ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, No 5, May 2013

Cooperative Defense against Pollution Attack in P2P System with Network Coding

Dr Siddaraju, Chitresha Mehta

***Abstract*— Network coding in Peer-to-Peer content distribution system provides a promising alternative to the traditional store-and-forward transmission paradigm but they suffer from pollution attacks which have detrimental effect on them. Existing systems verify blocks with traditional cryptographic signatures and hashes. However, these techniques do not apply well to more elegant schemes that use network coding techniques for efficient content distribution. In this work, we propose a novel, complete defense system for network coding-based P2P systems that can (i) detect polluted blocks early at intermediate peers (ii) identify the exact location of all colluding peers thus making it possible to eliminate them. (iii) Reduce the cost of verification thus preventing the propagation of malicious blocks. Here users not only cooperate to distribute the content, but (well-behaved) users also cooperate to protect themselves against malicious users by informing affected nodes when a malicious block is found. Our mechanisms introduce significantly less communication and computation overhead than other comparable state-of the- art schemes for P2P systems.**

***Index Terms*— Detection, Hash Generation, Identification, MAC, Network coding, Peer-to-Peer, Pollution Attack.**

I. INTRODUCTION

Network coding which has theoretically proven to maximize network throughput [1] ,enhance network robustness[2] and lower energy consumption [3] for communication networks allows algebraic mixing of information in the intermediate node. It allows participating nodes in a network to code incoming data flows rather than simply forwarding them. Its ability to achieve the maximum multicast flow rates in directed networks was first shown by Alswede et al. [1].

It has received extensive attentions and been applied to various computer network systems, such as multicast networks [4], wireless networks [5] and P2P systems [6].

P2P Systems have great advantages as compared to traditional content distribution solution to deliver large files using random network coding. It increases the throughput, decreases packet loss and delay, and gives robustness to dynamic network topologies. P2P cooperative network provides a cost effective distribution of software updates, video and other large files of thousand of simultaneous users in internet wide or private networks.

Despite their popularity, existing end-system suffers from number of problems which decreases their overall performance [7], [8], [9]. P2P systems implementing network coding are known to be vulnerable to pollution attacks. During such attacks malicious peers can upload fake data blocks into its downstream peers. Due to this the output of downstream peer will be polluted and fake.

The result is an epidemic propagation of corrupted blocks, as further downstream peer code and forward more corrupted data blocks which in turn results in wastage of network resources and prevents the peer from decoding the original block correctly.

In this paper we are proposing a cooperative system model. Our system can detect attacks quickly and can precisely pinpoint the malicious peers so that we can eliminate them from the network. In our systems peers not only distribute content, but well-behaved peers also cooperate to protect themselves against malicious peers by alerting peers when a malicious block is found. Our system is all well build to increase throughput even in the presence of some malicious nodes until they are detected and it also decreases cryptographic overhead.

The remainder of the paper is organized as follows .In Section II, we present related work on error correction and detection schemes. In Section III, we present our system model and adversary model along with all the assumptions. In Section IV, we describe our hash generation function. In Section V, we presented out cooperative scheme. In Section VI, we presented our detection scheme. In section VII, we present our identification scheme. Finally, we conclude the paper in Section VIII.

II. RELATED WORK

Due to severity of pollution attacks, large number of defense mechanism has been proposed by various research communities [10]. Homomorphic MAC schemes have been first proposed by Agrawal and Boneh [10]. This approach relies on cover-free set systems for pre-distributing keys to provide in-network detection, and thus, only c-collusion resistant. It is also susceptible to tag pollution attack. Li et al. [23] present a defense mechanism RIPPLE ,also built on a

***Dr Siddaraju**, HOD and Professor of CSE Department, VTU/ Dr Ambedkar Institute of Technology Bangalore, India, Phone/* 9449619956)

