ISSN 2278-7763

Network Coding Based Privacy Preservation using Blowfish Encryption against Traffic Analysis in Multi-hop Wireless Networks

Vrushali Dongre

***Abstract*- Attacks such as traffic analysis can be easily launched by a malicious adversary due to the open-air transmission. So, Privacy threat becomes one of the critical issues in multi-hop wireless networks. To avoid this, network coding can be used to prevent traffic analysis attacks since the coding/mixing operations are allowed to perform at intermediate nodes. The simple deployment of network coding is not only sufficient. It also requires the coding/mixing nature for employing the existing privacy-preserving techniques, such as Onion Routing, in network coding enabled networks. In this paper, we propose a novel network coding based privacy-preserving scheme against traffic analysis in multi-hop wireless networks which is different from any existing solutions. Along with Blowfish encryption on Global Encoding Vectors (GEVs), the proposed scheme offers two significant privacy-preserving features, packet flow untraceability and message content confidentiality, for efficiently preventing the traffic analysis attacks. The proposed scheme provides the random coding feature, and each sink can recover the source packets by inverting the GEVs with a very high probability. The validity and efficiency of the proposed scheme can be demonstrated by Theoretical analysis and simulative evaluation.**

Keywords- Traffic analysis, Privacy preservation, Network coding, Blowfish encryption.

I. INTRODUCTION

Wireless networks such as Wi-Fi provides benefits such as convenience, mobility, and low cost so, it has been widely deployed in the access network area. However, they still suffer from some drawbacks such as limited radio coverage, poor system reliability, as well as lack of security and privacy. Multi-hop Wireless Networks (MWNs) is a promising solution for extending the radio coverage range of the existing wireless networks. In multi-hop wireless networks, communication between two end nodes is carried out through a number of intermediate nodes whose function is to relay information from one point to another. Through Multi-path packet forwarding, system reliability can be enhanced, which is feasible in MWNs.

MWNs provide many security and privacy issues. Still MWNs can suffer from various kinds of attacks, such as eavesdropping, data modification/injection, and node compromising due to the open-air wireless transmission. Security properties such as confidentiality, integrity, and authenticity of MWNs can be broken by these attacks. To compromise the privacy of user some advanced attacks such as traffic analysis and flow tracing can also be launched including source anonymity and traffic secrecy. Our focus is on privacy preservation issues issue, i.e., how to prevent traffic analysis/flow tracing and achieve source anonymity in MWNs.

Among all privacy properties of MWNs, source anonymity is of special interest. Source anonymity provides communication through a network without confessing the identity or location of the source node. Preventing traffic analysis/flow tracing and provisioning source anonymity is critical for securing MWNs, especially military related, such as sensor or tactical, networks. Consider a simple example of multicast communication in military ad hoc networks (which can be regarded as a kind of MWNs), where through multi-hop packet forwarding the nodes can communicate with each other. If attacker able to block the packets and trace to the source with traffic analysis then there is possibility of disclosure of some sensitive information such as the location of some critical, important nodes (e.g., the commanders) and further damage to the location privacy. Afterwards, the attacker can take some critical actions to launch *Decapitation Strike* to destroy these critical nodes [16].

A very challenging issue is MWNs is how to efficiently prevent traffic analysis and flow tracing attacks and provide privacy protection. Some existing privacy preservation solutions, such as proxy-based schemes [2], [10], Chaum's mix-based schemes [1], [2], and onion-based schemes [6], [20], may either require a series of trusted forwarding proxies or result in severe performance degradation in practice.

Network coding was first introduced by Ahlswede et al [11]. Afterwards, the two key coding techniques, random coding [9] and linear coding [18], further boost the development of network coding technologies. The random coding is more practical, while the linear coding is proven to be sufficient and computationally efficient for network coding. Today,

ISSN 2278-7763

network coding has emerged as one of the most promising information dissemination approach to improving network performance. Some of the applications of network coding include the file distribution and multimedia streaming on P2P overlay networks [12], data transmission in sensor networks [20], tactical communications in military networks [13], etc. As compared with conventional packet forwarding technologies, network coding offers, by allowing and encouraging coding/mixing operations at intermediate forwarders [11], several significant advantages such as the potential throughput improvement [8], transmission energy minimization [7], and delay minimization [4].

