Signatures for Content Distribution with Network Coding

Fang Zhao*, Ton Kalker†, Muriel Médard*, and Keesook J. Han‡
* Lab for Information and Decision Systems
MIT, Cambridge, MA 02139, USA
E-mail: zhaof/medard@mit.edu
†Hewlett-Packard Laboratories
Palo Alto, Cambridge, CA, USA
E-mail: ton.kalker@hp.com
‡Air Force Research Laboratory
Information Directorate, Rome, NY, USA
E-mail: keesook.Han@rl.af.mil

***Abstract*— Recent research has shown that network coding can be used in content distribution systems to improve the speed of downloads and the robustness of the systems. However, such systems are very vulnerable to attacks by malicious nodes, and we need to have a signature scheme that allows nodes to check the validity of a packet without decoding. In this paper, we propose such a signature scheme for network coding. Our scheme makes use of the linearity property of the packets in a coded system, and allows nodes to check the integrity of the packets received easily. We show that the proposed scheme is secure, and its overhead is negligible for large files.**

I. Introduction

Network coding was first introduced in [1] as an alternative to the traditional routing networks, and it has been shown that random linear coding can be used to improve the throughput for multicast and even unicast transmissions [2], [3], [4]. More recently, several researchers explored the use of network coding in content distribution and distributed storage systems [5], [6]. Traditionally, the solutions for content distribution are based on a client-server model, where a central server sends the entire file to each client that requests it. This kind of approach becomes inefficient when the file size is large or when there are many clients, as it takes up a large amount of bandwidth and server resources. In recent years, peer-to-peer (P2P) networks have emerged as an alternative to traditional content distribution solutions to deliver large files. 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 [7], 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 [5], [6] 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 (see Fig. 1). 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 [5] and through live trials [8] 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

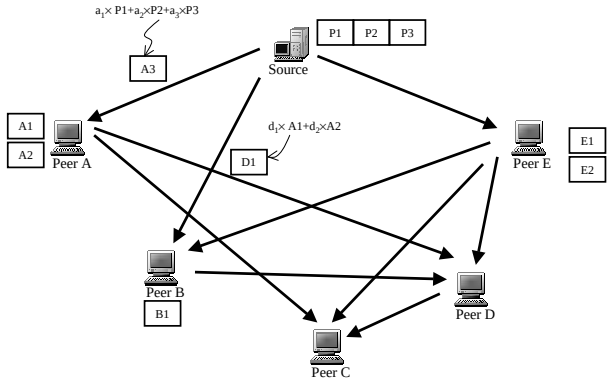

Fig. 1. Content distribution with network coding. Assume the file being distributed is broken into three blocks, $P1$, $P2$, and $P3$. Any packet being transmitted is a random linear combination of all the blocks the sender has. For example, the packet sent from the source to peer A is a combination of $P1$, $P2$, and $P3$, whereas the packet sent from peer A to D is a combination of blocks $A1$ and $A2$. A peer is able to decode the whole file when it receives 3 linearly independent blocks.

makes coding based content distribution even more vulnerable than the traditional P2P networks, such as BitTorrent. Similar security problems arise in all systems that use network coding, such as multicast networks. Several attempts were made to address this problem. Ho *et al* introduced Byzantine modification detection in multicast network with random network coding [9]. They added a simple polynomial hash value into each packet, and a receiver node can detect the presence of a Byzantine attacker with high probability, given that the attacker is unable to design and supply modified packets with complete knowledge of other packets received by other nodes. Jaggi *et al* [10] proposed a distributed network coding scheme for multicast network that is resilient in the presence of Byzantine adversaries. They view the adversarial nodes as a second source, and judiciously add redundancy at the real source to help the receivers distill out the source information from the received mixtures. References [5], [11] proposed to use homomorphic hash functions in content distribution systems to detect polluted packets, and [12] suggested the use of a Secure Random Checksum (SRC) which requires less computation than the homomorphic hash function. However, [12] requires a secure channel to transmit the SRCs to all the nodes in the network. Charles *et al* [13] proposed a signature scheme for network coding that does not require such a secure channel for transmitting hash values and associated digital signatures of received and transmitted blocks. This signature scheme is 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 paper, we propose a new signature scheme that is not based on elliptic curves, and is designed specifically for random linear coded systems. In this scheme, we view all blocks of the file as vectors, as in any network coding scheme, 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 design 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.

The rest of this paper is organized as follows. In Section II, we describe the setup of the problem, and introduce notations that will be used throughout this paper. We present the new signature scheme in Section III and prove that it is secure. Overheads and other aspects of the scheme are discussed in Section IV, and finally, the paper is concluded in Section V.

II. Problem Setup

In this section, we introduce the framework for a random linear coding based content distribution system. This framework can also be easily modified to be used for distributed storage systems. 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 to a set of client nodes, $T \subset N$. In this paper, 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 keeps track of the β values of the corresponding packet, i.e.,

$$\mathbf{w} = (\beta_1, ..., \beta_m, \bar{w}_{i1}, ..., \bar{w}_{in}),$$

where $(\bar{w}_{i1}, ..., \bar{w}_{in})$ is the payload part of the packet, and $(\beta_1, ..., \beta_m)$ is the code vector that is used to decode the packets.

As mentioned in the previous section, this kind of network coding scheme is vulnerable to pollution attacks by malicious

nodes [14], [15], and the pollution can quickly spread to other parts of the network if the peer just unwittingly mixes this polluted packet into its outgoing packets. 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 section, we introduce a novel signature scheme for the coded system.

