On network coding for security

Keesook Han, AFRL
Tracey Ho, Computer Science, Caltech
Ralf Koetter, ICE, TUM
Muriel Médard, EECS, MIT
Fang Zhao, EECS, MIT

Abstract **The use of network coding in military networks opens many interesting issues for security. The mixing of data inherent to network coding may at first appear to pose challenges, but it also enables new security approaches. In this paper, we overview the recent current theoretical understanding and application areas for network-coding based security in the areas of robustness to Byzantine attackers and of distributed signature schemes for downloads.**

I. Introduction

The Global Information Grid (GIG) is the infrastructure used to conduct Net-Centric Operations (NCO). The GIG is intended to be a single information-sharing network with multiple levels of security and bandwidth capabilities in net-centric environment. A net-centric information environment is inclusive of Core and Communities of Interest (COI) enterprise services, a data sharing strategy, and the Task-Post-Process-Use (TPPU) paradigm. The Global Information Grid Bandwidth Expansion (GIG-BE) Program was a major DoD net-centric transformational initiative executed by Defense Information System Agency (DISA).

The ultimate purpose of the GIG-BE projects is to provide a secure and reliable platform to enable worldwide Net-Centric Operations for intelligence, surveillance and reconnaissance and command and control massive amounts of information sharing by providing ”bandwidth-available” environment. Through GIG-BE, DISA leveraged DOD's existing end-to-end information transport capabilities, significantly expanding capacity and reliability to select Joint Staff-approved locations worldwide and under new hardware and software contracts to build a communications infrastructure. The GIG-BE that is intended to provide high-capacity communications linking DoD users at locations worldwide is a ground-based optical network with up to 10-Gbps connections and averaging 105 Gbps per link on the backbone networks. The GIG-BE program has greatly contributed to the development of the real-time Net-Centric Operations. However, a bottleneck link problem exists between core networks and edge networks due to the enormous difference in bandwidths.

The DoD supports NCO and GIG-BE projects to improve quality of services in net-centric environment. The current coding systems will not be appropriate in the near future. However, coding based scalable communication technology has not been applied to the Net-Centric Operations. This technology will satisfy the bandwidth requirements of tomorrow's warfighters.

Network coding is a recent development in which nodes in the network are allowed to perform algebraic operations inside the network. This scheme was first introduced in [1] and a powerful algebraic framework, which allows further developments, was provided in [2], [3]. For multicast settings, it was shown in [4], [5] that network coding performed in a distributed, random fashion is with high probability optimal. A tutorial on network coding can be found in [6], [7].

The specifics of the Scalable Information Operations (SIO) include: 1. scalable coding techniques for network coding, compression, channel coding, multimedia data transmission, encryption, data sharing, data anonymization, meta database management, caching, network security, and intrusion detection. 2. Bottleneck flow control. The purpose of this paper is to overview some of the recent developments in applying network coding to security in the areas of detection and correction of Byzantine attacks, and of cryptography for network coding based file downloads. The aim of this paper is mainly tutorial and further technical details can be found in [8], [9]. Especially, our goal is to sketch how network-coding based scalable information operations will mitigate some of the security issues in the future net-centric environment

II. Network-coding based detection and correction of Byzantine attackers

The mixture of data that occurs in network coding can lead to pollution attacks through rogue, or Byzantine, nodes in the network [10], [11]. Such nodes may be unreliable through failure or because of their being compromised. While the use of network coding would at first appear to render the problem of Byzantine attackers worse, it actually provides some strong protection for both the detection and correction of such nodes.

The results in this section have previously appeared in more detailed form in [8]. We consider network error correction in a distributed packet network setting with random linear network coding using coding vectors. A batch of r packets is multicast from a source node s to a set of sink nodes. An omniscient adversary can arbitrarily corrupt the coding vector as well as the data symbols of up to z_o packets. A packet that is not a linear combination of its input packets is called *adversarial*.

We describe below a polynomial-complexity network error-correcting code whose parameters depend on z_o, the maximum number of adversarial packets, and m, the minimum source-sink cut capacity (maximum error-free multicast rate) in units of packets over the batch. The number of packets in the batch is set as $r = m - z_o$. The proportion of redundant symbols in each packet, denoted ρ, is set as $\rho = (z_o+\epsilon)/r$ for some $\epsilon > 0$.