The deployment of network coding in MWNs provides performance benefits as well as a possible way to efficiently prevent the traffic analysis and flow tracing attacks as the coding/ mixing operation is carried out at intermediate nodes. Network coding provides an inherent message mixing mechanism, which suggests that privacy preservation can be efficiently achieved in a distributed manner, which is similar to the Chaum's mix-based schemes [1]. An important property for preventing traffic analysis/flow tracing, which is the unlinkability between incoming packets and outgoing packets can be carried out by mixing the incoming packets at intermediate nodes. Still, the privacy provided by such a mixing of packets is still sensitive, since the earliest decoding is still possible and linear dependence can be easily analyzed. Earliest decoding and traffic analysis cannot be prevent by a simple deployment of network coding considering the explicit Encoding Vectors (GEVs, also known as tags) prefixed in the encoded packets provides indirect access to adversaries to compromise the privacy of users. Once enough packets are collected by attacker, they can easily recover the original packets and then can easily conduct the traffic analysis/flow tracing attacks based on these original packets. By employing link-to-link encryption, this vulnerability can be prevented in a certain level. It introduces heavy overload, thus resulted into significant performance degradation of the whole system. If intermediate nodes are compromised by malicious adversaries, it cannot preserve the privacy of users. Such failures of privacy motivate us to explore an efficient privacy-preserving scheme for MWNs.

The proposed scheme offers the following attractive features: 1) **Enhanced Privacy against flow tracing and traffic analysis:** The employment of HEFs assured the confidentiality of GEVs, which makes difficult for attacker to recover original GEVs. As only sinks know the decryption key, the attacker cannot decrypt the GEVs, even if some intermediate nodes are compromised. The confidentiality of GEVs reflected into confidentiality of message content, because message decoding only depend on the GEVs. 2) **Efficiency**: Due to Homomorphism of HEFs, message recoding can be carried out on encrypted GEVs at intermediate nodes, without decryption keys or performing expensive decryption operations on each incoming packet. The performance evaluation on computational complexity determines the efficiency 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 proven that the influence the HEFs pose upon the invertible probability of GEVs is negligible. Thus, the random coding feature is kept in our network coding based privacy-preserving scheme.

II. PRELIMINARIES

A. Network Coding Model

Network coding allows intermediate nodes to perform computation on input messages, making output messages be the mixture of the input ones, which is different from traditional packet-forwarding systems. This delicate principle provides plenty of surprising opportunities, such as random coding [2]. As shown in Fig. 2, whenever there is a transmission opportunity on an outgoing link, an outgoing packet is formed by taking a random combination of packets in the current buffer.

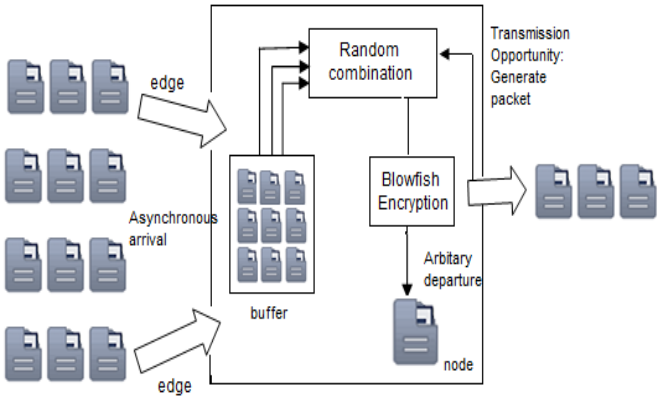

Fig1. Random coding at intermediate nodes

An overview of network coding and possible applications has been given in [5], and packet tagging and buffering are key for practical network coding. . In practical network coding, source information should be divided into blocks such that each block contains *h* blocks. All coded packets related to the *k*th block belong to generation *k* and random coding can be only performed among the packets in the same generation. Packets within a generation need to be synchronized by buffering for the purpose of network coding at intermediate nodes.