***Chitresha Mehta,** M.Tech Student of CSE Department, VTU/ Dr Ambedkar Institute of Technology /Bangalore, India, 7353926126.,*

homomorphic MAC scheme but utilizes time asymmetry (inspired by TESLA [16]) to achieve collusion and tag pollution resistance. This approach works only if it knows the longest path from each node to the source in order to precompute the MAC tags for the nodes. Both schemes have low computation overhead for both combining MAC tags and tag verification, since they only require simple addition and multiplication operations at intermediate nodes

In [17], Kehdi and Li proposed an in-network detection scheme which exploits subspace properties of network coding. The verification in this scheme is very efficient; however, this scheme is not collusion resistant: multiple nodes can collude to infer the null keys and make benign nodes accept corrupted vectors.

Existing mechanisms implementing homomorphic hashes [11] or homomorphic signatures [12] – [14] are computationally expensive. The mechanisms that implements message authentication codes (MACs) are also suffering from some problems like large number of colluding peers, works on fixed acyclic graph networks.

We are combining the use of homomorphic MAC and time asymmetry with pre-distribution of MAC tags to support general P2P network topologies.

Jafarisiavoshani, [21] identified the location of attackers and modified the subspace properties of a random network. In a network with multiple attackers, uncertainty of locating the attacker is with the attacker node, their parents and children. Even with multiple colluding attackers, [15] can pinpoint the exact location of attacker. SpaceMac provides exact identification of attackers and supports expanding subspaces. Directed acyclic graph network is not required for P2P systems. The non-repudiation protocol in [20] increases communication overhead by flooding multiple checksums for all the blocks sent by the source. TESLA avoids checksum dissemination and provides non-repudiation property.

III. MODELS AND ASSUMPTIONS

A. *Network Coding Model*

We consider an overlay P2P network where linear network coding is employed. Every peer can download from any other peer. Let there is one file F which exists at source S and every peer is interested in this file. For distributing this file, source S divides it into n blocks (k_1, k_2,…. k_n). Then each block k_i is subdivided into m codewords $k_{t,i}$, t Є1,..,m. F is considered as an m × n matrix of elements of Z_q (m is predetermined number of code words).

$$f = (k_1, k_2, \ldots . k_n) = \begin{pmatrix} k_{1,1} & \cdots & k_{1,n} \\ \vdots & \ddots & \vdots \\ k_{m,1} & \cdots & k_{m,n} \end{pmatrix}$$

With network coding, both source and peers perform encoding operation. At the beginning, all the peers are empty and peer P1 contact source to get a block. The source will pick some random coefficients rc_1, rc_2,….rc_n, then multiply each codewors of blocks k_i with rc_i, and add the result of the multiplications together to generate encoded block eb_1. Addition of blocks is defined as component-wise addition of the corresponding block codewords. That is, to combine the blocks of the file with coefficient vector, the source computes:

$$\vec{rc} = (rc_i), \sum_{i=1}^{i=n} rc_i\, k_i$$

The source will then transmit to peer P1 the result of the linear combination and the coefficient vector $\vec{rc}$ .Assume now that peer P1 has received another block of encoded information eb_2, either directly from the source or from another peer, with its associated vector of coefficients. If peer P1 needs to transmit an encoded block eb_3 to peer P2, P1 generates a linear combination of its two blocks eb_1 and eb_2 as follows. Peer P1 picks two random coefficients $rc_1^{''}$ and $rc_2^{''}$, multiplies each element of block eb_1 with the coefficient $rc_1^{''}$ and similarly for the second block eb_2, and adds the results of the multiplication. The block transmitted to peer P2 will be the addition of the multiplications $rc_1^{''}.eb_1$ and $rc_2^{''}.eb_2$. The coefficient vector $\overrightarrow{rc^{''}}$ associated with the new block is equal to $rc_1^{''}.\vec{rc} + rc_2^{''}.\overrightarrow{rc'}$.

From [11], peer can recover the original file after receiving n blocks for which the associated coefficient vectors are linearly independent to each other. The reconstruction process is same as solving a system of linear equations.