The corresponding information rate of the code approaches $m-2z_o$ asymptotically as the packet size increases. If instead of an omniscient adversary we assume that the source and sinks share a secret channel not observed by the adversary, a higher rate of $m-z_o$ is asymptotically achievable. Below we give the details of the code for the omniscient adversary case.

For $i=1,\ldots,r$, the ith source packet is represented as a length-n row vector $\mathbf{x}_i$ with entries in a finite field $\mathbb{F}_q$. The first $n-\rho n-r$ entries of the vector are independent exogenous data symbols, the next ρn are redundant symbols, and the last r symbols form the packet's coding vector (the unit vector with a single nonzero entry in the ith position). We denote by $\mathbf{X}$ the $r\times n$ matrix whose ith row is $\mathbf{x}_i$; it can be written in the block form $\begin{bmatrix}\mathbf{U} & \mathbf{R} & \mathbf{I}\end{bmatrix}$ where $\mathbf{U}$ denotes the $r\times(n-\rho n-r)$ matrix of exogenous data symbols, $\mathbf{R}$ denotes the $r\times\rho n$ matrix of redundant symbols and $\mathbf{I}$ is the $r\times r$ identity matrix.

The $r\rho n$ redundant symbols are obtained as follows. For any matrix $\mathbf{M}$, let $\mathbf{v}_{\mathbf{M}}^T$ denote the column vector obtained by stacking the columns of $\mathbf{M}$ one above the other, and $\mathbf{v}_{\mathbf{M}}$ its transpose, a row vector. Matrix $\mathbf{X}$, represented in column vector form, is given by $\mathbf{v}_{\mathbf{X}}^T=[\mathbf{v}_{\mathbf{U}},\mathbf{v}_{\mathbf{R}},\mathbf{v}_{\mathbf{I}}]^T$. Let $\mathbf{D}$ be an $r\rho n\times rn$ matrix obtained by choosing each entry independently and uniformly at random from $\mathbb{F}_q$. The redundant symbols constituting $\mathbf{v}_{\mathbf{R}}$ (or $\mathbf{R}$) are obtained by solving the matrix equation

$$\mathbf{D}[\mathbf{v}_{\mathbf{U}},\mathbf{v}_{\mathbf{R}},\mathbf{v}_{\mathbf{I}}]^T=\mathbf{0} \tag{1}$$

for $\mathbf{v}_{\mathbf{R}}$. The value of $\mathbf{D}$ is known to all parties.

An adversarial packet can be viewed as an additional source packet. The vector representing the ith adversarial packet is denoted $\mathbf{z}_i$. Let $\mathbf{Z}$ denote the matrix whose ith row is $\mathbf{z}_i$.

We focus on any one of the sink nodes t. Let w be the number of linearly independent packets received by t; let $\mathbf{Y}\in\mathbb{F}_q^{w\times n}$ denote the matrix whose ith row is the vector representing the ith of these packets. Since all coding operations in the network are scalar linear operations in $\mathbb{F}_q$, $\mathbf{Y}$ can be be expressed as

$$\mathbf{Y}=\mathbf{GX}+\mathbf{KZ} \tag{2}$$

where matrices $\mathbf{G}\in\mathbb{F}_q^{w\times r}$ and $\mathbf{K}\in\mathbb{F}_q^{w\times z}$ represent the linear mappings from the source and adversarial packets respectively to the sink's set of linearly independent input packets.

Since the last r columns of $\mathbf{X}$ form an identity matrix, the matrix $\mathbf{G}'$ formed by the last r columns of $\mathbf{Y}$ is given by

$$\mathbf{G}'=\mathbf{G}+\mathbf{KL}, \tag{3}$$

where $\mathbf{L}$ is the matrix formed by the last r columns of $\mathbf{Z}$. The sink knows $\mathbf{G}'$ but not $\mathbf{G}$. Thus, we rewrite (2) as

$$\begin{aligned}\mathbf{Y} &= \mathbf{G}'\mathbf{X}+\mathbf{K}(\mathbf{Z}-\mathbf{LX})\\ &= \mathbf{G}'\mathbf{X}+\mathbf{E}\end{aligned} \tag{4}$$

Matrix $\mathbf{E}$ gives the difference between the data values in the received packets and the data values corresponding to their coding vectors; its last r columns are all zero.

