Secure Network Coding for a P2P System

Hun Jeong Kang[1], Aaram Yun[1], Eugene Y. Vasserman[1], Hyung Tae Lee[2], Jung Hee Cheon[2], Yongdae Kim[1]
[1]University of Minnesota [2] Seoul National University

1. INTRODUCTION

Network coding is a data transmission technique which allows intermediate nodes in a network to re-code data in transit. In contrast to traditional network communication where a node repeats incoming data to its outgoing channel without modifying the payload, a node implementing network coding not only repeats but also alters data. Network coding has been demonstrated to increase network throughput compared to the traditional forwarding transmission. It has potentially broad applications in many areas, including traditional computer networks, wireless ad-hoc networks, and peer-to-peer systems.

Nevertheless, the usefulness of network coding in peer-to-peer (P2P) systems is still in dispute [6, 10, 4]. Previous results are based on simulations and theoretical analysis, and thus may not reflect real network conditions. (Although a real-world implementation is given in [5], there is no real-world performance comparison between a network coding enabled system and a system not using network coding.) Neither has prior work considered performance overhead due to security, even though network coding has a critical security vulnerability — the pollution attack.

Since data is re-coded in transit in a network coding-enabled network, all data blocks are combinations of original blocks of a file, and none of them is immediately verifiable as having correct content by using traditional signatures and hashes. Moreover, any node doing re-coding can introduce introduce invalid data, corrupting *every instance of the file currently being downloaded* that incorporates the corrupted block. Although the final decoded file can be identified as corrupted with a traditional hash, the amount of bandwidth, storage, and computation time wasted on the invalid file cannot be recovered. Several schemes have been proposed to solve this problem [8, 3, 11, 7], but they are generally rather expensive or not applicable to P2P systems.

In this paper, we seek to answer to the following question: can *secure network coding in real-world environments* provide better performance than a protocol without network coding? We first propose an efficient homomorphic signature scheme which allows for real-time verification of encoded data. We implement our scheme, modify a BitTorrent [1] client to use our secure network coding libraries, and measure its performance in a wide area network. Our measurements in PlanetLab show secure network coding is feasible in P2P systems; in general network secure coding archives better performance than BitTorrent, and exhibits much shorter download time in some special cases.

2. BACKGROUND

2.1 System Model and Network coding

Consider a generic model of a P2P file-sharing service in which all nodes participate in both downloading and uploading of content, and no centralized entity is required. Files are divided into blocks of arbitrary size and peers *barter* for blocks, exchanging blocks they have with other nodes to obtain missing blocks. As soon as a peer acquires at least one block, she can begin sharing that block with others. However, this system may encounter the *last block problem* where some blocks become rare and downloads finish slower since many nodes wait to download those rare blocks. Nodes with many blocks are also not likely find nodes who have new blocks to offer, further delaying download completion (we call this the *unfair barter problem*). Network coding can be a good solution for these problems.

In linear network coding [9], a file is divided into m blocks, each represented as n elements in a finite field $\mathbb{F}_p$ of size p. Then the i-th block can be considered as a vector $\tilde{\mathbf{u}}_i = (u_{i,1}, \ldots, u_{i,n}) \in \mathbb{F}_p^n$ and the file becomes a sequence of vectors $(\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_m)$. When the original file source performs encoding, it picks random coefficients $(\alpha_1, \ldots, \alpha_m)$ (referred to as *encoding vector*) and computes an *information vector*, a linear combination of $\tilde{\mathbf{u}}_1$, $\ldots$, $\tilde{\mathbf{u}}_m$,

$$\tilde{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i \tilde{\mathbf{u}}_i = (w_1, \ldots, w_n). \quad (1)$$

The source then sends the encoding and information vectors together in the *chunk* with the augmented form of $\mathbf{w} = (\alpha_1, \ldots, \alpha_m, w_1, \ldots, w_n)$. A node responding to a request for an augmented block sends a linear combination $\sum_j \beta_j \mathbf{w}_j$ of its received chunks $\mathbf{w}_1$, $\ldots$, $\mathbf{w}_k$.

A receiver can decode the original file after receiving at least m linearly independent augmented blocks. Let W be the information vectors of received augmented blocks and A be the matrix whose rows are the encoding vectors of received augmented blocks. The receiver can recover of all original blocks of file U by solving the linear equation, $W = AU$.

3. SECURE NETWORK CODING

We present a new homomorphic signature scheme which can be used to verify data transmitted via network coding.