ISSN 2278-7763

Consider an acyclic network (V, E, c) with unit capacity, i.e., c(e) =1 for all e∈E, meaning that each edge can carry one symbol per unit time. Assume that each symbol is an element of a finite field *q*F. Consider a network with multicast sessions, where a session is comprised of one source s∈V and a set of sinks T⊆V (or one single sink $_t$∈V). Let *h* = MinCut(s, T) be the multicast capacity and X1… X_n be the *h* symbols to be delivered from s to T.

For each outgoing edge *e* of a node *v*, let y(e)∈F 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 $\beta(e)=[\beta_{e'}(e)]$ is called as *Local Encoding Vector* (LEV).

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

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

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

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

Each packet can be considered as a vector of symbols $y(e)=[y_1(e),\ldots,y_N(e)]$. By likewise grouping the source symbols into packets $X_i=[X_{i,1,...,}X_{i,N}]$, the above algebraic relationships carry over to packets. To facilitate the decoding at the sinks, each message should be tagged with its GEV ***g***(*e*) , which can be easily achieved by prefixing the i^{th} source packet X_i with the i^{th} unit vector u_i. Then, each packet is automatically tagged with the corresponding GEV, since

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

With the help of tags, GEV can be found within packets themselves, so that the sinks can compute Gt without knowing the network topology or packet-forwarding paths. Nor side channel is required for the communication of Gt. As nodes can be added or removed in an ad hoc way, the network can be dynamic.

B. Blowfish Encryption Function

Blowfish Encryption Functions have the property of homomorphism. Homomorphism means operations on plaintext can be performed by operating on corresponding cipher text. For example, suppose *E*(.) 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*(.) needs to satisfy the following properties:

1) **Additivity**: Given the cipher text, *E*(*x*) and *E*(*y*), there exists a computationally efficient algorithm Add(.,.) 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(.,.) such that *E*(*t* . *x*) = Mul(E(x),t).

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] are such two additive HEFs cryptosystems, where the addition on plaintext can be achieved by performing a multiplicative operation on the corresponding cipher text, i.e., $E(x_1+x_2)= E(x_1) \cdot E(x_2)$. Further, the following two equations can be easily derived.

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

C. Privacy Measurement

Currently, anonymity set and privacy entropy are two main methods of privacy management. In the anonymity set method, members in an anonymity system is assumed to be uniformly distributed, i.e., all members are equally possible to be the target user in probability from the perspective of adversaries. The privacy entropy method can measure the privacy degree more accurately for those systems where the members are not uniformly distributed. Following Shannon's classic definition of information entropy, the entropy of privacy $E_p(S)$ can be defined as:

$$E_p(S) = -\sum_{i=1}^{n} p(x_i)\log p(x_i)$$

where

$$S = \{x_1, x_2, x_3, \cdots, x_i, \cdots, x_n\}$$
$$p(x_i) \text{ is the prob. of being } x_i$$
$$\sum_{i=1}^{n} p(x_i) = 1$$

Some common characteristics such as continuity, symmetry, and additivity re shared by privacy entropy with information entropy. For example, continuity corresponds to the change in the value of

ISSN 2278-7763

one of the probabilities by a very small amount should change the entropy by same amount. Symmetry corresponds to, if we interchanged the probability of two objects then privacy entropy should not be suffer. It must have to remains same. We will use the above two approaches to measure the enhanced privacy of the proposed scheme.

III. THE PROPOSED NETWORK CODING BASED PRIVACY-PRESERVING SCHEME FOR MWNs

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

A. The Proposed Privacy-Preserving Scheme

Though we are providing intrinsic mixing mechanism, original coding network enable to provide privacy guarantee due to explicit GEV's, because an attacker can recover the original message as enough packets are collected by him. Link- to-Link encryption is sensitive for inside attacker as they compromise several intermediate models& obtain secrete keys.

