlation, which is also commonly used in traffic analysis. Assume that the first protection on generation numbers is compromised, due to the vulnerability of a secure routing system or the compromise of an intermediate node. To launch content-correlation based traffic analysis, adversaries must compromise several forwarders to intercept packets of the same generation. Let the generation number going through these compromised nodes be w during a period of time v. To determine if an intercepted packet, say m_d, in some downstream link, is a linear combination of some known packets, the following steps should be performed:

1) The first step is to choose h columns of linearly independent coded symbols from messages with the same generation, as shown in Fig. 6. The independence can be easily tested by employing Gaussian elimination on the matrix of the h vectors. The computational complexity of this step is $\boldsymbol{O}(h^3)$ in terms of multiplication operations.

2) The second is to compute the coefficient vector of the linear combination associated to the message m_d from the chosen h vectors. With Gaussian elimination, the extra computational complexity is $\boldsymbol{O}(h^2)$ in terms of multiplication operations.

3) The third is to verify if the resultant coefficient vector satisfies all the other $l-h$ columns of code symbols. The computational complexity is $\boldsymbol{O}(h(l-h))$ in terms of multiplication operations.

From the above three steps, the whole computational complexity to check if an objective message is the linear combination of a generation of messages is $\boldsymbol{O}(h^3 + h(l-h)) \sim \boldsymbol{O}(h^3 + h \cdot l))$ in terms of multiplication. To check w generations, the computational complexity is $\boldsymbol{O}(w(h^3+h \cdot l))$ in terms of multiplication.

The probability of false positive verification (to incorrectly determine a generation of messages to be the coding source of a specified message) is considerably low. Because the correlation

between messages of the same generation is uncertain or unknown to network encoder, it is reasonable to assume that the other (*l*-*h*) columns of code symbols are random distributed in the field *F*. Thus the probability of false positive verification is $P_{fpv}(h,l) = |F|^{h-l}$, which is pretty small value since $l \gg h$.

In summary, under the strong assumption that an adversary can compromise a group of nodes which are associated with a bunch of Min-Cut edges, computational overhead of $\boldsymbol{O}(w(h^3+h\cdot l))$ in terms of multiplication is also required for the adversary to determine if a downstream packet is coming from one of the *w* generations of intercepted packets. If the first protection on generation numbers is not compromised, i.e., adversaries fail to identify the generation of a packet, the computational overhead will increase to $O(C_{wh}^{h}(h^3 + h\cdot l))$ in terms of multiplication. Compared to the existing network coding with explicit GEVs, the proposed privacy-preserving scheme greatly enhances the privacy, such that the traffic analysis attacks are almost computationally impossible.

B. Privacy-Enhancing Policy

An intuitive and effective measure to enhance privacy is employing end-to-end encryption on original message plaintext. This measure transforms the original plaintext into ciphertext, which can be seen as a random mapping from plaintext space to ciphertext space. The "random mapping" makes it more difficult for adversaries to launch content-analysis attacks successfully, since the ciphertext gives no specific knowledge to those without decryption keys. Generally, light-weight symmetric-key cryptosystems can be chosen as the end-to-end encryption. In addition, these encryption and decryption operations are performed only at the two ends of a session, which brings no additional overhead to other nodes in the network.

In reality, if the original message is in plaintext format, the number *k* of original message sets may not be large enough to resist the above backward deduction attacks. The proposed measure is to employ end-to-end encryption on message content, which brings more "confusion" to adversaries. Moreover, with the change of session keys, the original messages will be mapped to the whole ciphertext space. Thus, it can be regarded that the message set number k_e be the size of the whole ciphertext space, i.e., $k_e=|F|^{h\cdot l}$.

If using privacy entropy [24] to measure the privacy enhancement from end-to-end encryption, we can obtain:

$$\begin{aligned}\Delta E_p &= E_p(M_e) - E_p(M) \\ &= (-\textstyle\sum_{i=1}^{k_e} p^e{}_i \log p^e{}_i) - (-\sum_{i=1}^{k} p_i \log p_i)\end{aligned} \quad (16)$$

where M_e and M denote the anonymity set with and without end-to-end encryption, respectively, and p_i^e and p_i denote the appearance probability of element *i* in the corresponding anonymity set, respectively. If each message set appears with the same probability, i.e., $p_i^e = p_j^e$ $(i \neq j)$ and $p_i = p_j$ $(i \neq j)$, the privacy enhancement can be computed as follows in terms of privacy entropy:

$$\begin{aligned}\Delta E_p &= \log|F|^{h\cdot l} - \log k \\ &= h\cdot l\cdot \log|F| - \log k\end{aligned} \quad (17)$$

In practice, since the number *k* is much smaller than k_e, the increased entropy of privacy is considerably significant. With the increase of privacy entropy, the size of message space will increase exponentially. Thus, this measure greatly enhances the privacy of the proposed scheme since it is difficult to traverse the message space.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed scheme in term of invertible probability and computational overhead.