III. 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

$$\begin{aligned} d &= \prod_{i=1}^{m+n} h_i^{x_i w_i} \\ &= \prod_{i=1}^{m+n} (g^{\alpha_i})^{u_i w_i / \alpha_i} \\ &= \prod_{i=1}^{m+n} g^{u_i w_i} \\ &= g^{\sum_{i=1}^{m+n} (u_i w_i)} \\ &= 1, \end{aligned}$$

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. Note that in general, the $(p, n-1, n)$-Diffie-Hellman problem has no solution. This is because if V has rank $n-1$ and a $\mathbf{w}'$ exists such that $\mathbf{\Gamma}^{\mathbf{w}'} = 1$ and $\mathbf{w}' \notin V$, then $w' + V$ spans the whole space, and any vector $\mathbf{w} \in \mathbb{F}_p^n$ would satisfy $\mathbf{\Gamma}^{\mathbf{w}} = 1$. This is clearly not true, therefore, no such $\mathbf{w}'$ exists.

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* [16].

IV. 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, as in the case of hash function or SRC schemes [5], [11], [12]. Also, the computation involved in the signature generation and verification processes is very simple.

Next, we examine the overhead incurred by this signature scheme. Let the file size be M and let the file be divided into m blocks, each one of which is a vector in $\mathbb{F}_p^n$. The size of each 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. Specifically, let $\mathbf{x}_1$ be the signature of File 1, and $\mathbf{w}_1$ be a valid received vector for File 1, we have

$$d = \prod_{i=1}^{m+n} h_i^{x_{1i}w_{1i}} = 1.$$

If the source then distribute File 2 using the same public key, $\mathbf{K}_{pu}$, and a different signature, $\mathbf{x}_2$, a malicious node can construct a vector $\mathbf{w}_2$, where $w_{2i} = x_{1i}w_{1i}/x_{2i}$, which satisfies the signature check

$$d = \prod_{i=1}^{m+n} h_i^{x_{2i}w_{2i}} = \prod_{i=1}^{m+n} h_i^{x_{1i}w_{1i}} = 1.$$

However, $\mathbf{w}_2$ is not a valid linear combination of the vectors of File 2. To prevent this from happening, we can publish a 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 $6 \ll m \ll n$. Note that if we republish $\mathbf{K}_{pu}$ for every new file, we can reuse the signature vector $\mathbf{x}$. Let $\mathbf{u}_2$ be a vector that is orthogonal to all vectors in File 2, the source can compute a new private key, $\mathbf{K}_{pr} = \{\alpha_1, ..., \alpha_{m+n}\}$, given by

$$\alpha_i = u_{2i}/x_i, \quad i = 1, ..., m+n.$$

The source then publishes the new public key, $\mathbf{K}_{pu} = \{h_i = g^{\alpha_i}\}_{i=1,...,m+n}$. In this way, we do not need to publish new $\mathbf{x}$ vectors for the subsequent files.

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 publish the new $h_i = g^{\alpha_i}$. The overhead for this method is $(m+n)$ times smaller than that described in the previous paragraph, i.e., this overhead 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.

However, if we only change one element in the public key, 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 [5] and [11] suggested usage of homomorphic hash functions for network coding, [5] assumed that the intermediate nodes do not know the parameters used for generating the hash function, and [11] 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.

V. Conclusions

Security problem is a main obstacle in the implementation of content distribution networks using random linear network coding. To tackle this problem, instead of trying to fit an existing signature scheme to network coding based systems, in this paper, we proposed a new signature scheme that is made specifically for such systems. We introduced a signature vector for each file distributed, and the signature can be used to easily check the integrity of all the packets received for this file. We have shown that the proposed scheme is as hard as the Discrete Logarithm problem, and the overhead of this scheme is negligible for a large file.

Acknowledgment

This work was supported by the National Science Foundation under grants "ITR: Network Coding - From Theory to Practice" (CCR-0325496), "XORs in the Air: Practical Wireless Network Coding" (CNS-0627021), and by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-06-1-0155.

References

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

[2] T. Ho, M. Médard, M. Effros, and D. Karger, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE Symposium on Information Theory (ISIT'03)*, Kanagawa, Japan, July 2003.

[3] Z. Li and B. Li, "Network coding: the case of multiple unicast sessions," in *Proc. 42th Annual Allerton Conference on Communication, Control, and Computing*, September-October, 2004.

[4] D. S. Lun, M. Médard, and R. Koetter, "Network coding for efficient wireless unicast," in *Proc. 2006 International Zurich Seminar on Communications (IZS'06)*, Zurich, Switzerland, February 2006.

[5] 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, April 2005.

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

[7] BitTorrent file sharing protocol. http://www.BitTorrent.com.

[8] 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, October 2006.

[9] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Proc. International Symposium on Information Theory (ISIT'04)*, Chicago, IL, June-July 2004.

[10] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard "Resilient network coding in the presence of Byzantine adversaries," accepted to *IEEE INFOCOM'07*, Anchorage, Alaska, May 2007.

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

[12] M. N. Krohn, M. J. Freedman, and D. Maziéres, "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.

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

[14] N. Christin, A. Weigend, and J. Chuang, "Content availability, pollution and poisoning in peer-to-peer file sharing networks," *ACM E-Commerce Conference (EC'05)*, June 2005.

[15] Xiaosong Lou and Kai Hwang, "Prevention of index-poisoning DDoS attacks in peer-to-peer file-sharing networks," *IEEE Trans. on Multimedia, Special Issue on Content Storage and Delivery in P2P Networks*, November, 2006.

[16] D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme," in *Proc. of Crypto'99, Lecture Notes in Computer Science*, vol. 1666, Springer-Verlag, pp. 338–353, 1999.