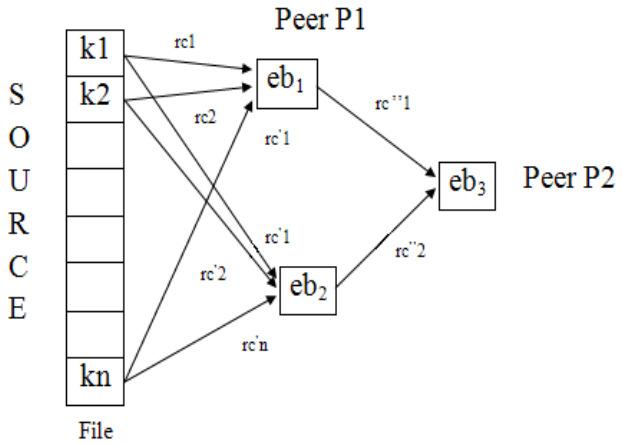

Fig. 1 Description of our Network Coding System

B. *Adversary Model*

We assume that source S is trusted and the adversaries may compromise an arbitrary subset of peers and inject corrupted blocks from any of these peers, at any time, so that legitimate users cannot decode the original file. The source encodes the file and distributes such encodings to multiple peers simultaneously. Peers download blocks from the source or other sub-peers and distribute new encoded blocks that are produced as linear combinations of all of the encoded blocks they hold. Thus, new encoded blocks are produced at each

peer even if the download has not been completed. The neighboring (subsets of pees) relationship is symmetric. Malicious peers are a fraction of the total peer population and can fully coordinate their activities.

We aim at providing a best effort security system which provides high throughput and can also identify and remove malicious peers. We do not require that peers build trust relationships. However, we do assume that legitimate users will disconnect from peers that frequently supply them with corrupted content.

C. Cryptographic Assumptions

We assume that each peer P shares with source S a pair of keys(PK_1^p, PK_2^p). We consider that there is a key distribution center (KDC) in our model which is a common tool for key distribution. If KDC is not available then shared keys can be established with the help of a public key infrastructure (PKI). Other approaches also assume the existence of a PKI [18], [23] or the existence of shared secret keys [10]. In general, the problem of establishing shared secret keys is a challenging problem of its own and is orthogonal to this work.

IV. HASH GENERATION

In [6], authors discussed how we can use homomorphic hash functions to validate encoded blocks. In the case of network coding, we need to verify random linear combination of blocks, which can be thought as weighted additions of all or of a subset of the blocks.

A hash function h(.), maps a large input, which is a block of information, say b, to an output h(b) typically of much smaller size. We will be using scalars, vectors and matrices defined over modular subgroups of Z. All nodes in the network must agree on same hash parameters so that any two nodes independently hashing the same file F should arrive at exactly the same hash. All additions are assumed to be taken over Z_q, and multiplications and exponentiations are assumed to be taken over Z_p. A trusted party (source) globally generates a set of hash parameters G = (p, q, g), where p and q are two large random primes such that $|p| = \lambda_p$, $|q| = \lambda_q$ and $q|(p-1)$. The parameter g is a vector of m (number of "sub-blocks" per block). All nodes are required to agree on same value of λ_q and λ_p. The hash of the file F is simply the vector of the hash of each k_i:

$$H(F) = (h(k_1), h(k_2), \ldots, h(k_n))$$

Where k_i can be defined as

$$h(k_i) = \prod_{t=1}^{m} g_t^{k_{t,i}} \bmod p$$

Whenever a peer joins the system, it downloads from the source the hash of the file H(F) as well as the security parameters G. In [4] it is presented that how hash of an encoded block can be constructed by the hashes of the original blocks. Hence, it is possible to check whether a received encoded block and coefficient vector are indeed correct.