A. Invertible Probability

Firstly, let each element of a LEV be randomly chosen from the field $\mathbb{F}_q$. The following theorems hold.

Theorem 3: For a Local Encoding Matrix (LEM), which is comprised of *h* LEVs with *h* elements each LEV and each element is from the finite field $\mathbb{F}_q$, the invertible probability of a GEM (also own *h* vectors) will be degraded by $s_q = \prod_{i=1}^{h}(1-q^{-i})$ after coding.

Proof: See Appendix D. ■

Corollary 2: The invertibility factor s_q imposed by a $h\times h$ LEM can be approximated to $1-q^{-1}-q^{-2}$ when $h\geqslant 4$, and the error of this approximation is within the magnitude of $\boldsymbol{O}(q^{-5})$.

This corollary can be easily proofed by expanding the multiplication of the polynomials. This corollary gives two important implications. Firstly, in practical network coding, the min-cut capacity *h* is much larger than the condition required in ***corollary 2***. That is, this corollary can be safely used in practical network coding. Secondly, the field size *q* is relatively a large number. Therefore, an amount in the magnitude $\boldsymbol{O}(q^{-5})$ is really a small amount, which can be omitted.

Based on ***Theorem 3*** and ***Corollary 2***, the invertible probability of a GEM can be easily calculated. For example, a network coding system with a min-cut capacity h $(h \geq 4)$ owns the invertible probability of $(1-q^{-1}-q^{-2})^t$, where *q* is the field size and *t* is the total encoding number from the source to sinks. In practical network coding, since *q* is a relatively large prime number, the above invertible probability can be further approximated to $1-tq^{-1}$.

If the Paillier cryptosystem is chosen as the homomorphic encryption method, the invertible probability will differ from the result in ***Theorem 3***. In the Paillier cryptosystem, the element is chosen from a ring $\mathbb{R}_n$, where $n = pq$ and *p*, *q* are two prime number.

Theorem 4: For a LEM, which is comprised of *h* LEVs, where the elements are randomly chosen from a ring $\mathbb{R}_n$ ($n = pq$), the invertible probability of a GEM (also with *h* vectors) will be degraded by $s_p + s_q - s_n$ after coding.

Proof: See Appendix E. ■

According to ***Corollary 2***, s_p, s_q, s_n, and the integral invertibility factor can be approximated to $1-p^{-1}-p^{-2}$, $1-q^{-1}-q^{-2}$, $1-n^{-1}-n^{-2}$, and $1-p^{-1}-p^{-2}-q^{-1}-q^{-2}+n^{-1}$, respectively, with the error in the magnitude of $\boldsymbol{O}(p^{-5}+q^{-5})$.

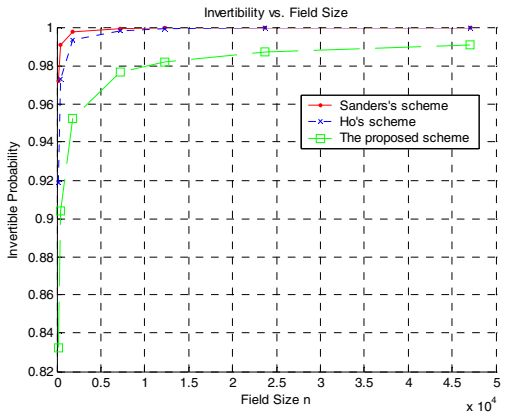

Fig. 7. Invertible Probability versus the Size of the Element Space

If $|p| \approx |q|$, the integral invertibility factor can be approximately reduced to $1-p^{-1}-q^{-1}$, with the error confined in $\boldsymbol{O}(p^{-2}+q^{-2})$. If a session performs totally t times of random encoding, the invertible probability of GEVs at sinks is $(1-p^{-1}-q^{-1})^t$. In practical network coding, since p and q are relatively large prime numbers, the above invertible probability can be approximated to $1-t(p^{-1}+q^{-1})$. From the above analytic results, it can be seen that the invertible probability is independent of the number of sinks, but dependent on the total encoding numbers.