3.1 Our Homomorphic Signature Scheme

Let p be the prime number where $\mathbb{F}_p$ is the base field of the network coding, and G be a group of order p where the discrete logarithm problem is hard. Select g as a generator of G. Specifically, we choose two primes p, q with $p \mid q-1$. Then G is defined as the subgroup of order p of $\mathbb{Z}_q^*$. If $|q| \geq 1024$ and $|p| \geq 160$, then the currently conjectured difficulty of the discrete logarithm problem on G is about 2^{80}.

Let $s_1, \ldots, s_{m+n} \in \mathbb{Z}_p$ be randomly chosen secret exponents, and let $y_i \stackrel{\text{def}}{=} g^{s_i}$. Then the public key is defined as $PK = (p, q, g, y_1, \ldots, y_{m+n})$, and the secret key is $SK = (s_1, \ldots, s_{m+n})$.

For a chunk $\mathbf{w} = (w_1, \ldots, w_m, w_{m+1}, \ldots, w_{m+n})$, the signature σ for $\mathbf{w}$ is defined as

$$\sigma \leftarrow s_1 w_1 + \cdots + s_{m+n} w_{m+n} \pmod{p}.$$

Signature verification is done by checking

$$g^{\sigma} \stackrel{?}{=} y_1^{w_1} \cdots y_{m+n}^{w_{m+n}} \pmod{q}$$

This signature is homomorphic: if $\mathbf{w}_1, \ldots, \mathbf{w}_k$ are input chunks, and $\sigma_1, \ldots, \sigma_k$ are corresponding valid signatures received along with $\mathbf{w}_i$, then for any output chunk $\mathbf{w} = \alpha_1 \mathbf{w}_1 + \cdots + \alpha_k \mathbf{w}_k$, the corresponding signature is $\sigma \leftarrow \alpha_1 \sigma_1 + \cdots \alpha_k \sigma_k$.

3.2 Comparison of our scheme with others

Table 1 provides a comparison between our scheme and those of Krohn et al. [8], Zhao et al. [11], and Boneh et al [3]. Our scheme presents excellent characteristics, especially in the initial, signature, and aggregation costs. In case of the homomorphic hash function of Krohn et al., the source has to compute mn exponentiations to generate the full hash list to be initially distributed to the receivers. In comparison, our scheme has initial cost of only $m+n$ exponentiations, about the same as verifying a single chunk, and only about $1/m$ of that of Krohn et al.

In comparison with the homomorphic signature of Boneh et al., our scheme has negligible signature generation and aggregation costs, therefore the source's computation amounts to approximately verification of a single chunk. On the other hand, in the scheme of Boneh et al., the signature generation cost is comparable to the verification cost. Also, our scheme does not use the costly bilinear pairing operation. Our total cryptographic cost is actually very similar to the scheme of Zhao et al. – the total cost for signature generation is essentially the same. However, since ours is a homomorphic signature scheme, the cost is divided into chunks, and also the initially-distributed data is much more compact.

Only signature schemes of Boneh et al. are explicitly designed to be able to re-use key pairs for multiple files. This is a desirable property, but in other schemes, including ours, considering appropriate parameter settings for application to P2P systems, the size of the initial data (including the public key) is typically less than 1% of the file size. For many applications where the size of the public key is negligibly small compared to the actual file, we may tolerate the inability to re-use key pairs. Also, if the application requires anonymity or repudiation, public-key re-use may not be a desirable property.

Table 2: Signature verification time

size (bytes)	1K	4K	16K	64K	256K	1M	4M
time (sec)	0.031	0.070	0.226	0.863	3.35	13.5	53.4

4. EVALUATION

To show the feasibility of secure network coding, we implemented our scheme as a dynamic library and modified the BitTorrent Mainline 5.2.2 client [1] to take advantage of it. In this section, we show the computational overhead of our scheme and compare the performance of BitTorrent and network coding-enabled BitTorrent (*ncTorrent*) in a wide area network.

We measured the performance of our homomorphic signature scheme in terms of time to sign and verify a block. Signature generation time is almost negligible: it takes only a few milliseconds to sign a block several megabytes in size. This is because signature generation only consists of multiplication and additions, not requiring expensive exponentiations. Signature verification is more costly, and depends on the block size and the CPU configuration. Table 2 shows the signature verification times for various block sizes. The experiment was performed on a machine with an Intel Core2 Duo E8400 3.0 GHz CPU with a 6 MB L2 cache. The size of each element was 512 bits. It took 13.5 seconds to verify the signature of a 1 MB block (i.e., a block having 8,192 512-bit elements). We found that, in general, signature verification for one chunk is strictly slower than downloading speed, hence performance is degraded if we verify each chunk as we receive it. To reduce verification cost, we use a batch verification strategy where groups of received blocks are verified together.