The optimal way to resolve this issue is to keep GFV's confidential to intermediate nodes by encryption message in end-to-end manner so that inside attacker unable to compromised intermediate nodes recovering the original messages. But in end-to-end approach "mixing" feature of network coding is disable so this approach is unable to prevent the adversaries from tracking cipher text message.

To solve this issue, we have employed Paillier cryptosystem to apply encryption to GEF's HEF provides confidentiality of GEV's as well as enable intermediate nodes to efficiency mix the coded messages. Given a message m & public key (n, g), the encryption function in Paillier cryptosystem is described as follows:

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

Where r = random factor in Paillier cryptosystem. $E(m)$ satisfies the following Blowfish property:

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

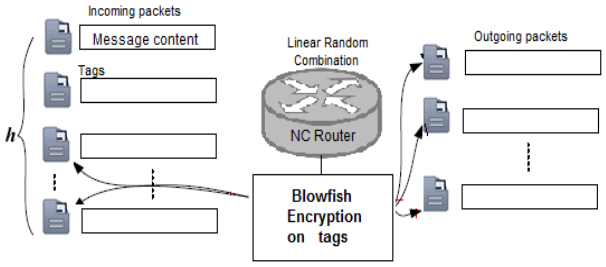

Fig.2 Blowfish encryption on packet tags

In HEF's intermediate nodes are allowed to perform linear coding /mixing operations on the coded messages encrypted tags as shown in fig(4). Due to Blowfish nature of HEF without knowing the decryption keys or performing decryption operations, linear network coding can be encoded messages & the cipher text of GEV's.

Our proposed system consist of three phases as source encoding, intermediate random linear recording & sink decoding. To prevent loss of generality, we assume that each sink acquire two keys, the encryption key e_k & decryption key d_k from an offline Trust Authority(TA) and encryption key e_k is provided to all other nodes. To support multicasting, we obtained a group of sinks from TA or negotiate the key pair in advance [10]; then they can decryption key private in the group.

Source Encoding: Consider that a source has *h* messages, say $x_{1,\ldots,}x_n$, to be sent out. The source first prefixes *h* unit vectors to the *h* messages, respectively. After tagging, the source can choose a random LEV and then perform a linear encoding operation on these messages. Thus, one LEV will generate an encoded message with the GEV (which is equal to the LEV temporarily) tagged.

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

$$c_i(e) = E_{ek}(g_i(e)), (1 \le i \le h)$$
$$c(e) = [c_1(e), c_2(e), \cdots, c_h(e)]$$

Where, the notation e_k denotes the encryption key.

Intermediate Random Linear Recoding: Once some packets of the same generation are received, an intermediate node can perform random linear coding on these packets. To generate an outgoing packet, firstly, a random LEV $[\beta_{1,\ldots,}\beta_h]$ is chosen independently; then, a linear combination of message content of the incoming packets is computed as the message content of the outgoing packet.

Since all tags of h incoming packets are in cipher text, intermediate node has no knowledge of decryption keys so it is very difficult to perform further operations such as earliest decoding to get the original message content. Due to Blowfish nature of encryption function, linear transformation can be carried out on encrypted tags of incoming packets to find out tags for the outgoing packet,

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

By utilizing the Blowfish characteristic of the encryption on GEVs, the ciphertext of the new GEVs for outgoing packets can be calculated as follows:

ISSN 2278-7763

$$E_{ek}(g(e)) = E_{ek}(\sum_{i=1}^{h} \beta_i(e) g(e'_i)) = \prod_{i=1}^{h} E_{ek}(\beta_i(e) g(e'_i)) . = \prod_{i=1}^{h} E_{ek}^{\beta_i(e)}(g(e'_i))$$

Without decryption keys, with the help of the ciphertext of GEVs of incoming packets, the ciphertext of new GEVs can be computed. Finally, the ciphertext of a new GEV is prefixed to the corresponding message content to form a new outgoing packet, which is sent out to downstream nodes.