Compared with the random coding schemes in [2] and [18], the invertible probability of the proposed scheme appears a bit less, as shown in Fig. 7. The difference can be regarded as the overhead from privacy preservation. Fortunately, the proposed scheme can still maintain a high invertible probability especially when the two prime numbers are relatively large.

According to the above analysis, the invertible probability will decrease along with the increase of the total times of random coding. The simulation results in Fig. 8 match the analytical results pretty well, thus demonstrate the validity of the performance analysis. The analysis and simulation results of the original random coding scheme are also shown in the same figure for comparison.

B. *Computational Overhead*

The computational overhead of the proposed scheme can be investigated respectively from three aspects, including source encoding, intermediate recoding, and sink decoding. Since the computational complexity of the proposed scheme is closely involved with the specific homomorphic encryption algorithm, in the following analysis, we will take the Paillier cryptosystem as the encryption method when necessary.

Source Encoding Overhead: Consider h GEVs with h elements in each GEV, which form an $h \times h$ GEM. After source encoding, every element in the GEM can be encrypted one by one. Thus, the computational overhead is $\boldsymbol{O}(h^2)$ in terms of encryption operations. Every encryption operation requires 2 exponentiations, 1 multiplication, and 1 modulus operation in the Paillier cryptosystem. Therefore, the computational complexity is $\boldsymbol{O}(h^2 \cdot \log n)$ in terms of multiplication operations.

Intermediate Recoding Overhead: In intermediate nodes, linear transformation on the elements of GEVs can only be performed by manipulating the ciphertext of these elements because intermediate nodes have no knowledge of decryption keys. According to Eq. (8), the computational complexity of producing one element in new GEVs is h exponentiations and $h-1$ multiplications on the ciphertext, which is $\boldsymbol{O}(h \cdot \log n)$ in terms of multiplications together. Thus, the computational complexity is $\boldsymbol{O}(h^2 \cdot \log n)$ for a GEV and $\boldsymbol{O}(h^3 \cdot \log n)$ for a GEM with h GEVs in terms of multiplication operations.

Sink Decoding Overhead: After receiving an encoded message, a sink can decrypt the elements in the GEV. According to the Paillier cryptosystem, decrypting an element requires 1 exponentiation, 1 multiplication, and 1 division operation. Therefore, the computational complexity of decrypting a GEV is $\boldsymbol{O}(h \cdot \log n)$ in terms of multiplication operations.

After receiving h messages, which requires $\boldsymbol{O}(h^2 \cdot \log n)$ multiplication operations to decrypt the h GEVs, a sink node can start to check the linear dependence of the GEVs. A method is the Gaussian elimination algorithm, which requires $\boldsymbol{O}(h^3)$ multiplication operations.

If h GEVs are linearly independent, we can further derive the inverse of the corresponding GEM. Based on Gaussian elimination, the computational complexity to find the inverse of a matrix is $\boldsymbol{O}(h^3)$ in terms of multiplication operations. With the inverse of a matrix, a sink can recover the original messages by decoding the encoded messages. The computational complexity for the recovery is $\boldsymbol{O}(h^2 \cdot l)$ in terms of multiplication, where l is the length of the messages.

In summary, the computational complexity for a sink to decode h messages is $\boldsymbol{O}(h^2 \cdot \log n)$ or $\boldsymbol{O}(h^2 \cdot l)$ in terms of multiplication operations, because normally $l \gg h$ and $n \gg h$.

VI. Related Works

Privacy preservation over traditional routing-based networks has been well investigated, and many efficient schemes have been proposed, which can be classified into three categories: proxy-based, mix-based, and onion-based ones. Proxy-based schemes include Crowds [11] and Hordes [12]. The common characteristic of these schemes is to employ one or more network nodes to issue service requests on the behalf of the originator. In Crowds, for example, servers and even crowd members can not distinguish the originator of a request, since it is equally likely to originate from any member of the crowd.

Chaum's mix based schemes include MorphMix [13] and Mixminion [14]. The common feature of these schemes is to employ techniques such as shaping, which divides messages into a number of fixed-sized chunks, and mixing, which caches incoming messages and then forwards them in a randomized order. These two techniques can be used to prevent attacks such as size correlation and time-order correlation.