V. COOPERATIVE SECURITY

In previous section we have shown how homomorphic function can be used with network coding to verify blocks as they are received at each peer. It is also been proved by number of authors [14], [4], [5]. But such homomorphic hashing functions are computationally expensive. So to improve computational time and reduce cryptographic work, cooperative security measure is implemented in which blocks are verified in batches, which is possible due to the homomorphic property of the hashing functions. This solution was first proposed in [19] and then used in [11] to improve the performance of homomorphic hashes for erasure codes. Suppose we have a window of W encoded blocks to verify all together. Let W represents block, coefficient relation as $((b_1, c_1), \ldots . (b_w, c_w))$, then resulting batched block $b_r = \sum_{i=1}^{W} b_i$, with combined coefficient vector$c_r = \sum_{i=1}^{L} c_i$. The advantage of batching is that peers only need to check one b_r block rather than individual blocks. Batching can decreases the computation time almost linearly with the size of window but it increases the risk of letting malicious packets propagate since packets are exchanges without being checked. The solution to this problem is presented in [11] where unverified blocks are isolated from window. One block is verified in the batch in involved in the coding process. The benefit of this solution is that all the packets are checked before they are forwarded to other peers. This in turn limits the possible scope of infection. To keep efficiency of system high, batching window need to be added to network coding process as soon as they are received.

Cryptographic work at each node can be reduced by implementing cooperative security scheme where peers cooperate in checking malicious peers which is presented in [8]. Peers cooperate by informing affected peers when malicious block is found to protect them against pollution attack. By having a large number of nodes checking at every point in time, expensive homomorphic hashing can be applied less frequently without significantly weakening the resistance of the scheme to resource adversarial behavior.

Now we will describe the details of how such cooperative security system works. We assume that Peers check blocks with probability P. Blocks that are error free are marked as safe, while blocks that have not yet been checked are kept in an insecure window. Blocks are checked in batches. The batch window is equal to the insecure window. Whenever a node verifies its insecure window, valid blocks are marked as S and the insecure window is reset.

To ensure that valid blocks are back into the system as soon as possible, nodes use a mechanism based on binary batching trees to efficiently identify the malicious blocks [11]. Such batching trees work as follows. A node verifies all blocks in the insecure window using batching. If this test does

not find any malicious block, then all packets are marked as safe. If the batch verification fails, then the insecure window is divided in two halves, which are then checked independently using batching. This process will continue recursively. Corrupted parts are again subdivided in two parts until the individual corrupted blocks are identified and discarded.

VI. DETECTION SCHEME

A. Implementation

Our detection scheme uses MAC scheme which has homomorphic property. Our requirement is that any peer, who wants to upload any block should able to construct a valid MAC tag for that block, if it has valid MAC tags of the source blocks. Any of the homomorphic schemes proposed in [10], [15], and [23] can be used. We are using the SpaceMac scheme proposed in [15].

SpaceMac consists of a triplet of algorithms: Mac (generate a tag for any block under given key), Combine (generates tag for combination of blocks by linearly combining their tags) and Verify (verifies whether a tag is valid for a particular block under given key). The construction and proof of algorithms is given in [8].

Let tg_i be the MAC tags of the source blocks, k_i, generated using the Mac algorithms with key K. Assume that P wants to upload a legitimate block e to its peers. Since e is a legitimate block it belongs to source space. Let $e = \sum_{i=1}^{m} \alpha_i tg_i$, where $\alpha_i \varepsilon Z_q$. Then Mac tag tg can be computed using Combine: $tg= Mac_K(e)$. A MAC tag is an element of Z_q, and the probability of faking a valid MAC tag without knowing the key is $\frac{1}{q}$. To increase the security one can use either multiple tags or can increases the field size. The proper implementation and formulation is presented in [3].