Sink Decoding: After receiving a packet, the sink first decrypts the packet tag using the corresponding decryption key d_k.

$$g_i(e) = D_{dk}(c_i(e)) \quad (1 \le i \le h)$$

$$\boldsymbol{g}(e) = [g_1(e), g_2(e), \cdots, g_h(e)]$$

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

$$\boldsymbol{G}^{-1} \cdot \boldsymbol{G} = \boldsymbol{U} \pmod n$$

$$\boldsymbol{G} = [\boldsymbol{g}(e_1), \boldsymbol{g}(e_2), \cdots, \boldsymbol{g}(e_h)]^T$$

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

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

B. Invertibility of a GEM

Let GEM *A* be comprised of *h* GEVs with *h* elements in 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.

To solve a system of linear congruence equations, Gaussian elimination can be used. 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:

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

If and only if every independent equation is solvable then only we can solve a system of linear congruence. 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*. A system of linear congruence equations as:

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

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

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 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. The generation numbers are 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, making size correlation useless to adversaries. The inherent buffering technique [5] of network coding can significantly improve the resistance to time-order correlation.

The third protection is to eliminate message content correlation, which is also commonly used in traffic analysis and flow tracing. Assume that the first protection on generation numbers is compromised, due to the vulnerability of a secure routing system. To launch content correlation based traffic analysis, adversaries must compromise several forwarders to intercept packets of the same generation. Let the number of packet generations going through these compromised nodes be *w* during a period of time *v*. To determine if an intercepted packet, say *md*, 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 of the same generation, as shown in Fig. 6. The independence can be tested using Gaussian elimination on the *h* vectors. The computational complexity of this step is $O(h3)$ in terms of multiplication operations.

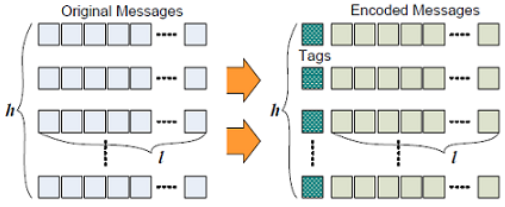

Fig. 3 Message Content Encoding

ISSN 2278-7763

2) The second step 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 computational complexity of this step is **O**(h2) in terms of multiplication.

3) The third step is to verify if the resultant coefficient vector satisfies all the other l − h columns of code symbols. The computational complexity is **O**(h(l − h)) in terms of multiplication.

The whole computational complexity to check if an objective message is the linear combination of a generation of messages is ***O*** (*h*3 + *h* . *l*) in terms of multiplication. To check *w* generations, the computational complexity is ***O*** (*w*.(*h*3 + *h* . *l*)) in terms of multiplication.

The probability of false positive verification is considerably low. Because the correlation between messages of the same generation is uncertain or unknown, it is reasonable to assume that the other (*l*-*h*) columns of code symbols are randomly distributed in the field *F*. Thus the probability of false positive verification is $P_{fpv}(h, 1)=|F|^{h-1}$, which is pretty small since $l >> h$.

B. Privacy-Enhancing Policy

An intuitive and effective measure to enhance privacy is employing end-to-end encryption on original message plaintext. 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. Light-weight symmetric-key cryptosystems can be chosen as the end-to-end encryption method.

If privacy entropy [17] is used to measure the privacy enhancement from end-to-end encryption, we can obtain:

$$\Delta E_p = E_p(M_e) - E_p(M)$$
$$= (-\sum_{i=1}^{k_e} p^e_i \log p^e_i) - (-\sum_{i=1}^{k} p_i \log p_i)$$

where *Me* and *M* denote the anonymity set with and without end-to-end encryption, respectively, and P^e_i 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.$P^e_i = P^e_j$ (i≠j) and $P_i=P_j$ (i≠j), the enhanced privacy entropy can be computed as follows:

$$\Delta E_p = \log|F|^{h \cdot l} - \log k$$
$$= h \cdot l \cdot \log|F| - \log k$$

V. PERFORMANCE EVALUATION AND OPTIMIZATION

A. Adding Blowfish Encryption technique

Blowfish is a symmetric encryption algorithm, meaning that it uses the same secret key to both encrypt and decrypt messages. Blowfish is also a block cipher[5], meaning that it divides a message up into fixed length blocks during encryption and decryption. The block length for Blowfish is 64 bits; messages aren't a multiple of eight bytes in size must be padded.