Lemma 1: With probability at least $1-\eta/q$, the matrix $\mathbf{G}'$ has full column rank, where η is the number of links in the network.

The decoding process at sink t is as follows. First, the sink determines z, the minimum cut from the adversarial packets to the sink. This is with high probability equal to $w-r$. Next, it chooses z columns of $\mathbf{Y}$ that, together with the columns of $\mathbf{G}'$, form a basis for the column space of $\mathbf{Y}$. We assume without loss of generality that the first z columns are chosen, and we denote the corresponding submatrix $\mathbf{G}''$. Rewriting $\mathbf{Y}$ in the basis corresponding to the matrix $[\mathbf{G}''\ \mathbf{G}']$, we have

$$\mathbf{Y}=[\mathbf{G}''\ \mathbf{G}']\begin{bmatrix}\mathbf{I}_z & \mathbf{Y}^Z & 0\\ 0 & \mathbf{Y}^X & \mathbf{I}_r\end{bmatrix} \tag{5}$$

This can be reduced by linear algebraic manipulations to

$$\mathbf{G}'\mathbf{X}_2=\mathbf{G}'(\mathbf{Y}^X+\mathbf{X}_1\mathbf{Y}^Z) \tag{6}$$

where $\mathbf{X}_1,\mathbf{X}_2$ are the matrices formed by the first z columns of $\mathbf{X}$ and the next $n-z-r$ columns of $\mathbf{X}$ respectively.

Proposition 1: With probability greater than $1-q^{n\epsilon}$, equations (1) and (6) can be solved simultaneously to recover $\mathbf{X}$. The decoding algorithm has complexity $O(n^3m^3)$.

III. Cryptography for Content Distribution with Network Coding

A. Background

Recently, several researchers explored the use of network coding in peer-to-peer (P2P) content distribution and distributed storage systems [12], [13], [14]. A P2P network has a fully distributed architecture, and the peers in the network form a cooperative network that shares the resources, such as storage, CPU, and bandwidth, of all the computers in the network. This architecture offers a cost-effective and scalable way to distribute software updates, videos, and other large files to a large number of users.

The best example of a P2P cooperative architecture is the BitTorrent system [15], which splits large files into small blocks, and after a node downloads a block from the original server or from another peer, it becomes a server for that particular block. Although BitTorrent has become extremely popular for distribution of large files over the Internet, it may suffer from a number of inefficiencies which decrease its overall performance. For example, scheduling is a key problem in BitTorrent: it is difficult to efficiently select which block(s) to download first and from where. If a rare block is only found on peers with slow connections, this would create a bottleneck for all the downloaders. Several *ad hoc* strategies are used in BitTorrent to ensure that different blocks are equally spread in the system as the system evolves. References [12], [13] propose the use of network coding to increase the efficiency of content distribution in a P2P cooperative architecture. The main idea of this approach is the following. The server breaks the file to be distributed into small blocks, and whenever a peer requests a file, the server sends a random linear combination of all the blocks. As in BitTorrent, a peer acts as a server to the blocks it has obtained. However, in a linear coding scheme, any output from a peer node is also a random linear combination of all the blocks it has already received. A peer node can reconstruct the whole file when it has received enough degrees of freedom to decode all the blocks. This scheme is completely distributed, and eliminates the need for a scheduler, as any block transmitted contains partial information of all the blocks that the sender possesses. It has been shown both mathematically [12] and through live trials [16] that the random linear coding scheme significantly reduces the downloading time and improves the robustness of the system.

A major concern for any network coding system is the protection against malicious nodes. Take the above content distribution system for example. If a node in the P2P network behaves maliciously, it can create a polluted block with valid coding coefficients, and then sends it out. Here, coding coefficients refer to the random linear coefficients used to generate this block. If there is no mechanism for a peer to check the integrity of a received block, a receiver of this polluted block would not be able to decode anything for the file at all, even if all the other blocks it has received are valid. To make things worse, the receiver would mix this polluted block with other blocks and send them out to other peers, and the pollution can quickly propagate to the whole network. This makes coding based content distribution even more vulnerable than the traditional P2P networks, and several attempts were made to address this problem. References [12], [17] proposed to use homomorphic hash functions in content distribution systems to detect polluted packets, and [18] suggested the use of a Secure Random Checksum (SRC) which requires less computation than the homomorphic hash function. However, [18] requires a secure channel to transmit the SRCs to all the nodes in the network. Charles *et al* [19] proposed a signature scheme based on Weil pairing on elliptic curves and provides authentication of the data in addition to pollution detection, but the computation complexity of this solution is quite high. Moreover, the security offered by elliptic curves that admit Weil pairing is still a topic of debate in the scientific community.