B. Security Analysis

Assume that Pa is the attacker who tries to make P1 accept a corrupted block CB. Pa may pick a small j_1 such that K_{j1} is already disclosed by the source and known to Pa, thus it can forge a valid tag of CB. This strategy fails because P1 will not accept CB if K_{j1} is no longer a safe key. Pa may also pick a large value of j_1, but this only delays the check by P1. Consequently, the only way Pa can make P1 accept CB is to forge a valid MAC tag of CB without knowing the key or using Combine (as CB does not belong to the source space). By the security guarantee of SpaceMac, the probability that Pa successfully make P1 accept a corrupted block is negligible.

VII. IDENTIFICATION SCHEME

A. Implementation

In order to identify and eliminate malicious peers, we need a central authority who manages the peers in the network. In this work, we use S for this task. When a peer P1 detects a corrupted block uploaded by a peer Pa it reports this block to S. S will determine if the report is correct and inform every peer to put Pa into its blacklist.

B. Security Analysis

Assume that Pa is an attacker who uploads corrupted blocks to P1, and that P1 detects the attack and sends a report to S. Pa may strategically pick a small value of j_2 such that by the time the report reaches S, S deems that the report invalid because the shared secret key between Pa and S used to generate the evidence tag of the corrupted block in the report is no longer safe. Since P1 drops any block that have such value of j_2, this means all corrupted blocks sent by Pa with such values of j_2 will be dropped even if they bypass the detection scheme. The only other option for Pa to void the report of P1 is to send along a bogus tag of the corrupted block. This, however, also results in P1 dropping corrupted blocks from Pa even if they bypass the detection scheme. Consequently, it is of Pa best interest to pick valid values for j_2 as well as upload valid evidence tags. If so then Pa has no way to avoid the identification by S when its attack is detected. The probability of successful identification depends on the probability of successful detection of two corrupted blocks within the suspicious window.

VIII. PERFORMANCE EVALUATION

A. Communication Overhead

1. Detection scheme: The overhead of our detection scheme is dominated by the pre distribution of source tags and the online overhead of the tags of each block. Since each MAC tag is a symbol in Z_q the overhead of this predistribution to each peer is $l_1 Nm\lceil \log_2 q \rceil$bits. When uploading a block, a peer sends along l_1 tags, thus the online overhead per block is $l_1 \lceil \log_2 q \rceil$ bits.

For comparison, our overhead per generation is significantly smaller than that of [22]. In [22], the predistribution of hash values of blocks of a generation requires $m\lceil \log_2 q \rceil$ bits. Since the homomorphic hash in [22] requires a large field size, 1024 bits, the overhead of the predistribution for a generation of size 128 is at least 16 KB. (The relative percentage overhead depends on the generation size in byte).

2. Identification scheme*:* The communication overhead of our locating scheme includes the tags carried by each packet, the reporting vectors sent by the nodes, and the announcement sent by the controller. The overhead is dominated by the online overhead of the evidence tags of each block. When uploading a block, a peer is required to send along l2 evidence tags, thus leading to an additional online overhead of $l_2 \lceil \log_2 q \rceil$ bits. Compared to the scheme in [20], our scheme achieves much higher security per byte. For instance, in order to achieve similar security as above, the scheme in [20] needs at least 22 bytes per block.

B Computation Overhead

Computational cost of the Mac and Verify algorithms dominate the Combine algorithm because the identification and detection overhead per block is l_1 and l_2 runs of Mac and Verify. The detection scheme for a packet include verify the integrity using Verify algorithm, generate verification tag for outgoing packet using Combine and computing the helper tag for outgoing packet using Mac algorithm. Source node needs to compute the end-to-end tag using the Mac algorithm and the receiver node needs to verify it by performing another Verify algorithm.

For comparison, our computation overhead per block is more than two orders of magnitude less than those of the schemes that use homomorphic hashes or signatures. These schemes perform expensive modular exponentiations, which incur hundred of milliseconds delay per block even when run on a modern PC.