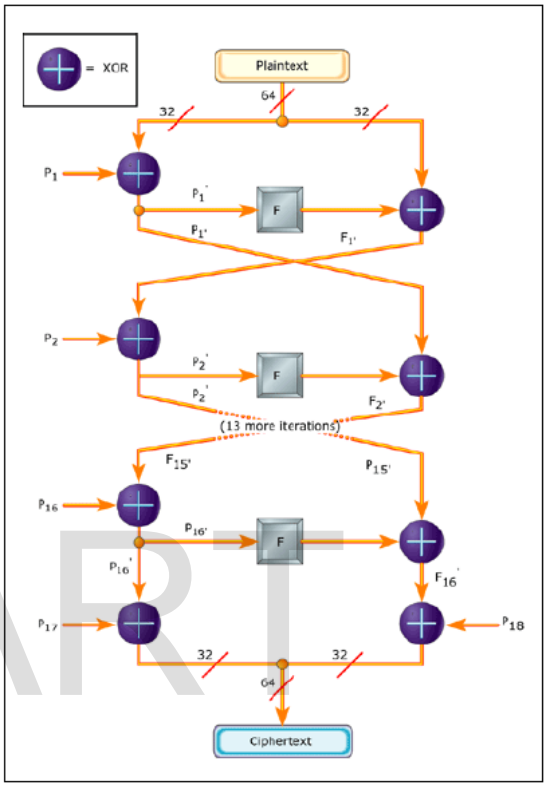

Fig.4 Blowfish algorithm

Blowfish consists of two parts: key-expansion and data encryption. During the key expansion stage, the inputted key is converted into several sub key arrays total 4168 bytes. There is the P array, which is eighteen 32-bit boxes, and the S-boxes, which are four 32-bit arrays with 256 en-tires each. After the string initialization, the first 32 bits of the key are XORed with P1 (the first 32-bit box in the P-array). The second 32 bits of the key are XORed with P2, and so on, until all 448, or fewer, key bits have been XORed. Cycle through the key bits by returning to the begin-ning of the key, until the entire P-array has been XORed with the key. Encrypt the all zero string using the Blowfish algorithm, using the modified P-array above, to get a 64 bit block. Replace P1 with the first 32 bits of output, and P2 with the second 32 bits of output (from the 64 bit block). Use the 64 bit output as input back into the Blowfish cipher, to get a new 64 bit block. Replace the next values in the P-array with the block. Repeat for all the values in the P-array and all the S boxes in order [21].

ISSN 2278-7763

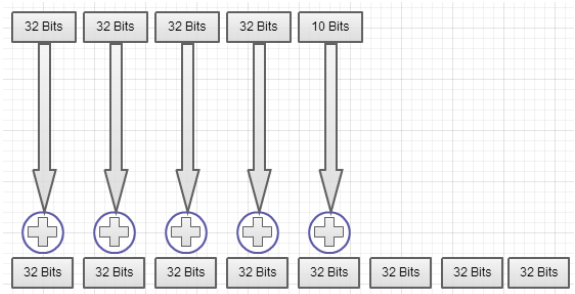

Fig.5. XORing bits once the key has been traversed through once

Encrypt the all zero string using the Blowfish algorithm [1], using the modified P-array above, to get a 64 bit block. Replace P1 with the first 32 bits of output, and P2 with the second 32 bits of output (from the 64 bit block). Use the 64 bit output as input back into the Blowfish cipher, to get a new 64 bit block. Replace the next values in the P-array with the block. Repeat for all the values in the P-array and all the S boxes in order.