Onion-based schemes include Onion Routing [15] and Onion Ring [16]. The common feature of this kind of schemes is employing the technique chaining, where onion routers are chained together to forward messages hop by hop to the intended recipient. The characteristic of this technique is that every intermediate onion router only knows about the other two routers directly in front of and behind itself, respectively, which can keep user privacy from being compromised if one or even several intermediate onion routers are compromised.

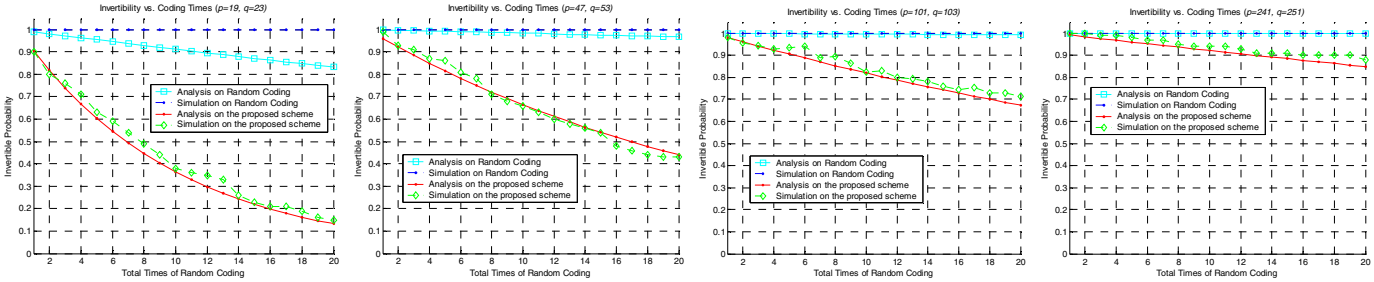

Fig. 8. Invertible Probability vs. Total Times of Random Coding (a. $p=19$, $q=23$; b. $p=47$, $q=53$; c. $p=101$, $q=103$; d. $p=241$, $q=251$)

However, network coding may face potential security threats, which include two primary types of attacks, *pollution attacks* [10] and *entropy attacks* [9]. *Pollution attacks* may be launched by untrusted nodes or adversaries through injecting polluted messages or modifying disseminated messages, which can be fatal to the whole network due to the rapid propagation of pollution. In *entropy attacks*, adversaries forge non-innovative packets that are linear combinations of "stale" ones, which may reduce the overall network throughput.

To secure network coding, some solutions have been proposed and they can be divided into two categories according to different theoretical bases. Information-theory based schemes [8] can only detect or filter out polluted messages at sinks, not at forwarders. Cryptography-based solutions include schemes based on homomorphic hash functions [9], homomorphic signatures [10], and secure random checksum [9]. These solutions either require an extra secure channel [9], or incur high computation overhead [10].

In summary, existing studies on secure network coding mainly focus on detecting or filtering out polluted messages [10]. Few stress on the privacy issues, especially on how to protect the encoded messages from tracking or traffic analysis.

VII. Conclusions

In this paper, we first identified the potential privacy issues, such as traffic secrecy and source anonymity which may be compromised by traffic analysis and flow tracing attacks, in network coding enabled networks. Then we proposed an efficient privacy-preserving scheme against traffic analysis for network coding. With the light-weight homomorphic encryption on Global Encoding Vectors (GEVs), the proposed scheme offers two significant privacy-preserving features, packet flow untraceability and message content confidentiality, which can efficiently thwart the traffic analysis attacks such as flow tracing. Moreover, with homomorphic encryption operations, the proposed scheme keeps the essence of random linear network coding, and each sink can recover the source messages by inverting the GEVs with a very high probability. The quantitative analysis on privacy enhancement and computational complexity, as well as the simulative evaluation, demonstrates the effectiveness and efficiency of the proposed scheme. Our future work includes designing more efficient schemes for privacy preservation in network coding enabled wireless networks.

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] T. Ho, M. Medard, 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, 2006.

[3] Z. Li, B. Li, and L.C. Lau, "On Achieving Maximum Multicast Throughput in Undirected Networks", *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2467-2485, Jun. 2006.

[4] 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.

[5] P. A. Chou and Y. Wu. "Network Coding for the Internet and Wireless Networks", MSR-TR-2007-70, Microsoft Research, Jun. 2007.

[6] M. Wang and B. Li, "Network Coding in Live Peer-to-Peer Streaming", *IEEE Trans. On Multimedia*, Vol. 9, No. 8, pp. 1554-1567, 2007

[7] E. Ayday, F. Delgosha, and F. Fekri, "Location-Aware Security Services for Wireless Sensor Networks Using Network Coding", *Proc. IEEE INFOCOM'07*, pp. 1226-1234, 2007.