In this section, we overview a new signature scheme, presented in greater detail in [9], that is not based on elliptic curves, and is designed specifically for random linear coded systems. We view all blocks of the file as vectors, and make use of the fact that all valid vectors transmitted in the network should belong to the subspace spanned by the original set of vectors from the file. We present a signature that can be used to easily check the membership of a received vector in the given subspace, and at the same time, it is hard for a node to generate a vector that is not in that subspace but passes the signature test. We show that this signature scheme is secure, and that the overhead for the scheme is negligible for large files.

B. *Problem Setup*

We model the network by a directed graph $G_d = (N, A)$, where N is the set of nodes, and A is the set of communication links. A source node $s \in N$ wishes to send a large file, of size M, to a set of client nodes, $T \subset N$, and we refer to all the clients as *peers*. The large file is divided into m blocks, and any peer receives different blocks from the source node or from other peers. In this framework, a peer is also a server to blocks it has downloaded, and always sends out random linear combinations of all the blocks it has obtained so far to other peers. When a peer has received enough degrees of freedom to decode the data, i.e., it has received m linearly independent blocks, it can re-construct the whole file.

Specifically, we view the m blocks of the file, $\bar{\mathbf{v}}_1, ..., \bar{\mathbf{v}}_m$, as elements in n-dimensional vector space $\mathbb{F}_p^n$, where p is a prime. The source node augments these vectors to create vectors $\mathbf{v}_1, ..., \mathbf{v}_m$, given by

$$\mathbf{v}_i = (0, ..., 1, ..., 0, \bar{v}_{i1}, ..., \bar{v}_{in}),$$

where the first m elements are zero except that the ith one is 1, and $\bar{v}_{ij} \in \mathbb{F}_p$ is the jth element in $\bar{\mathbf{v}}_i$. Packets received by the peers are linear combinations of the augmented vectors,

$$\mathbf{w} = \sum_{i=1}^{m} \beta_i \mathbf{v}_i,$$

where β_i is the weight of $\mathbf{v}_i$ in $\mathbf{w}$. We see that the additional m elements in the front of the augmented vector keep track of the code vector, β, of the corresponding packet.

As mentioned in the previous subsection, this kind of network coding scheme is vulnerable to pollution attacks by malicious nodes. Unlike uncoded systems where the source knows all the blocks being transmitted in the network, and therefore, can sign each one of them, in a coded system, each peer produces "new" packets, and standard digital signature schemes do not apply here. In the next subsection, we introduce a novel signature scheme for the coded system.

C. *Signature scheme for network coding*

We note that the vectors $\mathbf{v}_1, ..., \mathbf{v}_m$ span a subspace V of $\mathbb{F}_p^{m+n}$, and a received vector $\mathbf{w}$ is a valid linear combination of vectors $\mathbf{v}_1, ..., \mathbf{v}_m$ if and only if it belongs to the subspace V. This is the key observation for our signature scheme. In the scheme described below, we present a system that is based upon standard modulo arithmetic (in particular the hardness of the Discrete Logarithm problem) and upon an invariant signature $\sigma(V)$ for the linear span V. Each node verifies the integrity of a received vector $\mathbf{w}$ by checking the membership of $\mathbf{w}$ in V based on the signature $\sigma(V)$.

Our signature scheme is defined by the following ingredients, which are independent of the file(s) to be distributed:

- q: a large prime number such that p is a divisor of $q-1$. Note that standard techniques, such as that used in Digital Signature Algorithm (DSA), apply to find such q.
- g: a generator of the group G of order p in $\mathbb{F}_q$. Since the order of the multiplicative group $\mathbb{F}_q^*$ is $q-1$, which is a multiple of p, we can always find a subgroup, G, with order p in $\mathbb{F}_q^*$.
- Private key: $\mathbf{K}_{pr} = \{\alpha_i\}_{i=1,...,m+n}$, a random set of elements in $\mathbb{F}_p^*$. $\mathbf{K}_{pr}$ is only known to the source.