IX. CONCLUSION

In this paper, we introduce a complete cooperative defense mechanism for P2P network coding based systems. Our defense mechanism decrease the computational times and simultaneously provides both in-network detection and precise attacker identification for P2P systems. Our detection and identification schemes have very low communication and computation overhead, significantly less than other state-of-the-art schemes.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," IEEE Trans. on Iriformation Theory, vol. 46, no. 4, pp. 1204--1216, Jul. 2000.
[2] D. Lun, M. Medard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks;' in Proc. Of IEEE International Symposium on Information Theory, Sept. 2005.
[3] 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. 54, no. 11, Nov. 2005.
[4] Y. Zhu, B. Li, J. Guo, "Multicast with network coding in application layer overlay networks", IEEE JSAC on Rencent Advances in Service Overlay Networks, Vol. 22, no. 1, pp. 107- 120, Jan. 2004.
[5] D. Petrovic, K. Ramchandran, and J. Rabaey, "Overcoming Untuned Radios in Wireless Networks with Network Coding", IEEE Trans. on Information Theory, vol. 52, no. 6, pp. 2649-2657, Jun. 2006.
[6] C. Gkantsidis and P. Rodriguez,"Network Coding for Large Scale File Distribution", in Proc. of IEEE INFOCOM, Mar. 2005.
[7] P. Felber, and E.W. Biersack. "Self-scaling Networks for Content Distribution", In Proceedings of the International Workshop on Self-* Properties in Complex Information Systems (Self-*), Italy, 2004
[8] C. Gkantsidis, and P. Rodriguez. "Network Coding for Large Scale Content Distribution", In IEEE/Infocom, 2005.
[9] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting", ISIT, Yokohama, Japan, 2003.
[10] S. Agrawal and D. Boneh, "Homomorphic MACs : MAC-Based Integrity for Network Coding," in ACNS'09.
[11] M. N. Krohn, M. J. Freedman, and D. Mazieres, "On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution," in SP'04.
[12] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a Linear Subspace : Signature Schemes for Network Coding," in PKC'09.
[13] F. Zhao, T. Kalkert, M. Medard, and K. J. Han, "Content Distribution with Network Coding," in ISIT'07.
[14] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in Info Sciences and Systems, vol. 1, no. 1, 2006.
[15] A. Le and A. Markopoulou, "Cooperative Defense Against Pollution Attacks in Network Coding Using SpaceMac," in Technical Report. [Online]. Available: http://arxiv.org/abs/1102.3504
[16] Anh Le and Athina Markopoulou"TESLA-Based Defense Against Pollution Attacks in P2P Systems with Network Coding" , IEEE, 2011
[17] E. Kehdi and B. Li, "Null Keys : Limiting Malicious Attacks Via Null Space Properties of Network Coding," in IEEE INFOCOM 2009, Rio de Janeiro, Brazil, pp. 1224–1232.
[18] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. S. Shen, "Padding for Orthogonality : Efficient Subspace Authentication for Network Coding," in INFOCOM'11.
[19] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures", Advances in Cryptology, 1998.
[20] Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, "Identifying Malicious Nodes in Network-Coding- Based Peer-to-Peer Streaming Networks," in Mini INFOCOM'10.
[21] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, "On Locating Byzantine Attackers," in Proc. NetCod 2008, Hong Kong, China, pp. 1–6.
[22] C. Gkantsidis and P. R. Rodriguez, "Cooperative Security for Network Coding File Distribution," in INFOCOM'06.
[23] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, "RIPPLE Authentication for Network Coding," in INFOCOM'10.

AUTHORS

Dr Siddaraju is Professor and HOD of CSE Department in Dr Ambedkar Institute of Technology, Bangalore, India. He has 16 years of teaching experience. His areas of interests are Computer Networks, Neural Networks, WSN, etc. He has various National and International publications and conferences.

Chitresha Mehta is pursuing her M.Tech in Computer Science from Dr Ambedkar Institute of Technology, Bangalore. She is undergoing her training under the guidance of Dr Siddaraju. Her current research interest includes Computer Networks, Digital Image Processing, WSN, ect.