[8] K. Han, T. Ho, R. Koetter, M. Medard, and F. Zhao, "On Network Coding for Security", *Proc. IEEE MILCOM'07*, pp. 1-6, 2007

[9] C. Gkantsidis, P. Rodriguez Rodriguez, "Cooperative Security for Network Coding File Distribution", *Proc. IEEE INFOCOM'06*, pp. 1-13, 2006

[10] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An Efficient Signature-based Scheme for Securing Network Coding against Pollution Attacks", *Proc. of IEEE INFOCOM*, 2008.

[11] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions", *ACM Trans. on Information and System Security*, vol. 1, no. 1, pp. 66–92, Nov. 1998.

[12] C. Shields and B. N. Levine, "A Protocol for Anonymous Communication over the Internet", *Proc. of ACM CCS'00*, pp. 33–42, 2000.

[13] M. Rennhard and B. Plattner, "Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection", *Proc. of the ACM Workshop on Privacy in the Electronic Society*, pp. 91–102, 2002.

[14] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol", *Proc. of the 2003 IEEE Symposium on Security and Privacy*, pp. 2–15, May 2003.

[15] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing for Anonymous and Private Internet Connections", *Communications of the ACM*, Vol. 42, No. 2, pp. 39–41, Feb. 1999.

[16] X. Wu and N. Li, "Achieving Privacy in Mesh Networks", *Proc. of the fourth ACM workshop on Security of Ad hoc and Sensor Networks (SASN'06)*, pp. 13-22, 2006.

[17] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," *Proc. of 51st Allerton Conf. Communication, Control and Computing*, Oct. 2003.

[18] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial Time Algorithms for Network Information Flow", *Proc. of 15th ACM symposium on Parallel Algorithms and Architectures* (SPAA'03), pp. 286-294, 2003.

[19] J. Benaloh, "Dense Probabilistic Encryption", *Proc. of the Workshop on Selected Areas in Cryptography*, pp. 120-128, 1994.

[20] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuocity Classes", *Proc. of EUROCRYPT'99*, LNCS, vol. 1592, pp. 223-238, 1999.

[21] P. Venkitasubramaniam and L. Tong, "Anonymous Networking with Minimum Latency in Multihop Networks", *Proc. IEEE Symposium on Security and Privacy*, pp. 18-32, 2008.

[22] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards Statistically Strong Source Anonymity for Sensor Networks", *Proc. IEEE INFOCOM'08*, pp. 51-55, 2008.

[23] X. Lin, R. Lu, H. Zhu, P.-H. Ho, X. Shen and Z. Cao, "ASRPAKE: An Anonymous Secure Routing Protocol with Authenticated Key Exchange for Wireless Ad Hoc Networks", *Proc. IEEE ICC'07*, 2007.
[24] T. Jiang, H. J. Wang, and Y.-C. Hu, "Preserving Location Privacy in Wireless LANs", *Proc. ACM MobiSys'07*, pp. 246-257, 2007.
[25] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Trans. on Information Theory*, vol. IT-22, pp. 644-654, 1976.
[26] Y. Challal and H. Seba, "Group Key Management Protocols: a novel taxonomy," *International Journal of Information Technology,* vol. 2, pp. 105-119, 2005.

APPENDIX

A. Proof of Theorem 1.

Proof: From the theory of modulo congruence, for $n>1$, the mapping $f:\mathbb{Z}\mapsto\mathbb{Z}_n$ defined by $(\mathrm{mod}\ n)$ is a homomorphic mapping in terms of the addition, subtraction, and multiplication operations. Therefore, for each solution to $|A|x_i=|\tilde{A}_i|(\mathrm{mod}\ n)$, there must be one or more solutions to $|A|x_i=|\tilde{A}_i|+k\cdot n(k\in\mathbb{Z})$.

According to the theory of linear algebra, the necessary and sufficient condition for $|A|x_i=|\tilde{A}_i|+k\cdot n(k\in\mathbb{Z})$ to have a unique solution is $|A|\neq 0$. Thus, the necessary condition for a system of linear congruence equations to have a unique solution is $|A|\neq 0$. The original proposition is proved. ■

B. Proof of Theorem 2.

Proof: According to **Theorem 1**, $|A|\neq 0$ is a necessary condition for a system of linear congruence equations to have a unique solution. Based on this condition, the original system of linear congruence equations can be transformed to $|A|x_i=|\tilde{A}_i|(\mathrm{mod}\ n)(i=1,2,\cdots,h)$ by only applying addition and multiplication operations, which are preserved by the homomorphic mapping $f:\mathbb{Z}\mapsto\mathbb{Z}_n$ defined by $(\mathrm{mod}\ n)$.