- Public key: $\mathbf{K}_{pu} = \{h_i = g^{\alpha_i}\}_{i=1,...,m+n}$. $\mathbf{K}_{pu}$ is signed by some standard signature scheme, e.g., DSA, and published by the source.

To distribute a file in a secure manner, the signature scheme works as follows.

1) Using the vectors $\mathbf{v}_1, ..., \mathbf{v}_m$ from the file, the source finds a vector $\mathbf{u} = (u_1, ..., u_{m+n}) \in \mathbb{F}_p^{m+n}$ orthogonal to all vectors in V. Specifically, the source finds a non-zero solution, $\mathbf{u}$, to the equations
$$\mathbf{v}_i \cdot \mathbf{u} = 0, \quad i = 1, ..., m.$$
2) The source computes vector $\mathbf{x} = (u_1/\alpha_1, u_2/\alpha_2, ..., u_{m+n}/\alpha_{m+n})$.
3) The source signs $\mathbf{x}$ with some standard signature scheme and publishes $\mathbf{x}$. We refer to the vector $\mathbf{x}$ as the signature, $\sigma(V)$, of the file being distributed.
4) The client node verifies that $\mathbf{x}$ is signed by the source.
5) When a node receives a vector $\mathbf{w}$ and wants to verify that $\mathbf{w}$ is in V, it computes
$$d = \prod_{i=1}^{m+n} h_i^{x_i w_i},$$
and verifies that $d = 1$.

To see that d is equal to 1 for any valid $\mathbf{w}$, we have
$$d = \prod_{i=1}^{m+n} h_i^{x_i w_i} = g^{\sum_{i=1}^{m+n}(u_i w_i)} = 1$$
where the last equality comes from the fact that $\mathbf{u}$ is orthogonal to all vectors in V.

Next, we show that the system described above is secure. In essence, the theorem below shows that given a set of vectors that satisfy the signature verification criterion, it is provably as hard as the Discrete Logarithm problem to find new vectors that also satisfy the verification criterion other than those that are in the linear span of the vectors already known.

Definition 1. Let p be a prime number and G be a multiplicative cyclic group of order p. Let k and n be two integers such that $k < n$, and $\mathbf{\Gamma} = \{h_1, ..., h_n\}$ be a set of generators of G. Given a linear subspace, V, of rank k in $\mathbb{F}_p^n$ such that for every $\mathbf{v} \in V$, the equality $\mathbf{\Gamma}^{\mathbf{v}} \triangleq \prod_{i=1}^{n} h_i^{v_i} = 1$ holds, we define the (p, k, n)-Diffie-Hellman problem as the problem of finding a vector $\mathbf{w} \in \mathbb{F}_p^n$ with $\mathbf{\Gamma}^{\mathbf{w}} = 1$ but $\mathbf{w} \notin V$.

By this definition, the problem of finding an invalid vector that satisfies our signature verification criterion is a $(p, m, m+n)$-Diffie-Hellman problem.

Theorem 1. For any $k < n - 1$, the (p, k, n)-Diffie-Hellman problem is as hard as the Discrete Logarithm problem.

Proof: Assume that we have an efficient algorithm to solve the (p, k, n)-Diffie-Hellman problem, and we wish to compute the discrete algorithm $\log_g(z)$ for some $z = g^x$, where g is a generator of a cyclic group G with order p. We can choose two random vectors $\mathbf{r} = (r_1, ..., r_n)$ and $\mathbf{s} = (s_1, ..., s_n)$ in $\mathbb{F}_p^n$, and construct $\mathbf{\Gamma} = \{h_1, ..., h_n\}$, where $h_i = z^{r_i} g^{s_i}$ for $i = 1, ..., n$. We then find k linearly independent (and otherwise random) solution vectors $\mathbf{v}_1, ..., \mathbf{v}_k$ to the equations
$$\mathbf{v} \cdot \mathbf{r} = 0 \text{ and } \mathbf{v} \cdot \mathbf{s} = 0.$$
Note that there exist $n-2$ linearly independent solutions to the above equations. Let V be the linear span of $\{\mathbf{v}_1, ..., \mathbf{v}_k\}$, it is clear that any vector $\mathbf{v} \in V$ satisfies $\mathbf{\Gamma}^{\mathbf{v}} = 1$. Now, if we have an algorithm for the (p, k, n)-Diffie-Hellman problem, we can find a vector $\mathbf{w} \notin V$ such that $\mathbf{\Gamma}^{\mathbf{w}} = 1$. This vector would satisfy $\mathbf{w} \cdot (x\mathbf{r} + \mathbf{s}) = 0$. Since $\mathbf{r}$ is statistically independent from $(x\mathbf{r} + \mathbf{s})$, with probability greater than $1 - 1/p$, we have $\mathbf{w} \cdot \mathbf{r} \neq 0$. In this case, we can compute
$$\log_g(z) = x = \frac{\mathbf{w} \cdot \mathbf{s}}{\mathbf{w} \cdot \mathbf{r}}.$$
This means the ability to solve the (p, k, n)-Diffie-Hellman problem implies the ability to solve the Discrete Logarithm problem. ■