VI. PROPOSED SYSTEM

This system basically uses the Blowfish encryption algorithm to encrypt the data file. This algorithm is a 64-bit block cipher with a variable length key. This algorithm has been used because it requires less memory. It uses only simple operations, therefore it is easy to implement. It is a 64 bit block cipher and it is fast algorithm to encrypt the data. It requires 32 bit microprocessor at a rate of one byte for every 26 clock cycles. It is variable length key block cipher up to 448 bits. Blowfish contains 16 rounds. Each round consists of XOR operation and a function. Each round consists of key expansion and data encryption. Key expansion generally used for generating initial contents of one array and data encryption uses a 16 round Feistal network [21].

Plain text and key are the inputs of this algorithm. 64 bit Plain text is taken and divides into two 32bits data and at each round the given key is expanded & stored in 18 p-array and gives 32bit key as input and XORed with previous round data. Functionality is to divide a 32-bit input into four bytes and uses those as indices into an S-array. The lookup results are then added and XORed together to produce the output. At 16th round there is no function .The output of this algorithm should be 64bit cipher text.

Algorithm Steps:

It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookup tables for each round.

Algorithm

Divide x into two 32-bit halves: xL, xR
For i = 1to 32:
xL = XL XOR Pi

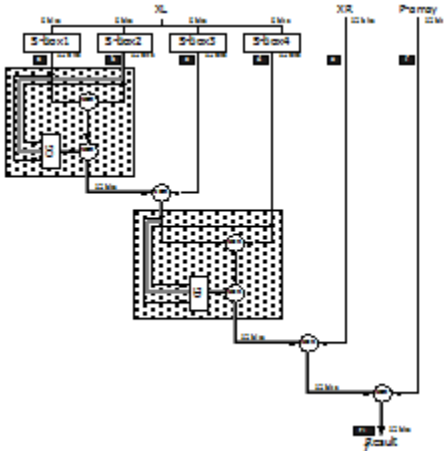

Fig. 6 DFG of loop body

xR = F(XL) XOR xR
Swap XL and xR
Swap XL and xR (Undo the last swap.)
xR = xR XOR P17
xL = xL XOR P18
Recombine xL and xR

For decryption, the same process is applied, except that the sub-keys Pi must be supplied in reverse order. The nature of the Feiestel network [1] ensures that every half is swapped for the next round (except, here, for the last two sub-keys P17 and P18)[21].

VII. CONCLUSIONS

The paper proposed an efficient network coding based privacy-preserving scheme against traffic analysis and flow tracing in multi-hop wireless networks. The proposed scheme offers two significant privacy- preserving features, packet flow untraceability and message content confidentiality, which can efficiently prevent the traffic analysis attacks such as flow tracing, by performing light-weight Blowfish encryption on Global En coding Vectors (GEVs). With Blowfish encryption, the proposed scheme provides random linear network coding and by using very high probability each sink can recover the source messages by inverting the GEVs. The quantitative analysis on privacy enhancement and computational overhead, and the simulative evaluation, demonstrate the effectiveness and efficiency of the proposed scheme. The proposed algorithm of the Blowfish can achieve efficient data encryption up to 4 bits per clock and architecture should satisfy the need of high-speed data encryption. In our future work, we will further improve the performance of the proposed privacy-preserving scheme, such as the invertible probability of GEVs

ISSN 2278-7763

REFERENCES

[1] 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.

[2] 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.

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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

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

[14] 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.

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

[16] 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.

[17] 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.

[18] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance", Proc. IEEE INFOCOM'09, Rio de Janeiro, Brazil, April 19-25, 2009.

[19] 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.

[20] E. Ayday, F. Delgosha, and F. Fekri, "Location-Aware Security Services for Wireless Sensor Networks Using Network

[21] Ms. Neha Khatri –Valmik, Prof. V K Ksirsagar, " Blowfish Algorithm", e-ISSN:2278-0661, p-ISSN:2278-8728, volume 16, pp 80-83, Mar-Apr 2014.