In the system of equations $|A|x_i=|\tilde{A}_i|(\mathrm{mod}\ n)$, variables $x_i\ (i=1,2,\cdots,h)$ can be solved independently from each other. Thus, for every independent equation $|A|x_i=|\tilde{A}_i|(\mathrm{mod}\ n)$, solutions can be solved by employing the theory of linear congruence equations.

A linear congruence equation $ax=b(\mathrm{mod}\ n)$ is solvable if and only if the congruence $b=0(\mathrm{mod}\ d)$ with $d=\gcd(a,n)$ is solvable, where $\gcd(a,n)$ denotes the greatest common divisor of the two operands a and n. Let one solution to the original equation be $x_0<n/d=s$. Then the solutions are $x=x_0,x_0+s,\cdots,x_0+(d-1)s$. If $d=1$, then there is only one unique solution which is less than n.

According to the theory of linear congruence equations, if $|\tilde{A}_i|\equiv 0(\mathrm{mod}\ d)\ (i=1,2,\cdots,h)$, then every equation $|A|x_i=|\tilde{A}_i|(\mathrm{mod}\ n)$ has d solutions. The solutions to the system of linear congruence equations are the combination of these independent solutions to every single congruence equation. The combination number is d^h. Thus, the proposition is proved. ■

C. Proof of Corollary 1.

Proof: From **Theorem 2**, if $d=\gcd(|A|,n)=1$, the system of linear congruence equations has $d^h=1$ solution, which is also the unique solution to original linear system. In addition, according to the congruence theory, when $\gcd(|A|,n)=1$, the modular inverse of the integer $|A|$, which is denoted as $|A|^{-1}$, can be calculated using the extended Euclidean algorithm. The result satisfies the following equation: $|A|^{-1}|A|=1(\mathrm{mod}\ n)$. The unique solution to the original system of linear congruence equations is $x_i=|A|^{-1}|\tilde{A}_i|(\mathrm{mod}\ n)$ $(i=1,2,\cdots,h)$. In matrix form, the result is $|A|^{-1}|A|X=X=|A|^{-1}A^*Y(\mathrm{mod}\ n)$. Thus the original corollary is proved. ■

D. Proof of Theorem 3.

Proof: The invertibility factor imposed by any *h* LEVs depends only on the linear dependence of the *h* LEVs themselves.

Firstly, the elements in the first LEV can be any combination except for all zeros. Therefore, the invertibility factor imposed by the first LEV is $1-q^{-h}$.

For the second LEV, it is required to consider the linear dependence on the first one, where the all-zero situation is included. Thus, the invertibility factor imposed by the second LEV is $1-q/q^h=1-q^{1-h}$.

For the third LEV, considering the linear dependence on the former two ones, the invertibility factor is $1-q^2/q^h=1-q^{2-h}$.

Similarly, for the remaining LEVs, the invertibility factors are $1-q^{3-h}$, $1-q^{4-h}$, $\cdots$, $1-q^{-1}$, respectively.

Thus, the integrated invertibility factor imposed by the whole LEM can be calculated by multiplying these individual factors together as follows:

$$s_q=\prod_{i=1}^{h}(1-q^{-i}).$$

Thus, the original proposition is proven. ■

E. Proof of Theorem 4.

Proof: The problem can be decomposed into two separate sub-problems in terms of the prime number *p* and *q*, respectively. As for the sub-problem in terms of the prime number *p*, there is a mapping from the original problem to the problem modulo *p*. According to ***Theorem 3***, the imposed invertibility factor is $s_p=\prod_{i=1}^{h}(1-p^{-i})$. Similarly, the invertibility factor imposed by the sub-problem of prime number *q* is $s_q=\prod_{i=1}^{h}(1-q^{-i})$.

The above two sub-problems have overlap, which happens on these points where the number is congruent to zero modulo *n*. The invertibility factor related to the overlap area is $s_n=\prod_{i=1}^{h}(1-n^{-i})$. According to the union principle of the set theory, the integral invertibility factor can be calculated as follows:

$$1-((1-s_p)+(1-s_q)-(1-s_n))=s_p+s_q-s_n.$$

Thus, the original proposition is proven. ■