This proof is an adaptation of a proof that appeared in an earlier publication by Boneh *et. al* [20].

D. Discussion

Our signature scheme nicely makes use of the linearity property of random linear network coding, and enables the peers to check the integrity of packets without the requirement for a secure channel. Also, the computation involved in the signature generation and verification processes is very simple.

Next, we examine the overhead incurred by this signature scheme. The size of each file block is $B = n \log(p)$ and we have $M = mn \log(p)$. The size of each augmented vector (with coding vectors in the front) is $B_a = (m + n) \log(p)$, and thus, the overhead of the coding vector is m/n times the file size. Note that this is the overhead pertaining to the linear coding scheme, not to our signature scheme, and any practical network coding system would make $m \ll n$. The initial setup of our signature scheme involves the publishing of the public key, $\mathbf{K}_{pu}$, which has size $(m + n) \log(q)$. In typical cryptographic applications, the size of p is 20 bytes (160 bits), and the size of q is 128 bytes (1024 bits), thus, the size of $\mathbf{K}_{pu}$ is approximately equal to $6(m + n)/mn$ times the file size.

For distribution of each file, the incremental overhead of our scheme consists of two parts: the public data, $\mathbf{K}_{pu}$, and the signature vector, $\mathbf{x}$.

For the public key, $\mathbf{K}_{pu}$, we note that it cannot be fully reused for multiple files, as it is possible for a malicious node to generate a invalid vector that satisfies the check $d = 1$ using information obtained from previously downloaded files. To prevent this from happening, we can publish a new public key for each file, and as mentioned above, the overhead is about $6(m + n)/mn$ times the file size, which is small as long as $6 \ll m \ll n$.

Alternatively, for every new file, we can randomly pick an integer i between 1 and $m + n$, select a new random value for α_i in the private key, and just publish the new $h_i = g^{\alpha_i}$. The overhead for this method is only $6/mn$ times the file size. As an example, if we have a file of size 10MB, divided into $m = 100$ blocks, the value of n would be in the order of thousands, and thus, this overhead is less than 0.01% of the file size. This method should provide good security except in the case where we expect the vector $\mathbf{w}$ to have low variability, for example, has many zeros. Security can be increased by changing more elements in the private key for each new file.

In addition, for each new file distributed, we also have to publish a new signature $\mathbf{x}$, which is computed from a vector $\mathbf{u}$ that is orthogonal to the subspace V spanned by the file. Since the V has dimension m, it is sufficient to only replace m elements in $\mathbf{u}$ to generate a vector orthogonal to the new file. Since the first m elements in the vectors $\mathbf{v}_1, ..., \mathbf{v}_m$ are always linearly independent (they are the code vectors), it suffices to just modify the entries u_1 to u_m. Assume that the ith element in the private key is the only one that has been changed for the distribution of the new file, and that i is between 1 and m, then we only need to publish x_1 to x_m for the new signature vector. This part of the overhead has size $m \log(p)$, and the ratio between this overhead and the original file size N is $1/n$. Again, take a 10MB file for example, this overhead is less than 0.1% of the file size.

Therefore, after the initial setup, each additional file distributed only incurs a negligible amount of overhead using our signature scheme.

Finally, we would like to point out that, under our assumptions that there is no secure side channel from the source to all the peers and that the public key is available to all the peers, our signature scheme has to be used on the original file vectors not on hash functions. This is because to maintain the security of the system, we need to use a one-way hash function that is homomorphic, however, we are not aware of any such hash function. Although [12] and [17] suggested usage of homomorphic hash functions for network coding, [12] assumed that the intermediate nodes do not know the parameters used for generating the hash function, and [17] assumed that a secure channel is available to transmit the hash values of all the blocks from the source node to the peers. Under our more relaxed assumptions, these hash functions would not work.