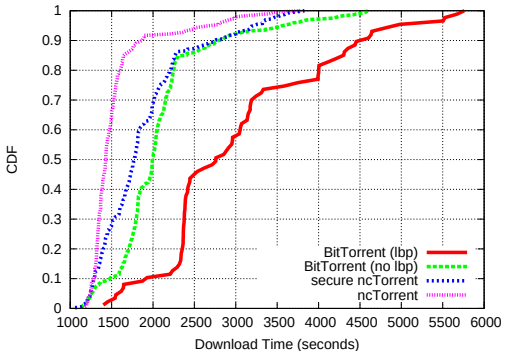

Figure 1: Download finish time.

We then compared the performance of BitTorrent and *ncTorrent*. Due to space constraints, we provide

Table 1: Comparison between some homomorphic authentication schemes

	Krohn et al.	Zhao et al.	Boneh et al.	Our design
Type	hash	sign.	sign.	sign.
Group	$G \subseteq \mathbb{Z}_q^*$, $\lvert G\rvert = p$	$G \subseteq \mathbb{Z}_q^*$, $\lvert G\rvert = p$	bilinear group	$G \subseteq \mathbb{Z}_q^*$, $\lvert G\rvert = p$
Public Key size[a]	constant	$m+n$	constant	$m+n$
Signature size[a]	m	$m+n$	m (1 per block)	m (1 per block)
Initial cost[b]	mn exp.	$m+n$ exp. and mn mult.	N/A	$m+n$ exp.
Signature cost	N/A	N/A	$m+n$ exp.	$m+n$ mult.
Aggregegation cost	N/A	N/A	l exp.[c]	l mult.[c]
Verification cost	$m+n$ exp.	$m+n$ exp.	$m+n$ exp.	$m+n+1$ exp.
Key pair reuse	no	no	yes	no

[a] Public key size and the signature size are represented by number of group elements.
[b] The cost to produce the data to be disseminated *before* the actual network coding based propagation can begin, e.g., the public key (in case key pair cannot be reused), the signature (in case of a non-homomorphic signature scheme).
[c] l is the number of chunks to be combined.

the measurement data of a simple case: all nodes start downloading simultaneously and leave network as soon as they finish downloading the file. (The more measurement results are provided in [2]. We used 100 PlanetLab nodes, each with 80KB/sec upload bandwidth limit and unlimited download bandwidth. Figure 1 shows the time to download a 128MB file with 256 blocks. Although *ncTorrent* equipped with our signature scheme (secure *ncTorrent*) is slower than *ncTorrent* without the scheme, we can see that even secure *ncTorrent* completes downloading faster than BitTorrent. When the last block problem occurred (BitTorrent-lbp line), *ncTorrent* exhibits much better performance. This improvement is due to the immunity of network coding systems to the last block problem and the *unfair barter problem*. These advantages of network coding can compensate for the overhead due to coding/decoding and signature verification. Although the performance improvement varies according to the situation, we have observed that in general network coding archives better performance than BitTorrent.

5. REFERENCES

[1] BitTorrent. http://www.bittorrent.com/.

[2] ncTorrent: Practical and secure network coding for P2P file sharing. Tech. rep., University of Minnesota, 2009.

[3] Boneh, D., Freeman, D., Katz, J., and Waters, B. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography* (2009), pp. 68–87.

[4] Chiu, D. M., Yeung, R. W., Huang, J., and Fan, B. Can network coding help in P2P networks? In *WiOpt* (2006), pp. 1–5.

[5] Gkantsidis, C., Miller, J., and Rodriguez, P. Comprehensive view of a live network coding P2P system. In *Internet Measurement Conference* (2006), pp. 177–188.

[6] Gkantsidis, C., and Rodriguez, P. R. Network coding for large scale content distribution. In *INFOCOM* (2005), pp. 2235–2245.

[7] Gkantsidis, C., and Rodriguez, P. R. Cooperative security for network coding file distribution. In *INFOCOM* (2006), pp. 1–13.

[8] Krohn, M. N., Freedman, M. J., and Mazières, D. On-the-fly verification of rateless erasure codes for efficient content distribution. In *IEEE Symposium on Security and Privacy* (2004), pp. 226–240.

[9] Li, S.-Y. R., Yeung, R. W., and Cai, N. Linear network coding. *IEEE Transactions on Information Theory 49*, 2 (2003), 371–381.

[10] Wang, M., and Li, B. How practical is network coding? In *IWQoS* (2006), pp. 274–278.

[11] Zhao, F., Kalker, T., Médard, M., and Han, K. J. Signatures for content distribution with network coding. In *ISIT* (2007), pp. 556–560.