IV. Conclusions

In this paper, we have overviewed some of the security capabilities of network coding, particularly in the area of robustness to Byzantine attacks and to distributed authentication in peer-to-peer downloads. The implications of network coding for security are not limited to these applications. For instance, network coding's mixture of data can be used to use data for effective countermeasures to eavesdropping. In effect, data is used, after compression, as a one-time-pad in the system, [21], [22], [23], [24]. None of these techniques or the ones summarized in this paper present in themselves a complete security solution, and we have not attempted to implement any of our security techniques. However, as network coding opens entirely new venues for the operation of networks, we expect to see security challenges inherited from traditional forms of networking, the mitigation of current problems but also the emergence of new classes of data sharing problems in Net-Centric environment. We will further develop scalable and secure network coding techniques to solve multimedia delivery and massive data sharing problems in Airborne/UAV networks.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow: Single source," *submitted to IEEE Transactions on Information Theory*, 1999.

[2] R. Koetter and M. Médard, "An algebraic approach to network coding," in *IEEE International Symposium on Information Theory (ISIT)*, vol. 1, p. 104, 2001.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (selected as one of the outstanding papers from INFOCOM for transfer to IEEE/ACM Transaction on Networking)*, vol. 11, pp. 782–796, October.

[4] T. Ho, M. Médard, R. Koetter, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE International Symposium on Information Theory (ISIT)*, p. 442, 2003.

[5] T. Ho, M. Mdard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," in *IEEE Transactions on Information Theory*, vol. 52, pp. 4413–4430, October 2006.

[6] M. Effros, R. Koetter, and M. Médard, "Breaking network logjams," *Scientific American*, vol. 6, pp. 78–85, June.

[7] C. Fragouli, J.-Y. L. Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, 2006.

[8] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *Proc. IEEE INFOCOM 2007*, (Anchorage, AK), May 2007.

[9] F. Zhao, T. Kalker, M. Médard, and K. Han, "Signatures for content distribution with network coding," in *Proc. of IEEE ISIT'07*, (Nice, France), July 2007.

[10] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[11] R. Perlman, "Network layer protocols with byzantine robustness," *MIT Ph.D. Thesis*, 1988.

[12] S. Acedański, S. Deb, M. Médard, and R. Koetter, "How good is random linear coding based distributed network storage?," in *Proc. 1st Workshop on Network Coding, Theory, and Applications (Netcod'05)*, (Riva del Garda, Italy), Apr. 2005.

[13] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE INFOCOM'05*, (Miami, FL), Mar. 2005.

[14] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. of IEEE INFOCOM'07*, (Anchorage, Alaska), Mar. 2007.

[15] "Bittorrent file sharing protocol. http://www.bittorrent.com,"

[16] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive view of a live network coding p2p system," in *Proc. ACM SIGCOMM/USENIX Internet Measurement Conference (IMC'06)*, (Rio de Janeiro, Brazil), Oct. 2006.

[17] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *Proc. of IEEE INFOCOM'06*, (Barcelona, Spain), Apr. 2006.

[18] M. N. Krohn, M. J. Freedman, and D. Mazières, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proc. of the IEEE Symposium on Security and Privacy*, (Oakland, CA), May 2004.

[19] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *Proc. of Conference on Information Sciences and Systems (CISS'06)*, (Princeton, NJ), Mar. 2006.

[20] D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme," in *Proc. of Crypto'99, Lecture Notes in Computer Science*, 1999.

[21] N. Cai and R. W. Yeung, "Secure network coding," in *IEEE International Symposium on Information Theory*, July 2002.

[22] K. Bhattad and K. Narayanan, "Weakly secure network coding," in *Proc. of the First Workshop on Network Coding, Theory, and Applications (NetCod)*, 2005.

[23] J. Tan and M. Médard, "Secure network coding with a cost criterion," in *Proc. of the Second Workshop on Network Coding, Theory, and Applications (NetCod)*, 2006.

[24] L. Lima, M. Médard, and J. Barros, "Random network coding: A free cypher?," in *IEEE International Symposium on Information Theory (ISIT)*, 2007.